FREE DVD

NOW INCLUDES LINUXVOICE

POWER MANAGEMENT

PRO LINUX MAGAZINE

LINUX PRO MAGAZINE

SEPTEMBER 2017

POWER MANAGEMENT

Keep your laptop running longer

Amahi
Stream audio and back up files with this cool home server

4 Fabulous Doc Management Tools

Hack Your Ride
Reading data from your car's diagnostic port

Mark Shuttleworth
Canonical's prince ponders Ubuntu's future

CoffeeScript
And other leading JavaScript alternatives

Googler
Command-line search engine

LINUXVOICE

- Executables and ELF Format
- Network Scanning
- Phipps: Data Protection Laws
- FAQ: Flathub

FOSSPicks
- Calibre eBook Reader
- Tor Browser 7
- KDE Tiling Extension

Tutorials
- Apache Spark
- CherryTree

Issue 202
Sep 2017
US$ 15.99
CAN$ 17.99

HETZNER ONLINE

RYZE ‘N’ SHINE

More parallelization. More power.

Dedicated Root Server AX60-SSD

AMD Ryzen 7 1700X
Octa-Core “Summit Ridge” (Zen)
64 GB DDR4 RAM
2 x 500 GB SATA 6 Gb/s SSD
100 GB Backup Space
30 TB traffic inclusive*
No minimum contract
Setup Fee $131

monthly $ **65**

The ideal solution for highly-parallelizable processes.

Our new Dedicated Root Server AX60-SSD houses the latest generation AMD processor, the octa-core AMD Ryzen 7 1700X, which is based on Zen architecture. Its performance is truly impressive, especially when used for purposes such as virtualization, encryption, and data compression, which require several processor cores.

www.hetzner.de/us

* There are no charges for overage. We will permanently restrict the connection speed if more than 30 TB/month are used. Optionally, the limit can be permanently cancelled by committing to pay $1.30 per additional TB used.

STEVE, O STEVE

Dear Reader,

A recent article in several popular news sources recounts the story of a security robot at The Washington Harbour commercial district, affectionately called Steve, that drove itself into a water fountain. The water feature is at the bottom of a few short steps that start at floor level – without a planter or bench to serve as a casual barricade. The poor little robot was just rolling along, watching things, and the floor dropped out from under it. I wonder if any sight-impaired people have fallen at this spot, although a well-handled cane would presumably give some indication of a change in the vertical dimension. I also wonder if a clear-sighted but absent-minded pedestrian with a tendency to look in all directions and not necessarily straight ahead (like the author of this column) could have eventually fallen prey to the same water trap.

The picture accompanying the article shows the robot, which has a tin-can-like R2D2 morphology, "face" down in the water, attended by a maintenance official, with a few shoppers gathered around snapping pictures. I felt a certain kinship with the robot, knowing that it could have been me.

You are probably wondering why I'm mentioning this, and I'm probably wondering too, except to say that it seems to illustrate how we tend to oversell our technologies. Is this device really a robot in the way we experience them in science fiction, or is it more like a self-driving vehicle that shoots surveillance videos? If you call it a robot, you get to charge more for it, but people have higher expectations. Interestingly enough, high expectations are always welcome in the sales department, so we call it a robot, which means we assign it human characteristics and laugh when it falls face-first into a puddle.

I'm sure someday we'll figure out how to make robots that don't slip on banana peels and fall into water fountains. We're usually about 10 years behind where we say we already are. That's good for high tech news, because we have a steady stream of articles on Boeing Dreamliners catching fire and self-driving cars crashing into bridge abutments.

The real problem is, when we base our decisions on self-fulfilling prophecies, we decide in desperation to invest in something because we're falling behind, when actually, the train hasn't quite left the station yet. Eventually, those decisions start to create the world we thought we were already in, and we wonder how we survived all those primitive afternoons at The Washington Harbor without the blessing of a robot driving around recording videos of the shoppers.

Steve, O Steve, how will we make it without thee?

Joe Casad,
Editor in Chief

LINUX MAGAZINE

WHAT'S INSIDE

If you work from a laptop or other portable device, extending the time between charges is a great way to improve your productivity. The Linux environment includes several utilities that will help you configure and monitor the system's power consumption settings. This month we focus on tools for power management. Also inside:

- **Amahi** – This hearty home server system lets you turn your old computer into a file server, streaming server, or network backup target (page 36).
- **JavaScript Alternatives** – If you like JavaScript but you're looking for a language with fewer issues, try CoffeeScript, Dart, Elm, or TrueScript (page 42).

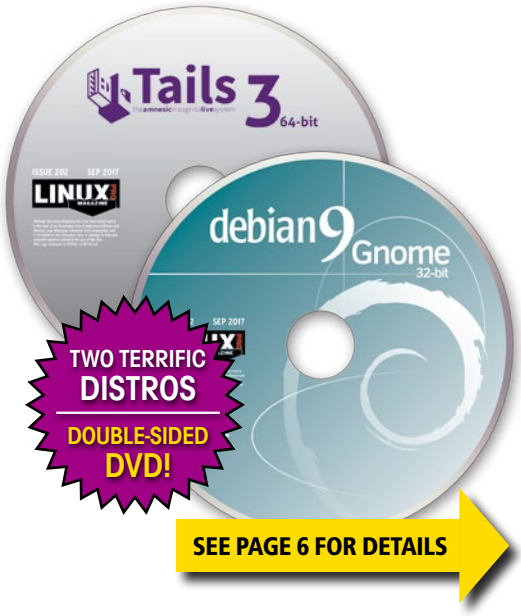
This month's Linux Voice section includes articles on ELF executables, network scanning, Apache Spark, and more (page 67).

SERVICE

NEWS

COVER STORIES

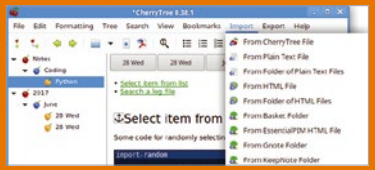
REVIEWS



SEE PAGE 6 FOR DETAILS

IN-DEPTH

LINUXVOICE



On the DVD

TWO TERRIFIC DISTROS

DOUBLE-SIDED DVD!

Debian 9

This issue comes with the recently released Debian 9 "Stretch" 32-bit Live edition. Debian is the quintessential Free Linux, with hundreds of volunteers and over 51,000 total software packages in the project repositories. The Debian distro is the basis for several other important Linux projects, such as Ubuntu and Knoppix, so a new major Debian release is an important event for the entire FOSS community.

Debian 9 comes with better encryption and improved UEFI support. The system includes many application updates and comes with improved security features, such as an X display system that doesn't need to run with root privileges.

Tails 3 (64-bit)

The Amnesic Incognito Live System (Tails) is a privacy-focused Linux designed for anonymous surfing and censorship circumvention. Tails is pre-configured to use the Tor anonymity network and comes with a convenient Tails Greeter application for easy setup.

The latest version is based on Debian 9 and comes with many security fixes and software updates.

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

ADDITIONAL RESOURCES

[1] Debian: *https://www.debian.org/*

[2] Debian Documentation: *https://www.debian.org/doc/*

[3] Debian Wiki: *https://wiki.debian.org/*

[4] Tails Project: *https://tails.boum.org*

[5] Getting Started with Tails: *https://tails.boum.org/getting_started/index.en.html*

[6] Tails Documentation: *https://tails.boum.org/doc/index.en.html*

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

GitHub Announces Open Source Friday Program

After Linux, Git is the second major contribution of Linus Torvalds. Git has become a global phenomenon with GitHub, the planet's largest repository of open source projects. Even Microsoft has moved the source code of Windows to a private GitHub repository. Companies like Google have shut down their own code-hosting platforms and moved their code to GitHub.

GitHub is now becoming active in "promoting" the open source development model. The company has announced celebrating every Friday as Open Source Day.

Mike McQuaid, a senior software engineer at GitHub wrote in a blog post, "Open source software powers the Internet. Anyone using a computer uses open source, either directly or indirectly. Although it has become the industry standard, getting involved isn't always straightforward."

McQuaid disclosed that the company has been running a program internally for the last three years in which they encourage employees to work on some open source project every fourth Friday of the month. Now, they are opening up the program for anyone to get involved with open source.

"Open Source Friday isn't limited to individuals. Your team, department, or company can take part, too. Contributing to the software you already use isn't altruistic – it's an investment in the tools your company relies on. And you can always start small: spend two hours every Friday working on an open source project relevant to your business," wrote McQuaid.

Source: *https://opensourcefriday.com/*

Blog Post: *http://mikemcquaid.com/*

System76 Announces Their Own Distro, Pop!_OS

System76, the hardware vendors that sell Ubuntu PCs, has announced their own Linux distribution called Pop!_OS.

Pop!_OS is based on Ubuntu and uses the Gnome stack and customized themes and extensions. An alpha version of the distribution is available for testing, and the final release is expected in October. At the moment, it seems like a customized version of Gnome, but it's expected that System76 will be doing more tweaking and fine tuning of the distribution to offer a more polished, out-of-the-box experience on their hardware.

The distro seems to be a response to the recent Canonical decision to stop pursuing the consumer PC segment and focus purely on enterprise. The company stopped its in-house desktop project Unity and laid off engineers who were working on those projects.

The company decided to go back to the stock Gnome Shell experience. Since stock Ubuntu offers a very vanilla Gnome experience, System76 wants to offer an experience that they think is well suited for their customers.

In an interview, System76 Community Manager Ryan Sipes told us that the inspiration behind Pop!_OS reflects where the company has been heading for a while.

"We receive a lot of great customer feedback and are really attuned to our customers' needs. We've been focusing our engineering effort on helping our customers achieve more and have a fantastic computer experience. This desire to create the best machines for our customers has driven our recent moves into manufacturing, and those same goals are also behind tailoring a beautiful software experience that is Pop!_OS. Our customers are creators: engineers, developers, 3D modelers, and makers. The work that goes into Pop!_OS will serve them and their needs," said Sipes.

System76 is also planning on bringing hardware design and manufacturing in-house, which aligns very well with the plans to work on their own distribution to offer an Apple-like seamless integration between hardware and software.

You can download Pop!_OS Alpha from their download page (*https://system76.com/pop*).

Linus Torvalds Talks About His Motivation

The Linux Foundation took LinuxCon, one of the biggest open source events to one of the biggest economies, China. One of the biggest highlights of the event was a discussion (*https://www.linux.com/blog/event/lc3-china/20176/6/linus-torvalds-explains-how-linux-still-surprises-and-motivates-him*) between Linus Torvalds and VMware Head of Open Source, Dirk Hohndel, who also happens to be Torvalds' closest friend and fellow scuba diver.

One of the things that Torvalds said continues to impress him is that things continue to improve. "There are things we haven't touched for many years, then someone comes along and improves them or makes bug reports in something I thought no one used," he said. It allows Linux to continue to support very old and basic things that people still care about and use.

CC BY-SA 4.0

Talking about his motivations, Torvalds said that he really likes his job. "I like waking up and having a job that is technically interesting and challenging without being too stressful, so I can do it for long stretches; something where I feel I am making a real difference and doing something meaningful not just for me."

It also seems that despite doing Linux since 1991, he is not bored and has not burned out. He said that once in a while, he does take a break, but every time he takes longer breaks he gets bored and looks forward to going back to doing what he does best – the Linux kernel.

When asked about the future of Linux leadership, Torvalds said that the process they have in place has been working fine for the last 25 years. He agreed that even if they don't have enough maintainers, the group that they do have is very strong, and it continues to grow at a steady pace: "… as these maintainers get older and fatter, we have new people coming in. It takes years to go from a new developer to a top maintainer, so I don't feel that we should necessarily worry about the process and Linux for the next 20 years," said Torvalds.

Debian 9 Stretches Its Wings

The latest release of Debian, code-named Stretch, has been released after 26 months of development. Debian 9 will be supported for the next five years, making it one of the longest supported community-based distributions. Ubuntu LTS is supported for three years on desktops and five years on severs; CentOS is supported for 10 years.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

ClusterHAT • Jeff Layton
Inexpensive, small, portable, low-power clusters are fantastic for many HPC applications. One of the coolest small clusters is the ClusterHAT for Raspberry Pi.

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

Monitor Windows Systems and Linux Servers Thomas Joos
The Microsoft Operations Management Suite provides a comprehensive monitoring solution in the cloud that offers a high degree of flexibility and scalability.

Harden Your OpenStack Configuration Martin Loschwitz
Any OpenStack installation that hosts services and VMs for several customers poses a challenge for the security-conscious admin. Hardening the overall system can turn the porous walls into a fortress – but you'll need more than a little mortar.

The Cuckoo Sandboxing Malware Analysis Tool • Thorsten Scherf
The open source Cuckoo Sandbox malware analysis system investigates malicious software.

Software-Defined Networking with Windows Server 2016 • Thomas Joos
Windows Server 2016 takes a big step toward software-defined networking, with the Network Controller server role handling the centralized management, monitoring, and configuration of network devices and virtual networks. This service can also be controlled with PowerShell and is particularly interesting for Hyper-V infrastructures.

Debian has done some reshuffling with default software: MariaDB has replaced MySQL as the default database, and since the Mozilla and Debian communities have sorted out their trademark dispute, you can now use vanilla Firefox and Thunderbird instead of rebranded Iceweasel and Icedove.

Debian is primarily a leading server operating system, but it's well revered among the desktop users who need reliable and stable systems. Debian is a Gnome distribution, and Stretch comes with a generation-older Gnome Shell 3.22. That's the only downside of using Debian on the desktop; you are often stuck with very old packages.

Looking at the continuous disclosure of security bugs in Linux, Debian is maintaining a very tight grip on security.

"Thanks to the Reproducible Builds project, over 90% of the source packages included in Debian 9 will build bit-for-bit identical binary packages. This is an important verification feature which protects users from malicious attempts to tamper with compilers and build networks. Future Debian releases will include tools and metadata so that end-users can validate the provenance of packages within the archive," said the release announcement.

The X display server no longer needs "root" privileges, which has been a major criticism and security risk.

This is also the first release of Debian that features the modern branch of GnuPG in the *gnupg* package. "This brings with it elliptic curve cryptography, better defaults, a more modular architecture, and improved smart card support. We will continue to supply the classic branch of GnuPG as gnupg1 for people who need it, but it is now deprecated," said the release announcement. This release has also improved UEFI support, which now also supports installing on 32-bit UEFI firmware with a 64-bit kernel. The Debian Live images now include support for UEFI booting as a new feature, too.

Debian is known for wide support for architecture. This release supports 10 architectures, including 64-bit PC/Intel EM64T/x86-64 (AMD64), 32-bit PC/Intel IA-32 (i386), 64-bit little-endian Motorola/IBM PowerPC (ppc64el), and 64-bit IBM S/390 (s390x) for ARM; `armel` and `armhf` for older and more recent 32-bit hardware, plus `arm64` for the 64-bit AArch64 architecture; and, in addition to the two 32-bit `mips` (big endian) and `mipsel` (little endian) for MIPS, a new `mips64el` architecture for 64-bit little-endian hardware.

Debian 9 is available for free download

Serious Stack Clash Bug Affects Linux Systems

Security researchers at Qualys have discovered an old vulnerability in Linux systems that can be exploited executing arbitrary code on system.

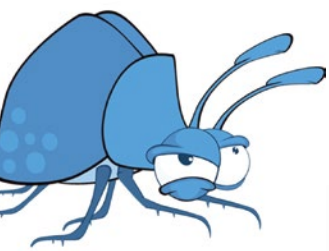
The flaw is related to the way the computer uses the stack (a special memory region). As the programs need more memory, this region grows and can come close to another stack. This vicinity may confuse the program with other memory regions.

"An attacker could use this flaw to jump over the stack guard page, causing controlled memory corruption on the process stack or the adjacent memory region, thus increasing their privileges on the system," Red Hat explained in a security advisory.

The vulnerability has been christened Stack Clash and assigned CVE-2017-1000364 for the Linux kernel and CVE-2017-1000366 for glibc.

Ironically this jump is not a new problem, it has been around for more than a decade now and was exploited earlier in 2005 and 2010. Linux fixed the issue by adding a protection called stack guard page after the 2010 exploit.

"Access to the stack guard page triggers a trap, so it serves as a divider between a stack memory region and other memory regions in the process address space so that sequential stack access cannot be fluently transformed into access to another memory region adjacent to the stack (and vice versa)," wrote Red Hat.



However, Qualys discovered that despite stack guard page protection, stack clashes are still exploitable.

Qualys worked closely with Linux vendors to develop patches. The company also managed to develop seven exploits and seven proofs of concept for this weakness to help write patches.

Bugs © Liudmila Pantelejenkova, 123RF.com

OPEN SOURCE MONITORING CONFERENCE

November 21 – 24
2017 | Nuremberg

PROGRAM ONLINE

Alba Ferri Fitó | Vodafone

James Shubin | Red Hat

Kris Buytaert | Inuits.eu

Holger Koch | DB Systel

Icinga 2 | Distributed Monitoring
Elastic Stack | Modern Logfile Management
Timeseries & Analysis | with Graphite and Grafana
Ansible | Configuration Management

www.osmc.de

Zack's Kernel News

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

By Zack Brown

ZACK BROWN

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**.

Identifying the Oldest Compatible GCC Versions

Arnd Bergmann decided to test all versions of GCC to see how far back he could go and still compile a proper Linux kernel. He reported:

gcc-4.9 and higher is basically warning-free everywhere, although gcc-7 introduces some interesting new warnings (I have started doing patches for those as well). gcc-4.8 is probably good, too, and gcc-4.6 and 4.7 at least don't produce build failures in general, though the level of false-positive warnings increases (we could decide to turn those off for older compilers for build test purposes).

In gcc-4.5 and below, dead code elimination is not as good as later, causing a couple of link errors, and some of them have no good workaround.

In Arnd's ideal universe, GCC 4.5 would be declared too old for Linux and thus unsupported. But he pointed out that there were a bunch of older distributions still widely in use that relied on it.

Going farther back, Arnd reported that with GCC 4.3, "We need a couple of workaround patches beyond the problem mentioned above; more configuration options are unavailable and we get a significant number of false-positive warnings."

Going back to GCC 4.2, Arnd found that whole architectures would no longer work, in particular ARM versions 6 and 7.

He posted a big pile of patches to address all of these problems.

Sebastian Andrzej Siewior pointed out that the currently documented minimum supported GCC version was GCC 3.2, considerably older than the point at which Arnd reported GCC versions to start breaking down. Sebastian also said that H. Peter Anvin had reported that the x86 architecture wouldn't compile properly with any GCC version less than v3.4.

Meanwhile Geert Uytterhoeven reported that he'd been building test versions of Linux using GCC 4.1.2 for years, and while there were a lot of warnings, he'd gotten used to seeing the ignorable ones and was able to sift through them successfully to find any that really mattered.

Arnd, somewhat flabbergasted, asked Geert why on earth he was using such an old compiler for regular work; Geert said, "It's just the cross compiler I built `.debs` [on] a long time ago. As long as it works, I see no reason to upgrade, especially as long as I see warnings for bugs that no one else is seeing."

Meanwhile, Sebastian asked if Arnd's quest for the oldest viable GCC version applied to the ARM architecture only, and Arnd replied, "Clearly having the same minimum version across all architectures simplifies things a lot, because many of the bugs in old versions are architecture independent. Then again, some architectures implicitly require a new version because an old one never existed (e.g., arm64 or risc-v), while some other architectures may require an old version."

But Sebastian said that if an architecture required an older version of GCC, then that had to be some kind of bug in the kernel code that should be fixed. But Russell King pointed out that he used older compilers to build ARM kernels because it eliminated a variable when hunting for bugs. If the kernel version is the only change, he said, then he can be sure that the bugs he finds are kernel related. If he always upgraded his GCC version, it might never be obvious that a given bug was kernel-related or tools-related.

Nearby, Heiko Carstens pointed out that the S390 architecture enforced a minimum GCC version of 4.3 as of two years ago.

The discussion went on hiatus for a bit, until Kees Cook asked if there had been any clear conclusion. He suggested that it might be time to raise the minimum supported GCC version to 4.7. He remarked, "I'm curious what gcc 4.6 binaries are common in the wild besides old-stable Debian (unsupported in maybe a year from now?) and 12.04 Ubuntu (going fully unsupported in 2 weeks). It looks like 4.6 was used only in Fedora 15 and 16 (both EOL)."

Arnd replied, "I think we are better off defining two versions: One that we know a lot of people care about, and we actively try to make that work well in all configurations (e.g., 4.6, 4.7, or 4.8), fixing all warnings we run into, and an older version that we try not to break intentionally (e.g., 3.4, 4.1, or 4.3) but that we only fix when someone actually runs into a problem they can't work around by upgrading to a more modern compiler."

Kees replied, "For 'working well everywhere' I feel like 4.8 is the better of those three (I'd prefer 4.9). I think we should avoid 4.6 – it seems not widely used. For an old compiler … yikes. 3.4 sounds insane to me."

Arnd offered his own detailed recommendation:

I suspect that 4.9 might be the one that actually works best across architectures, and it contained some very significant changes. In my testing gcc-5 tends to behave very similarly to 4.9, and gcc-6 introduced a larger number of new warnings, so that would clearly be too new for a recommended version.

The suggestion of 4.9 or higher is appealing as a recommendation because it matches what I would personally tell people:

If you have gcc-4.9 or newer and you don't rely on any newer features, there is no need to upgrade – with gcc-4.8, the `-Wmaybe-uninitialized` *warnings are now turned off because they were too noisy, so upgrading is probably a good idea even though the compiler is otherwise ok and in widespread use – gcc-4.6 and 4.7 are basically usable for building kernels, but the warning output is often counterproductive, and the generated object code may be noticeably worse. … Anything before gcc-4.6 is missing too many features to be useful on ARM, but may still be fine on other architectures.*

On the other hand, there is a noticeable difference in compile speed, as a 5% slowdown compared to the previous release apparently is not considered a regression. These are the times I see for building ARM `vexpress_defconfig`*:*

```
gcc-4.4: real 0m47.269s user 11m48.576s
gcc-4.5: real 0m44.878s user 10m58.900s
gcc-4.6: real 0m44.621s user 11m34.716s
gcc-4.7: real 0m47.476s user 12m42.924s
gcc-4.8: real 0m48.494s user 13m19.736s
gcc-4.9: real 0m50.140s user 13m44.876s
gcc-5.x: real 0m51.302s user 14m05.564s
gcc-6.x: real 0m54.615s user 15m06.304s
gcc-7.x: real 0m56.008s user 15m44.720s
```

That is a factor of 1.5x in CPU cycles between slowest and fastest, so there is clearly a benefit to keeping the old versions around, but there is also no clear cut-off other than noticing that gcc-4.4 is slower than 4.5 in this particular configuration.

But some people did not want to give up their old GCC versions. Maciej W. Rozycki mentioned that he used GCC 4.1.2 (the same version Geert had mentioned using). And Geert said, "If there's no real good reason (brokenness) to deprecate gcc-4.1, I would not do it. I guess most people using old compilers know what they're doing. My main motivation [to] keep on using gcc-4.1 is that it gives many warnings that were disabled in later gcc versions. I do look at all new warnings, and send patches when they are real bugs, or are trivial to silence."

But Geert acknowledged that the value of such an old GCC version had been diminishing lately, thanks to Arnd's work on reducing the number of compiler warnings with recent GCC versions.

There was some technical back-and-forth between Arnd, Geert, and Maciej. Finally Arnd recommended:

How about this approach then:

To keep it simple, we update the `README.rst` *to say that a minimum gcc-4.3 is required, while recommending gcc-4.9 for all architectures …. Support for gcc-4.0 and earlier gets removed from* `linux/compiler.h`*, and instead we add a summary of what I found, explaining that gcc-4.1 has active users on a few architectures. We make the Makefile show a warning once during compilation for gcc earlier than 4.3.*

Kees and Geert both said this would be fine with them, and the thread ended.

The issue of minimum supported GCC versions is more crucial than one might think. It's not just about kernel developers hunting for kernel bugs or about Linux having bragging rights for supporting 10-year-old compilers. There are a lot of enterprise systems still chugging along out in the world that for one reason or another can't be fully upgraded. Maybe the engineers who understood the system have moved on to other jobs;

maybe the source code to crucial binaries is so old that changing to a newer compiler would represent too great a risk. There are plenty of rickety old systems that form the foundation of an entire company. When the Linux kernel finally ends support for the version of GCC they rely on, those companies may be forced to upgrade and risk having to pour resources into fixing breakages they've been praying never to have to deal with. Or worse, they may decide not to upgrade their kernel at all anymore, which would leave them open to increasing numbers of attack vectors. So it's pretty cool that Arnd and others are still working to keep modern kernels compatible with ancient compilers.

Identifying the Oldest Compatible Make Versions

Masahiro Yamada noticed that the oldest officially supported version of GNU Make was version 3.80, which had not worked since Linux 3.18. In fact, he said he himself was the one who had submitted the kernel patch that broke that version of Make.

Instead of reverting his patch, Masahiro suggested updating the documentation to list Make v3.81 as the officially supported minimum version. He preferred this solution partly because it would be a lot of work to redo everything that would need to be redone to support version 3.80 again and because in the three years since his patch broke that version of GNU Make, no one had noticed or complained about the breakage.

Greg Kroah-Hartman had no problem with this, and Michal Marek agreed, as did Jan Beulich. Finally, Linus Torvalds also agreed, saying, "From earlier (unrelated) discussion, I think we have other cases where the 'minimum recommended' may not be true (even gcc – it might be true on some architectures with simple configs, but we've had long-standing known issues with some more complex configurations where the 'minimum' gcc version simply didn't cut it and could generate incorrect code).

"For the GNU make case, you make a strong case of 'it hasn't worked for a while already and nobody even noticed'."

It's interesting to see the difference between the debate over minimum GCC version and minimum Make version. For GCC, there was a lot of back-and-forth, and a lot of effort to patch to kernel such that the oldest possible versions of GCC would still be supported. For Make, none of this took place, most likely because it is a case of the-proof-is-in-the-pudding. That is, if GCC versions older than version 4.6 or so had been broken for years and no one had noticed, I suspect everyone would have been content to set the minimum GCC version at 4.6 and be done with it.

Creating Mount Contexts

David Howells posted some patches to implement a mount context for all filesystem mounts. The context would contain the mount options and some useful binary data and associate it with the mounted filesystem.

In general, this would make it possible for the mount procedure to return more informative error messages. As David put it, "so many things can go wrong during a mount that a small integer isn't really sufficient to convey the issue."

More specifically, a mount context would be able to hold namespace information, making it easier to isolate a mounted filesystem within a virtualized running Linux system.

So far, David said, he had implemented mount contexts for ProcFS and NFS, with ext4 being next on his list.

Jeff Layton was thrilled with this work and offered some minor technical suggestions.

Miklos Szeredi was also thrilled with David's work, but had some more invasive objections to David's design choices. In particular, he said, redoing the mount API needed to accomplish additional things. It needed to distinguish properly among creating a filesystem instance, attaching a filesystem instance into a directory tree, configuring the superblock, and changing the mount properties.

David's work, Miklos said, only partly achieved the separation that the kernel really needed. He summarized the goals as follows:

Why is `fsopen()` *creating a 'mount context'? It's supposed to create a 'superblock creation context'. And indeed, there are mount flags and root path in there, which are definitely not necessary for creating a super block.*

Is there a good reason why these mount-specific properties leaked into the object created by `fsopen()`*?*

Also I'd expect all context ops to be fully generic first. I.e., no filesystem code needs to be touched to make the new interface work. The context would just build the option string and when everything is ready (probably need a `commit` *command) then it would go off and call* `mount_fs()` *to create the superblock and attach it to the context.*

Then, when that works, we could add context ops, so the filesystem can do various things along the way, which is the other reason we want this. And in the end it would allow gradual migration to a new superblock creation api and phasing out the old one. But that shouldn't be observable on either the old or the new userspace interfaces.

David agreed in principle with most of Miklos's ideas. But he felt that some of the distinctions Miklos wanted to enforce were less crucial than others; he also pointed out that his own motivation in doing this work was not so much to redo the mount API as it was to support better namespace features.

Miklos recognized that David couldn't be expected to do everything and suggested, "let's just say, that everything that works now should work the same way on the old as well as the new interfaces."

There was a bit more discussion, and the thread petered out. Generally, when someone attempts to implement a new feature such as a mount context, there's a certain expectation that the person will also clean up the surrounding code and take care of any little nitpicky details that have been waiting for someone to handle. At the same time, sometimes those nitpicky details are just too thorny, and requiring contributors to deal with them would only discourage anyone from updating that particular piece of code. So there tends to be a bit of give and take. Ideally, anyone contributing code to thorny areas would at least make the surrounding issues easier to deal with rather than harder. In this case, it seems as though David's code has made the surrounding code easier to deal with, even if it didn't eliminate the problems altogether. ■■■

Mark Shuttleworth talks about Canonical's next steps

Do Differently

We sat down with Mark Shuttleworth, the founder and CEO of Canonical, to talk about the future of Ubuntu and the company. *By Swapnil Bhartiya*

Canonical has sponsored the popular Ubuntu Linux since the founding of the Ubuntu project in 2004. Over the years, Ubuntu has become one of the most popular desktop operating systems, and Canonical has branched out into several other areas of the IT space. On the enterprise side, the company has emerged as the third major Linux vendor, alongside SUSE and Red Hat.

Over most of its history, Canonical has been a fixture in the consumer space, where it tried to establish Ubuntu as a viable alternative to Windows and *almost* succeeded. Canonical has also taken on big players like Apple and Google for a place in the phone market. The company has launched many in-house projects, including the Unity desktop, the MIR display server, and the Snap package environment.

Eventually, Shuttleworth and other Canonical leaders realized the company was trying to bite off more than it could chew, as it fought on simultaneous fronts with heavyweights such as Microsoft, Apple, Google, and Samsung. Canonical is now restructuring and is getting out of the consumer space to focus on the enterprise market.

"I loved that work, but it had to go," Shuttleworth said. "We have made some tough calls, but we continue to build Ubuntu and a growing business around it effectively. That business is focused on Cloud and IoT."

Earlier this year, Canonical stopped work on the Unity desktop and Ubuntu phone and laid off many of the employees who were working on these projects. Ubuntu will go back to using Gnome as the default desktop environment. A new vision for Canonical also meant new leadership; Canonical CEO Jane Silber stepped down from the position, and Shuttleworth took over the helm.

Companies like System76 have succeeded with selling custom laptops that run on Ubuntu. Even mass-marketer Dell is offering Ubuntu as an option on their high-end systems. It is fair to say Canonical took Linux as close as it has ever been to making a dent in the PC market. So, what went wrong? Did Linux ever have a chance in the desktop space?

"I would say yes," said Shuttleworth. "I think it's clearly been shown that the world wasn't guaranteed to be Windows-only forever. You have Android. You have Mac OS. You have iOS. You have all these other platforms. I think there were two big challenges for Linux. The first was it would never be enough just to do what Windows did. You had to change the game. That leads to the second problem, which involves building community consensus on how you change the game. Almost anything we did would be criticized by those other communities because they saw Canonical as a threat. And so that's what made it very, very difficult. That's why Linux ultimately is so fragmented. Lots of different ideas and underlying fragmentation in the competitive landscape make it a bit difficult for everybody to back one idea. Android is great: Chrome OS is great. I think these are Linux for consumers. It's easy to criticize them, but you have to give Google credit for what they have achieved."

Despite its recent drift from the desktop, Canonical continues to invest in Ubuntu as a base system for their IoT and cloud/data center market. In fact, the stability of the Ubuntu base system is one reason Canonical can now shift its emphasis to the enterprise. According to Shuttleworth, "One of the great things that I think got done since we started is that we now have a sustainable model for Ubuntu that makes its future very secure. Until recently, there was a fear that

if I got hit by a bus, Ubuntu would go away, whereas now there are enough people who commercially depend on Ubuntu. I think arguably Ubuntu is able to sustain itself securely."

Shuttleworth dismissed the suggestion that Ubuntu might one day adopt a model similar to SUSE or Red Hat, with separate community and enterprise versions. "No, we will not do that," he said. "And the reason is that it ultimately leads to a very fragmented experience. If you look at the Red Hat world, they have Fedora, CentOS, and RHEL. One is essentially a proprietary product. One is an unfinished product, and the other is freely available but unsupported. What that means is that it's very difficult to know what the flow of code is going to be. The whole point of developing Ubuntu was to enable developers to build what they want to build without having to worry about whether or not they are going to buy support. There was one platform to target. It then becomes a business decision as to what portion of their business they want support for. When I talk to enterprise customers, they really don't like the fragmented model. They prefer our model for their own code base."

Shuttleworth added, "We have achieved a point where Ubuntu itself is sustainable without the need for us to divided the world into a free community edition and an enterprise edition. I think that is a very significant achievement."

Data Center and IoT

Canonical's business model offers an enterprise-ready Linux to anyone for no cost, with professional support services available on an as-needed basis. Customers can run hundreds of Ubuntu machines without paying a dime for each node, and, at the same time, they have the option of commercial support for critical systems. But Shuttleworth added that the Linux enterprise environment is about much more than an operating system: "We have to structure the economics of Linux to be competitive with public cloud, and we have to structure economics that work on public cloud. A lot of what we do is about shaping the product and shaping the commercial terms so that they work in a company that is trying to build large-scale infrastructure."

On the private cloud side, Canonical has heavy investments in OpenStack. In addition to working with OpenStack vendors like Mirantis, Canonical has also created a remotely managed private cloud service called BootStack that enables customers to consume private cloud instead of having to make heavy investments in its management. In a nutshell, instead of selling OpenStack, Canonical manages OpenStack clouds for customers.

On the IoT side of the story, Canonical is betting big on the Ubuntu Core platform, which is intended for both cloud and IoT devices. That's where Canonical's continued investment in the Snaps container-like package system comes into play.

Is Canonical Profitable?

Red Hat and SUSE are publicly traded, so we have an idea of what they are actually worth. Red Hat plays in the ballpark of $2.4 billion annual revenues, and SUSE is around the $350 million mark.

As a still-private company, Canonical does not have to show its books to the world. When pressed on the question of whether Canonical is profitable, Shuttleworth was willing to talk about it broadly, "Ubuntu itself breaks even. Then there are other lines of business like OpenStack, public clouds, IOTs."

As Canonical becomes a target for potential investors, the question of its value has led to much speculation. Recent efforts to cut costs and close down consumer-focused projects have served to create a clearer focus for investors.

Shuttleworth is confident any changes to the corporate structure won't change the company's approach to developing and supporting Ubuntu, although the recent restructuring has already led to changes to other projects within Canonical's Linux universe. "The potential decision to go public doesn't compromise key things that people believe in around Ubuntu. It was clear that we have to make sure that our projects were either sustainable, like Ubuntu itself is sustainable, or are commercial. That's why we decided to cut the Unity work – because I could see no model where that would become commercial work. No one was going to ship a phone with Ubuntu."

Will Canonical go public? Shuttleworth did not rule it out, but he emphasized that laying the groundwork for such things can take up to a couple of years. "There is a natural order to that. You don't just go from zero to running a marathon. But there is no immediate pressure for us. We're not actively looking for a round now."

Another trending rumor is that Microsoft plans to acquire Canonical. Shuttleworth rebuffed such reports: "I can definitively say that I have no expectations and that there is no such conversation underway with Microsoft. It's not a topic of discussion between the two companies. The focus between the companies is very specifically to make sure that Ubuntu works extremely well on Azure; on Azure stack, which is their private Cloud; and on Windows itself. That work is driven by shared customer interest. To me, that is very healthy and constructive work, which I'm very happy to be doing."

I asked whether the work with Microsoft involved any licensing deals. "We have never signed any sort of intellectual property patent deal with Microsoft," he confirmed.

The work that Microsoft and Canonical are doing together has lead to Windows Subsystem for Linux (WSL), which allows sys admins to run Ubuntu's version of Bash on Windows, essentially bringing the entire collection of Linux system tools to the Windows environment. Does it pose a threat to the adoption of Linux, because one can now run these tools in Windows without having to install Linux?

"I think there is a very genuine shift at Microsoft in favor of making sure that they support all platforms equally," Shuttleworth said. "This started in the Azure team, but we now see it across the board – Microsoft wants to make sure that their customers can choose to run Ubuntu workloads, and they will have a good experience regardless. I don't think we should worry so much about what does this mean for the Ubuntu desktop. The question is, what does this mean for innovation on GitHub in the open source community and the fact that you can now potentially run all of that stuff on a Windows laptop? I think that will mean more users. More bug reports. More engagement. More eyeballs. More participation. More usage. I think that's all very good." ■■■

openSUSE community gathering

Face to Face

Swapnil shares some highlights of openSUSE Conference 2017. *By Swapnil Bhartiya*

Nuremberg – the city of castles, museums, and the best sausages in the world – is also home to SUSE and openSUSE. This year the city of Nuremberg hosted openSUSE Conference 17 from May 26 to May 28. I attend many conferences, but this was my first openSUSE Conference.

It reminded me of FOSDEM, which I used to attend when I lived in Brussels. This year's conference featured talks, workshops, BoF sessions, and some cool hardware demonstrations.

Douglas DeMaio, Corporate Communications at openSUSE, has been running the show since he joined SUSE in 2014. In an interview, he said that the primary goal of openSUSE Conference is to bring the openSUSE community together and give them an opportunity to get face to face.

"But we are also open to other projects coming in, as we are part of a much bigger open source ecosystem. So, we encourage and sponsor people from different projects," he said.

This year many notable open source projects participated in the conference including Free Software Foundation Europe (FSFE), Nextcloud, ownCloud, Snaps Canonical, AppImage, KDE, GNU Health, and Gnome. SUSE was the primary sponsor of the conference along with ARM, MySQL, Fedora, and ownCloud. Teamix donated the WiFi infrastructure to the attendees.

DeMaio said that the conference has been extremely productive and resulted in many effective collaborations and projects. One example is KDE's LTS release. During a previous openSUSE Conference, the openSUSE community sat down with the KDE community members and expressed their concerns about keeping up with fast moving KDE software on Leap. The two communities reached a solution where KDE announced their LTS release so that Leap users can continue to use a stable version of KDE. Another notable example is collaboration between ARM and the openSUSE community that lead to the first 64-bit operating system for Raspberry Pi.

The Sessions

The first keynote of the conference was delivered by FSFE president Matthias Kirschner, who talked about the lessons learned from Munich's LiMux deployment, a project that is seen as an ideal use case for the open source community. He talked about the challenges that the free software community faces when dealing with public administration.

OpenQA is one of the gems of the openSUSE world, which is now also being used by the Fedora community. Stephan Kulow, openSUSE release manager, gave an interesting presentation on how OpenQA works. Another interesting

talk was a session by Richard Brown, chairman of the openSUSE Board, where he talked about his favorite topic – how rolling releases are the future of Linux distributions.

Thorsten Kukuk, Distinguished Engineer, Senior Architect of SUSE, talked about a new container-based distribution called Kubic, which is a free of cost community version of SUSE MicroOS (Container as a Service Platform – CaaSP). The project is available on GitHub and is available for testing and feedback.

Thomas Hatch, CTO of SaltStack, delivered a very funny talk (although his jokes failed to crack up the German audience) around why he moved from Arch Linux to openSUSE Tumbleweed. He praised Tumbleweed for how well tested and "beautiful" it is and argued that the six-month release cycle used by many distributions is waste of everyone's time. He stated that they needed to install an operating system for a small cloud deployment and ended up deploying openSUSE.

Another highlight of the day was a cool demonstration of an openSUSE-powered robotic vehicle by Simon Lees, Senior Software Engineer at SUSE. He used open source components including ODROID-C1 and AlaMode, running openSUSE Linux.

As someone who covers containers and cloud, I was interested in a talk delivered by Wolfgang Engel, Operations Engineer at SUSE Linux Products GmbH. He talked about Package Hub, a place to offer open source applications to SLE customers. In an interview, Engel said that, with the popularity of Docker containers, there is a possibility of also offering containers via Package Hub.

The next morning, I took the subway and walked to the venue. Hatch was back on stage, and he delivered a keynote speech about new concepts in the data center and how to move beyond the old patterns of configuration management and infrastructure to what's required to survive in a world of distributed systems.

There were many talks around YaST, but the most interesting talk was by Simon Peter, the creator of AppImage. He delivered a thought-provoking talk about the benefits that openSUSE technologies like Open Build Service can bring to application developers. Richard Brown has been extremely critical of AppImage-like solutions, but after listening to Peter's talk even Brown changed his mind and told me that he is open to using AppImage-like solutions.

Zygmunt Krynicki, Technical Lead in UES Commercial Engineering at Canonical, talked about bringing the competing app distributing and packaging format Snaps to openSUSE.

Nextcloud founder Frank Karlitschek talked about data breaches in open source cloud solutions, in the context of increasing attacks by hackers and state sponsored agents. His focus was on how open source platforms like openSUSE and Nextcloud can empower users in ensuring security and privacy of their data.

As a display of healthy competition, the conference also hosted Holger Dyroff, VP Sales & Marketing, Co-Founder at ownCloud, who delivered a thought-provoking talk about the need of product management at open source projects. Dyroff later said that the talk actually led to asking more questions than answering them. There is a tightrope walk between open source projects and products based on them, and Dyroff tried to walk that rope.

On the last day of the conference Aleix Pol, Vice President at KDE eV Board, talked about how the KDE project gets things done; he gave a glimpse of the process that the KDE community has in place to ensure timely releases.

There was also a lot of action happening outside the sessions where people were hanging out in the beer garden and talking about projects and work.

This was a very productive and enjoyable event, and I am looking forward to the next openSUSE Conference. That event is tentatively being planned to take place in Prague, which is the second biggest hub for SUSE and openSUSE outside of Nuremberg. ■■■

Tools for checking ACPI battery data

Fully Charged

Linux offers some useful command-line tools that leverage the ACPI power management framework to display detailed data on the status of a laptop battery.

By Frank Hofmann

The quality of life for any laptop owner depends on the battery maintaining enough charge to operate off the power grid. Eventually, however, you will see a message informing you that the charge has reached a minimum threshold, and you have to save your data quickly or else switch to mains mode to continue working while the battery charges in the background.

The data for the warning message come from the Advanced Configuration and Power Interface (ACPI) [1] [2]. ACPI, which first appeared in 1996, has now replaced its predecessor, Advanced Power Management (APM); the current version 6 dates from April 2015. (For more information on APM and ACPI check out TuxMobil [3] and the ACPI how-to [4].)

ACPI used to be a separate component in Linux, but the Linux kernel developers now maintain ACPI functionality within the framework of the Unified Extensible Firmware Interface (UEFI) [5]. Linux comes with some command-line tools that let you read information on battery status and health. For Debian and its derivatives, you'll need the `acpi`, `acpid`, and `acpi-support-base` packages.

Up to Linux 2.6.26, ACPI information was found on the proc filesystem [6] below `/proc/acpi/` [7]. The developers have now reorganized the filesystem and have moved the information to `/sys/class/power_supply/`.

Checking on Power

Figure 1 shows power data for a Lenovo ThinkPad X 250 running Debian 8. The device has two batteries, one built-in and one replaceable, so the first command in

```
frank@fehmarn: ~
File  Edit  View  Search  Terminal  Help
frank@fehmarn:~$ ls /sys/class/power_supply/
AC  BAT0  BAT1
frank@fehmarn:~$ ls /sys/class/power_supply/BAT0/
alarm           energy_full_design  present         uevent
capacity        energy_now          serial_number   voltage_min_design
capacity_level  manufacturer        status          voltage_now
cycle_count     model_name          subsystem
device          power               technology
energy_full     power_now           type
frank@fehmarn:~$ cat /sys/class/power_supply/BAT0/status
Charging
frank@fehmarn:~$
```

Figure 1: Using the sysfs pseudo filesystem, you can determine the available power supplies and discover detailed information on the power components.

AUTHOR

Frank Hofmann works on the road – preferably from Berlin, Geneva, and Cape Town – as a developer, trainer, and author. He is also the coauthor of the *Debian Package Management Book* (*http://www.dpmb.org*).

Figure 1 reveals three power supplies: `AC` power, as well as the first and second batteries (`BAT0` and `BAT1`). The following command in Figure 1:

```
cat /sys/class/power_supply/BAT0/status
```

outputs the contents of the `.../BAT0/status` file, which shows the status of `BAT0`. In this case, the command reveals that `BAT0` is charging. The other possible messages are `Discharging` and `Fully charged`.

You can always obtain some basic information by parsing individual files in the `/sys/class/power_supply/` directory. But finding and interpreting the data can cause some effort. A number of utilities make life easier and serve up the data in a human-readable format. These utilities include Upower [8], Acpi [9], and Acpitool [10], as well as the Intelligent Battery Monitor, Ibam [11].

All four programs provide information on the various power sources. Upower uses the `--enumerate` switch (short form: `-e`) (Listing 1); Acpi has two options: `--ac-adapter` (`-a`) and `--battery` (`-b`) (Listing 2).

Acpitool provides a compact overview (Listing 3). Whereas Upower only lists the individual energy sources, Acpi and Acpitool provide more details.

If the output in the terminal does not appeal to you, most Linux desktop systems provide a tool for monitoring power usage (Figure 2).

Details

If you call Upower with the `--show-info` (`-i`) option and the name of the power source, it displays all the matching details. The `--dump` (`-d`) switch lists the details for all the power sources.

LISTING 1: Upower with enumerate

```
$ upower --enumerate
/org/freedesktop/UPower/devices/line_power_AC
/org/freedesktop/UPower/devices/battery_BAT0
/org/freedesktop/UPower/devices/battery_BAT1
/org/freedesktop/UPower/devices/DisplayDevice
```

LISTING 2: Acpi Power Info

```
$ acpi --ac-adapter --battery
Battery 0: Charging, 50%, 01:00:01 until charged
Battery 1: Unknown, 75%
Adapter 0: on-line
```

LISTING 3: Overview in Acpitool

```
$ acpitool
Battery #1  : Unknown, 75.35%
Battery #2  : Charging, 50.98%, 00:59:05
AC adapter  : online
Thermal info: <not available>
```

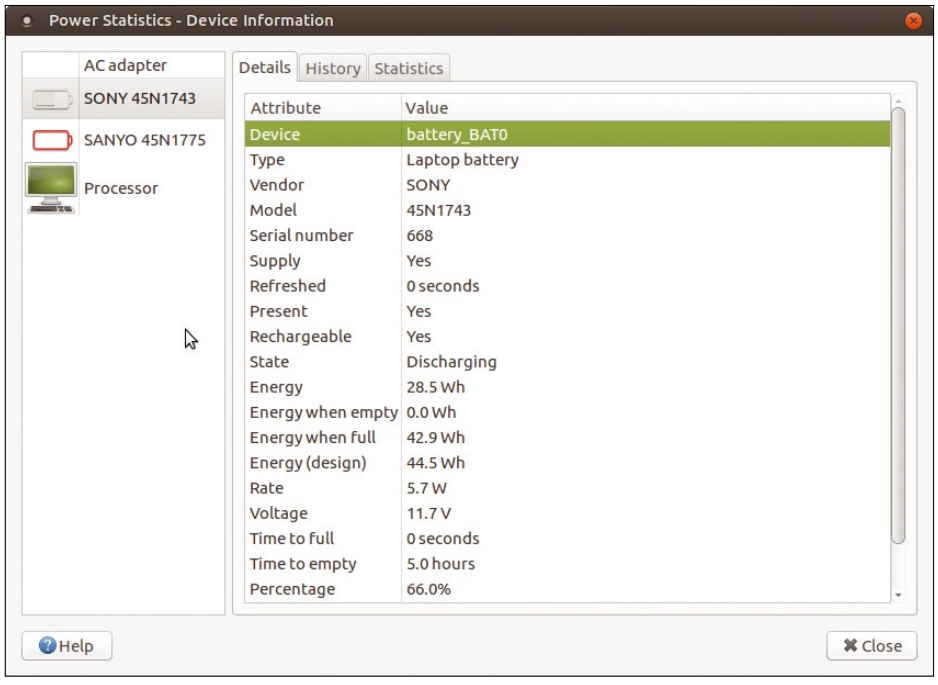


Figure 2: Checking the battery state in Mate's Power Statistics tool.

LISTING 4: Details in Upower

```
$ upower --show-info /org/freedesktop/UPower/devices/
  battery_BAT1
    native-path:          BAT1
    vendor:               SANYO
    model:                45N1777
    serial:               27945
    power supply:         yes
    updated:              Tue May 9, 2017 11:24:13 CEST
                          (1 seconds ago)
    has history:          yes
    has statistics:       yes
    battery
      present:             yes
      rechargeable:        yes
      state:               Charging
      warning-level:       none
      energy:              34.2 Wh
      energy-empty:        0 Wh
      energy-full:         64 Wh
      energy-full-design:  71.28 Wh
      energy-rate:         21.177 W
      voltage:             11.566 V
      time to full:        1.4 hours
      percentage:          53%
      capacity:            89.7868%
      technology:          lithium-ion
      icon-name:          'battery-good-charging-symbolic'
    History (charge):
      1494321853  53,000  charging
    History (rate):
```

Listing 4 shows the results for the second battery in the device. In addition to the manufacturer, the model, the serial number, and the type – a lithium-ion battery in this case – Upower lists more specifications that are useful for everyday operations. Among other things, it shows the battery's charge capacity (only around 90 percent here). Designed for a capacity of 71Wh, the battery now only manages 64Wh due to its advanced age. The `time to full` line shows that it will take around 90 minutes to fully charge the battery.

The `acpi` command provides less detail but at least enough information for a quick overview. Call `acpi` with the `--details` option (`-i`) enabled. The command in Listing 5 uses grep to limit the output to the information for the first battery (`Battery 0`).

For a complete overview of all power sources, call `acpi --everything` (short form `-V`) or `acpitool -Be`. In addition to the charge status of the batteries, the output includes the value of the temperature sensor, as well as information about the cooling system (Figure 3). You can retrieve the necessary information using the options `--thermal` (short form `-t`) or `--cooling` (short form `-c`).

LISTING 5: Quick Overview in Acpi

```
$ acpi -i | grep "Battery 0"
Battery 0: Charging, 56%, 1:18:07 AM until charged
Battery 0: design capacity 6127 mAh, last full capacity 5502
           mAh = 89%
```

LISTING 6: Diagnosing Battery Problems

```
$ acpi -s
Battery 0: Unknown, 4%
Battery 1: Charging, 75%, charging at zero rate -- will
           never fully charge.
```

LISTING 7: Ibam: Checking Battery Time

```
$ ibam --battery
Battery time left:           0:19:12
Charge time left:            1:34:48
Adapted charge time left:    1:34:48
```

The `--show-empty` (`-s`) option tells `acpi` to show you devices that are not currently working. You can thus identify defective or improperly connected system components. The output in Listing 6 indicates that something is wrong with two batteries.

Lifetime and Charging Duration

The Ibam tool calculates both the remaining lifetime and the probable charging time. Listing 7 shows a program call with the `--battery` option, which returns a remaining battery life of 19 minutes and forecasts a charging time of around 90 minutes.

For graphical output, combine Ibam with Gnuplot [12]. Ibam always stores the time values for the charge state locally below `~/.ibam/`. The `--plot` option evaluates this data. You will see a

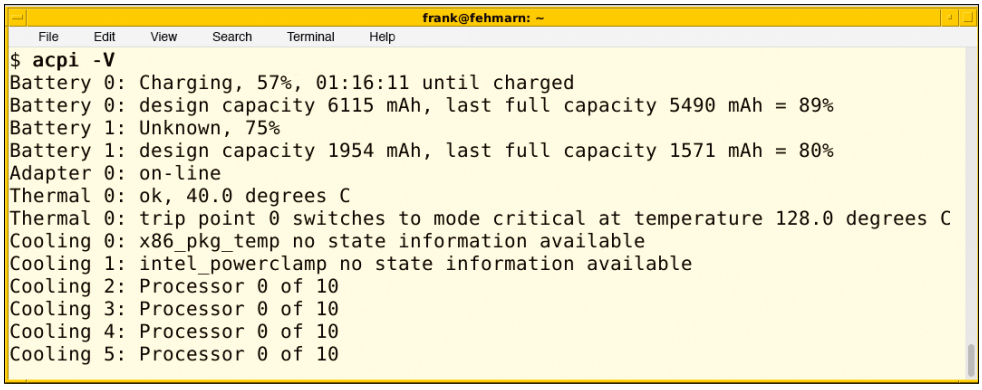


Figure 3: In addition to the information about power sources, Acpi provides data on the temperature sensor and cooling system.

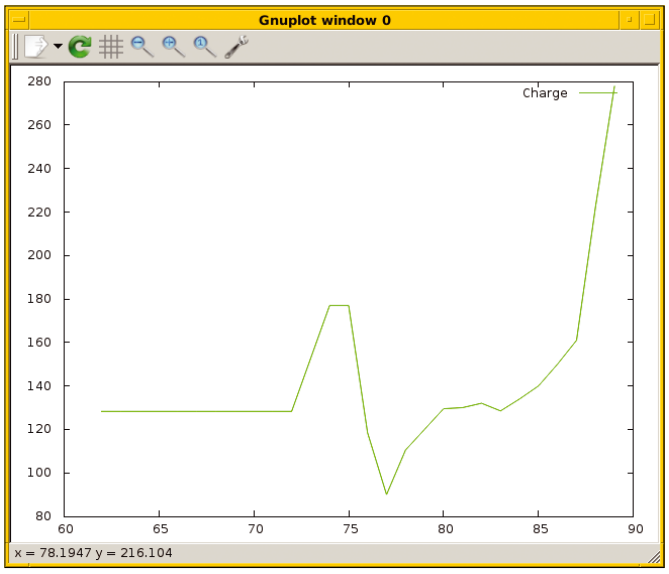


Figure 4: **The Ibam tool stores the values from each call, letting you output the data as a chart.**

Gnuplot window displaying the charge state (Figure 4), where the x axis represents the recorded charge state and the y axis shows the time.

Conclusions

Four tools, Upower, Acpi, Acpitool, and Ibam let you keep track of your laptop's battery health state using the ACPI power management framework. Experience shows that Acpi and Upower, in particular, deliver precise, reliable values. ■■■

ACKNOWLEDGEMENTS

I would like to thank Werner Heuser, Arne Wichmann, and Justin Kelly for their appraisal and suggestions in the course of writing this article.

INFO

[1] Linux ACPI project: *https://01.org/linux-acpi*

[2] Acpi.info: *http://www.acpi.info*

[3] Heuser, Werner. "Power Management on Linux Laptops and Notebooks – APM, ACPI, PMU": *http://tuxmobil.org/apm_linux.html*

[4] Hogbin, Emma Jane. "ACPI: Advanced Configuration and Power Interface": *http://www.tldp.org/HOWTO/ACPI-HOWTO/*

[5] UEFI: *https://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface*

[6] Proc filesystem: *https://en.wikipedia.org/wiki/Procfs*

[7] ACPI tips: *http://www.thinkwiki.org/wiki/How_to_make_ACPI_work*

[8] Upower: *https://upower.freedesktop.org*

[9] Acpiclient: *https://sourceforge.net/projects/acpiclient/*

[10] Acpitool: *https://sourceforge.net/projects/acpitool/*

[11] Ibam: *http://ibam.sourceforge.net*

[12] Gnuplot: *http://www.gnuplot.info*

Tools for optimizing Linux power management

The Frugal Penguin

Linux offers several tools that help users analyze the power consumption of hardware components. *By Frank Hofmann and Mandy Neumeyer*

A computer consists of many electronic components. In addition to the motherboard with the processor (CPU) and the graphic chip (GPU), the list includes random access memory (RAM), drives and storage media, card readers, a hard disk controller, sensors, and network interfaces. In the case of mobile devices, the display, keyboard, and touchpad all run on the same battery.

All these components need power. Modern CPUs have different power-saving mechanisms that significantly affect the power consumption and thus the battery life. This article studies some tools that monitor and manage power usage for the system. These tools will give you the information you need to adjust the necessary settings for optimal power usage.

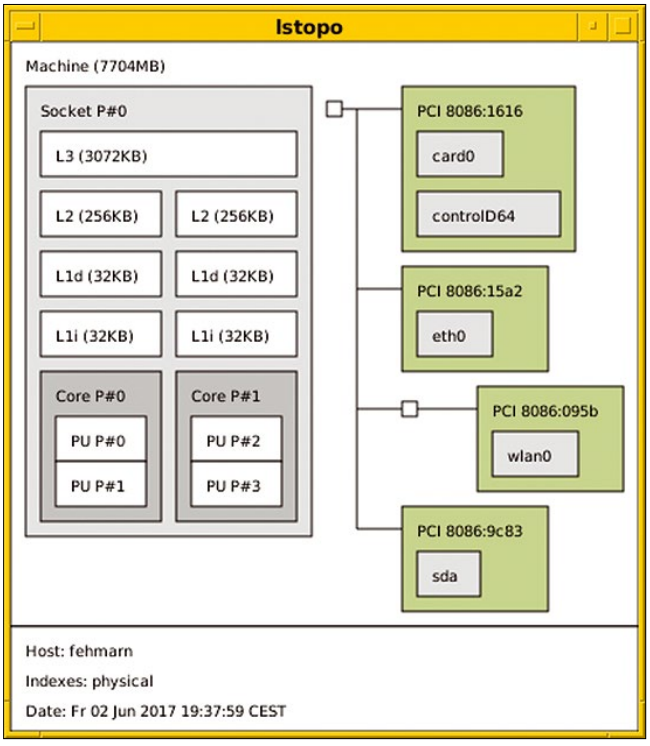


Figure 1: Lstopo shows the components within a laptop.

I2C

The components of an electronic system need a reliable shared communication channel. On many systems, that communication channel is the I2C or I2C bus (also known as the Inter-IC bus). Raspberry Pi DIY aficionados may already have worked with the I2C bus directly. The bus itself dates back to the 1980s and was originally invented by Philips engineers to support communication between individual components.

The I2C bus only uses two conductors, not including ground and the supply voltage. The two lines are *SDA* (*data*) and *SCL* (*clock*). The SDA conductor is used to transfer the data, and clock pulses are sent on the SCL conductor. All devices in the system are attached to these two lines. Each connected device has its own address, by which it can be addressed during communication (Figure 2).

The bus distinguishes between master and slave devices. The master/slave setting defines the right to communicate on the bus: The master can initiate communication, and slaves can only respond. The master sends a byte address to the I2C system and states whether it wants read or write access to it. The connected devices check the start condition sent with the address to see if the master is talking to them. Then the corresponding component responds and the actual data exchange occurs. At the end of communication, a stop condition releases the line [16].

Power Consumption

Power consumption depends on the electronic components enabled and installed in the device and on the programs running on the computer [1]. A small tool by the name of Lstopo [2] visualizes the information on electronic components running on the system (Figure 1). Lstopo is part of the *hwloc* package of hardware tools that is pre-installed or available in package form for many Linux systems. Communication between the individual components is via the internal I2C bus (see the box entitled "I2C").

Tasks such as image and video processing, cryptography, data exchange, synchronization, and data transfer (I/O) are often computationally expensive and therefore expensive for power consumption. Also, kernel bugs can have a massive effect on power usage, as dealing with the leap second showed in 2012.

Laptops are typically more efficient than desktop PCs: The components installed in these devices take a more frugal approach. For example, a 2.5-inch SATA hard disk for a laptop will consume between 3 and 5W, whereas a 3.5-inch desktop HDD will need 10 to 20W of power. A modern SSD, however, is content with just 0.2 to 2W. The values depend on the activity: When idle, the disk requires less power than when reading or writing data.

The processor is one of the biggest power hogs. Tuning the settings can quickly achieve up to 30 percent savings. The Cpufrequtils [3] command-line tool lets you conserve power by controlling the CPU frequency scaling daemon. Cpufrequtils comes with plugins for desktop environments, such as XFCE [4]. You also need to look at the power consumption in idle mode. The system, including all active applications, wakes up quickly from the suspend to disk (STD) or suspend to RAM (STR) modes (see the box entitled "Suspend Modes").

An external graphics card that only computes when needed consumes about 10 percent less. Test measurements show that the choice of graphics driver can also save about 5W.

Directly managing the display brightness, so you don't use more energy than you need for the display, can quickly add another 20 percent to your power savings account. If you do not need WiFi and switch the chip off, you'll save another 10 percent. Bluetooth helps you realize at least 2 percent savings, and a properly controlled fan can save up to 4 percent.

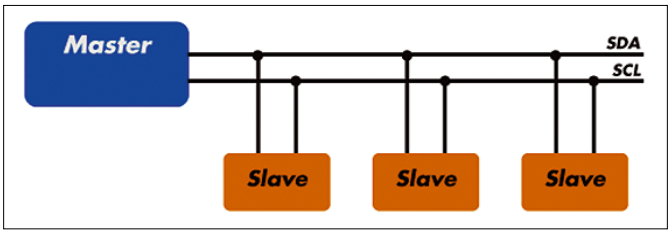


Figure 2: **Circuit and devices on the I2C bus.**

SUSPEND MODES

The purpose of a suspend mode is to back up the computer's current execution state and reduce power consumption by hardware components you do not currently need. When the system wakes up, the state is reloaded, so that the system can proceed with the next instruction. Three modes available in Linux are:

- Suspend to RAM (STR for short, or *stand-by suspend*) keeps the RAM active
- Suspend to Disk (STD, or *hibernate*) swaps out the RAM content to the hard disk
- Suspend to Both (STB) saves the machine to swap space then invokes a STR

During STR, the CPU runs in the internal S2 mode, and the system's power consumption drops to the extent that it can sleep for several days in this state. Only the RAM retains power. You can wake the computer from this mode very quickly, because it does not need to read any data from the hard disk. You only need to make sure the device does not entirely run out of power, in which case you would lose your unsaved changes.

In STD (mode S3), all data is dumped on the swap partition or swap file of the hard disk. The system then shuts off completely. The next time the system is powered up, it does not completely reboot but loads the saved state.

STB, which became available in kernel 3.6, uses the in-memory copy of the state upon awakening, provided that the battery is not completely discharged. If the battery is discharged and the copy of the state in RAM is lost, the system reverts to the state saved on the swap partition.

To enable this function on Debian and Ubuntu, access the *pm-utils* [17] or *uswsusp* [18] packages. *uswsusp* includes the `s2disk`, `s2ram`, and `s2both` tools.

The Powerstat tool [5] will help you determine the power consumption for a laptop. The program output in Figure 3 shows an average power consumption of just under 10W for the test computer.

CPU Performance States

The original x86 processors always ran at full power. As of the 486 generation, with the Intel 486DX4 from 1994, various power-saving mechanisms were introduced to let the system reduce the consumption in an idle state. Other chip manufacturers introduced similar features, such as ARM with the ARMv4, StrongARM, and ARMv5 and Sun with the microSPARC in the mid-1990s.

Within these idle states was a gradual division of several *C-States* [6]. C0 always means full activity and is the same for all manufacturers; higher C-states – C1 to C7 [7] – correspond to sleep state levels. The supported modes depend on the CPU manufacturer and the specific processor model [8]. Measurements show that a processor in the C-States C6 and C7 does not need more than 0.5W.

You can read the maximum usable C-State for Intel CPUs from the `/sys/modules/intel_idle/max_cstate/` file (Listing 1). You can set a maximum C-State as an additional parameter in the Grub configuration. The entry `intel_idle.max_cstate=1` defines a maximum of C1 (Listing 2).

C-State and Power Consumption

The power consumption depends on the current power saver mode, the clock frequency, and the number of currently active CPU cores. The system dynamically adjusts depending on the situation and requirements.

The faster the processor is clocked, and the more cores it uses at the same time, the higher the power consumption will be. The Turbostat tool from the Debian *linux-cpupower* package

LISTING 1: Reading the Max C-State

```
$ cat /sys/module/intel_idle/parameters/max_cstate
9
```

LISTING 2: Setting the Max C-State

```
linux /vmlinuz-3.16.0-4-amd64 ... processor.max_cstate=1
```

shows you exactly what is going on. Get the details about power consumption by setting the `-S` and `--debug` options (Listing 3). The three last columns of the output are of interest for power consumption – these settings represent the consumption per CPU (`PkgWatt`), consumption per core (`CorWatt`), and the GPU share (`GFXWatt`). In the case of server processors, instead of `GFXWatt`, you can specify `RAMWatt`, which returns the power consumption of the RAM modules (DIMMs).

ACPI

The Advanced Configuration and Power Interface (ACPI) [9] lets you intervene actively with the system's power management. Tools such as Acpitool [10] and Acpiclient [11] inhabit the ACPI framework. To reduce the power used by unnecessary interfaces and devices, you can turn to GUI power management tool for your desktop interface, or use a tool such as Hdparm or Sdparm for hard drives.

Mobile devices such as laptops benefit from tools in the *laptop-mode-tools* package [12]. A graphical interface lets users change many parameters with a single mouse click (Figure 4). For example, if you are not connected to a wireless network, simply disable the wireless chip (*Enable modules wireless-ipw-power*, *wireless-iwl-power*, or *wireless-power*) to easily save the power these components otherwise require. For information on the parameters, check out the *laptop-mode-tools* project documentation, as well as the wikis for Arch Linux [13] and Ubuntu [14]. And see the article on ACPI elsewhere in this issue.

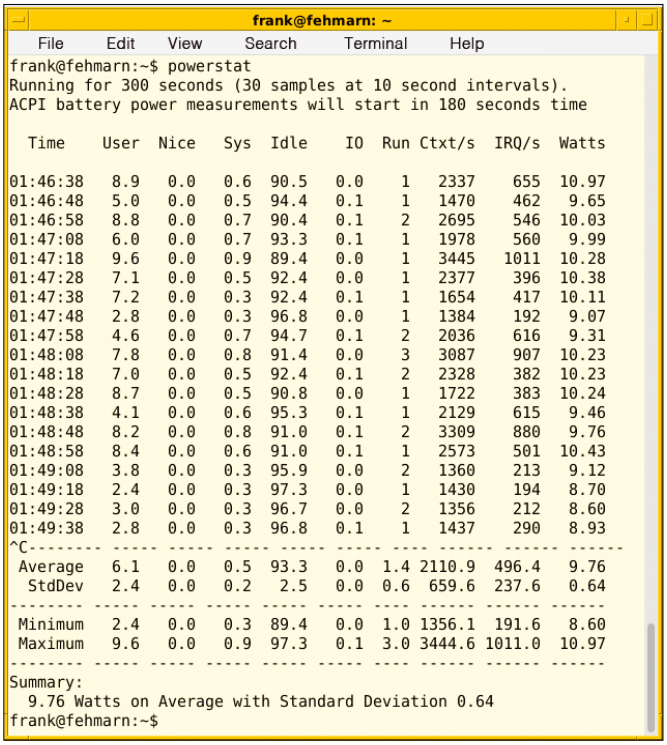


Figure 3: **Power consumption measured while writing this article.**

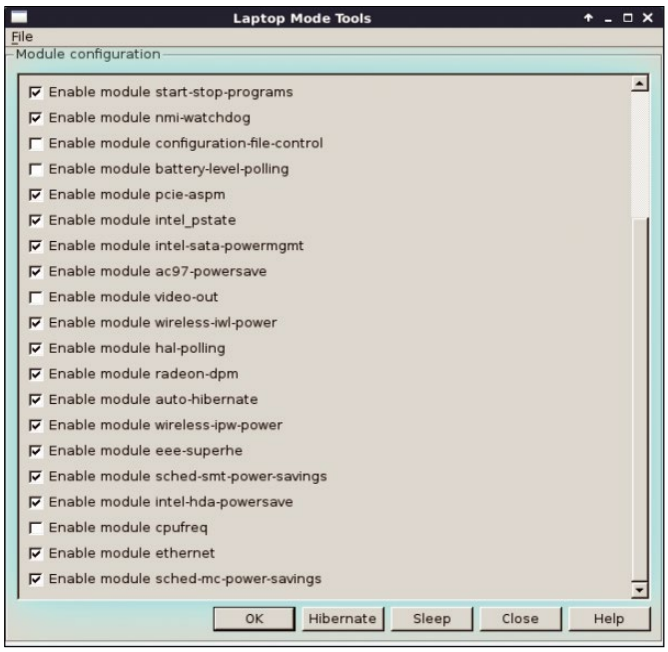


Figure 4: **The graphical interface of *laptop-mode-tools*.**

LISTING 3: Turbostat Details

```
# turbostat -S --debug
turbostat version 4.12 5 Apr 2016 - Len Brown <lenb@kernel.org>
CPUID(0): GenuineIntel 20 CPUID levels; family:model:stepping 0x6:3d:4 (6:61:4)
CPUID(1): SSE3 MONITOR SMX EIST TM2 TSC MSR ACPI-TM TM
CPUID(6): APERF, DTS, PTM, No-HWP, No-HWPnotify, No-HWPwindow, No-HWPepp, No-HWPpkg, EPB
CPUID(7): No-SGX
...
 Avg_MHz Busy% Bzy_MHz TSC_MHz IRQ SMI CPU%c1 CPU%c3 CPU%c6 CPU%c7 CoreTmp PkgTmp GFX%rc6 GFXMHz Pkg%pc2 Pkg%pc3
      66  3.46    1921    2295   0   0   9.49    0.00    0.00   87.05      47      47    0.00       0   77.33    0.00
Pkg%pc6 Pkg%pc7 Pkg%pc8 Pkg%pc9 Pk%pc10 PkgWatt CorWatt GFXWatt
   0.00    0.00    0.00    0.00    0.00    1.98    0.44    0.00
[...]
```

Powertop [15] gives you details of the system load. The program comes with five views: *Overview*, *Idle stats*, *Frequency stats*, *Device stats*, and *Tunables*. You can press Tab or Shift + Tab to change tabs. The device statistics answer the question of which devices are consuming too much power: In addition to the battery discharge rate, Powertop shows the extent to which a device or interface is busy (Figure 5).

Conclusions

Linux is not known for maximizing the theoretical maximum battery life of a mobile device, but current kernels and modern desktop environments with integrated power saving mechanisms have helped Linux narrow the gap with other systems. The Linux environment includes several useful tools you can use to configure the power configuration manually.

Keeping unneeded components and processes active costs energy without giving the user anything in return. You can find these power hogs with the current crop of tools and put your laptop on a battery diet. ■■■

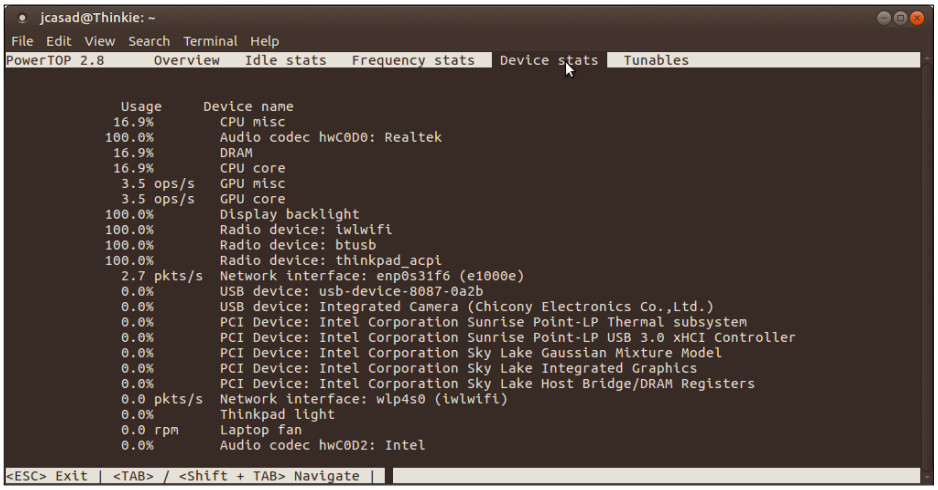


Figure 5: **Powertop shows how much each component is being used.**

AUTHOR

Frank Hofmann works on the road – preferably from Berlin, Geneva, and Cape Town – as a developer, trainer, and author. He is also the co-author of the Debian package management book (*http://www.dpmb.org*).

Mandy Neumeyer has lived in Cape Town for nine years and likes to travel around the world. She works in tourism and is currently building up an extra source of income as a digital nomad.

INFO

[1] Travers, Matthew. CPU Power Consumption Experiments and Results Analysis on Intel i7-4820k, uSystems Research Group, Newcastle University, *http://async.org.uk/tech-reports/NCL-EEE-MICRO-TR-2015-197.pdf*

[2] Lstopo (from hwloc): *https://packages.debian.org/stretch/hwloc*

[3] Processor clocking: *https://wiki.ubuntuusers.de/Prozessortaktung/*

[4] Xfce4-cpufreq-plugin: *http://goodies.xfce.org/projects/panel-plugins/xfce4-cpufreq-plugin*

[5] Powerstat: *https://sourceforge.net/projects/powerstat*

[6] C-States, ACPI: *https://en.wikichip.org/wiki/acpi/c-states*

[7] Power management states: *x://software.intel.com/en-us/articles/power-management-states-p-states-c-states-and-package-c-states#_Toc383778910*

[8] "What are CPU C-states and how to disable them if needed?": *https://gist.github.com/wmealing/2dd2b543c4d3cff6cab7*

[9] The Linux ACPI Project: *https://01.org/linux-acpi*

[10] Acpitool: *https://sourceforge.net/projects/acpitool/*

[11] Acpiclient: *https://sourceforge.net/projects/acpiclient/*

[12] Laptop-mode-tools: *https://github.com/rickysarraf/laptop-mode-tools*

[13] Arch wiki on laptop-mode-tools: *https://wiki.archlinux.org/index.php/Laptop_Mode_Tools*

[14] Ubuntu wiki on laptop-mode-tools: *https://wiki.ubuntuusers.de/laptop-mode-tools/*

[15] Powertop: *https://01.org/powertop*

[16] "Using the I2C Bus": *http://www.robot-electronics.co.uk/i2c-tutorial*

[17] Pm-utils: *https://pm-utils.freedesktop.org/wiki*

[18] Uswsusp: *http://suspend.sourceforge.net*

Document management for the small office

Defying Chaos

Even in a small office, countless letters, email messages, and PDFs arrive daily. Document management systems help you avoid drowning in the flood of documents. *By Erik Bärwaldt*

It's been more than a decade since the proclamation of the paperless office, with special document management systems (DMSs) proposed as the tool to manage arbitrary documents without miles of shelving. DMSs typically operate as client-server applications that users can access by means of a database back end.

Most of these DMS applications are at home in medium to large enterprises and are hopelessly oversized for use in small home offices. Successfully using a DMS becomes even more difficult when the requirements include Linux support. Nevertheless, I searched for DMSs for Linux workstations that relieve the strain on small offices without time-consuming training and permanent maintenance. In my search, I've taken a look at Krystal DMS, LogicalDOC, Paperwork, and Referencer (see also the "Not Tested" box).

Requirements

Ideally, the DMS should reproduce the workflow of a document starting with its creation, through its entire lifecycle, to final deletion. The DMS should handle not only printed documents, but also files that exist electronically in various formats (e.g., email).

The DMS does not just act as an archiving system for quick access to archived documents using keywords, date stamps, or other attributes. It also needs to optimize the flow of information in

NOT TESTED

OpenKM [1] was intended to be the fifth candidate in this test. Although it has a Linux version – including a community release and commercial and cloud packages – in our lab, the software proved to be extremely recalcitrant, with no usable installation routines for small offices or for less savvy admins, as well as no current documentation. Instead, you are expected to install the required packages manually, individually, and separately (including a Tomcat application server, a MySQL database, and applications such as ImageMagick and Ghostscript), followed by editing of complex configuration files – again by hand.

Although the manufacturer provides help documents, they are hopelessly out of date and caused attempted installations on current Linux distributions to fail. Some recent Linux versions also no longer offer the required packages. For Fedora and Red Hat Linux, the documentation refers to OpenOffice Suite 3.1.1, which was released August 31, 2009, and has seen countless new releases in the meantime.

The Debian and Ubuntu documentation also is out of date: It describes the configuration for the long-since-replaced SysVinit system but does not tell you how to handle the service units of the current systemd session manager. The Apache web server configuration no longer works as described, either. For all of these reasons, I did not test OpenKM for this article.

organizations by introducing distribution mechanisms for eligible recipients, document linking, and access monitoring.

A modular design should also ensure trouble-free processing of documents in third-party applications, including popular office suites or Enterprise Content Management (ECM) systems.

Multiplatform capability to allow the use of the client on mobile devices like tablets is also becoming increasingly important. Today, this also includes cloud connections for access to documents in the DMS independently of stationary IT. Last, but not least, regulatory requirements for archiving also need to be met wherever you are in the world.

In the Small Office

Small offices do not typically require large DMSs that are usually difficult to install and configure and require regular maintenance on top. However, alternatives for small offices also need to handle input sources, such as printed documents, files of different formats, and stored email. Ideally, they should also include a scan engine that enables reading and text recognition of printed originals. Keywording and other storage functions are in the DMS's domain, as well as interfaces for the major office suites (see Table 1).

Less relevant in small DMS solutions, however, is sophisticated mechanisms for granting rights and modules for interacting with major league ERP and ECM solutions. Also, the ability to use an app to access the DMS software from a mobile device, such as a tablet or smartphone, is less important in this working environment. What proves to be as important in the service portfolio solution for small offices and individual workstations, however, is easy installation and configuration of the software.

The Trouble with OCR

Reliable detection of scanned originals remains problematic on Linux. If the DMS applications do not have their own OCR modules, users are forced in many cases to rely on third-party solutions. In a *Linux Magazine* lab, we tested an OCR team consisting of Tesseract and gImageReader. The solution turned out to be technologically mature and therefore usable (see the "Tesseract and gImageReader" box).

TABLE 1: Overview DMS Functions

	Krystal DMS	LogicalDOC	Paperwork	Referencer
Modular design	Yes	Yes	Yes	No
Localization	Yes*	Yes	Yes	Yes
Client-server architecture	Yes	Yes	No	No
Web-based interface	Yes	Yes	No	No
Scanning module	Yes*	Yes*	Yes	No
Multiple sheet scanning	Yes*	Yes*	Yes	No
OCR module	Yes (external)	Yes (external)	Yes	No
Import function	Yes	Yes	Yes	Yes
Export function	Yes*	Yes	Yes	Yes (external)
Viewer	Yes	Yes	Yes	No
Indexing and searching	Yes	Yes	Yes	Yes
Version history	Yes	Yes	No	No
Comments	No	Yes	No	Yes
Cloud connection	No	Yes	No	Yes
Mobile apps	Yes	Yes	No	No
Link to CMS systems	No	Yes	No	No

*Available only in the commercial versions.

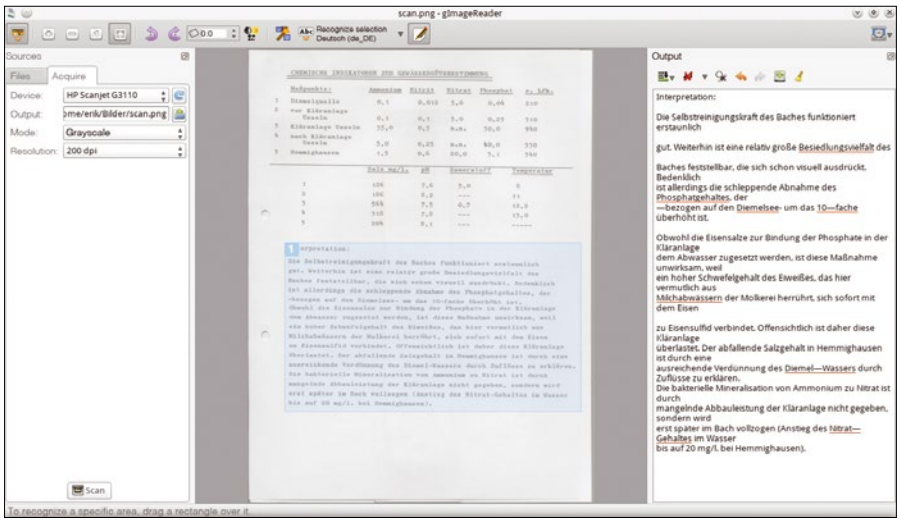


Figure 1: Tesseract is best suited for digitizing printed originals.

TESSERACT AND GIMAGEREADER

Hewlett-Packard (HP) worked on the Tesseract [2] text recognition engine between 1985 and 1995. For 10 years, development lay dormant because HP had abandoned this market segment. In 2005, Google acquired the software and, after revising the code, released it to the developer community as free software under the Apache license. Subsequently Tesseract spread throughout the Linux universe. Thanks to the modular design, Tesseract is also multilingual, and even German blackletter types are now detected if you have the matching modules in place. Not even foreign languages with many nonstandard characters can pose unsolvable problems for the software.

Because OCR engines are typically command-line-only applications, third parties have developed various graphical interfaces over the years to make the programs easier to use. The GUI environments often cover one or several special engines.

gImageReader [3] has established itself as a relatively unknown front end for Tesseract OCR. In addition to ease of use, it promises a particularly lean design and therefore comes without unnecessary bells and whistles. Both software packages are available in software repositories of the popular Linux distributions. You can thus install at the push of a button on your flavor of Linux, then simply call the graphical front end, which automatically launches the OCR engine in the background, so you can scan originals and launch the recognition process (Figure 1).

Krystal DMS

India's Primeleaf Consulting distributes Krystal DMS [4] in four commercial versions, plus a Community Edition. Compared with the commercial versions, the Community Edition has functional restrictions and does not include support.

The Community Edition manages documents up to 512MB and is more suitable as a test platform. The client-server application can be used with a database back end, included with the Community Edition, via a web user interface; you do not need an additional database.

Although Krystal DMS is certified for Red Hat Enterprise Linux v5.*x*, which is no longer being updated, it also works without problem on recent distributions. Because it is written in Java, users need to install a Java Runtime Environment on the client and use a browser with a Java plugin. The Community Edition for Linux download [5] requires you to register on the manufacturer's website.

Distribution Model

The price for the four commercial versions [6] depends on the number of users. All versions offer a one-year subscription model that includes free email and telephone support. During this period, all updates are free of charge. Pricing for the Xpress Edition ranges from $300 for one user to approximately $800 for five concurrent users.

The Enterprise Edition for medium and large offices costs $8,750 for five concurrent users and reaches a subscription price of $40,000 per year for 100 users. These rates are always for on-premise installations on the user's intranet.

Alternatively, Primeleaf also offers a hosted cloud solution starting with the Standard Edition, in which offices are billed on a monthly or annual basis, and prices are based on the number of users.

First Impressions

After downloading and unpacking the `tar.gz` archive, which weighed in at around 17MB, you first have to change to the newly created `krystalce` subdirectory, then make the `krystaldms.sh` script executable with root privileges using:

```
chmod +x krystaldms.sh
```

Now, simply start the DMS by typing `./krystaldms.sh`.

The shell script indicates successful activation by outputting a short status message to the terminal. At the same time, this step enables the internal database back end and the built-in web server. Now, the document management program can be accessed from any Java-enabled web browser on the intranet. Users enter the URL *http://<IP host address>:8080* and type *ADMINISTRATOR* as the username, with *admin* as the password in the login window that appears. The work area (Figure 2) then appears.

Before using Krystal DMS, you need to configure the system. The DMS comes with sample data that nicely illustrates the use of the system.

The *Control Panel* button top left in the browser window takes you to the configuration options.

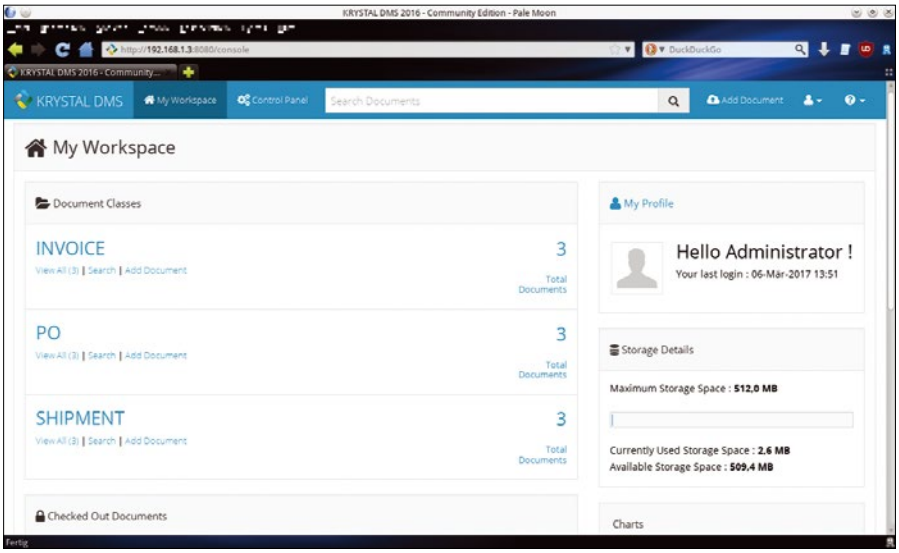


Figure 2: The web interface in Krystal DMS looks cluttered.

The control panel provides most of the important settings in the *Manage Users* and *Manage Document Classes* option windows. Creating one or more users is obligatory, which the admin does in the *Add User* dialog. *Document Classes* refers to the various document groups, which can be compared to individual folders that contain a specific group's documents.

Users can delete the three existing document classes and replace them with their own. You can add new document classes with the help of a dialog that appears after pressing *Add Document Class*.

The class then appears in the control panel, but not in the workspace. For this to happen, you need to assign user-specific rights by pressing the *Manage Permissions* button below the document class in the control panel. After enabling these by pressing *Submit*, the new document class then finally appears in the workspace.

Users can then press the *Add Document* button below the class to add documents in a straightforward dialog. The documents can then be accessed via the corresponding document class and appear in a tabular overview in the workspace.

Point of View

Krystal DMS also lets you view documents. A viewer is integrated into the software for this purpose. Right-clicking on *View Document* in the tabular view displays the document in a separate window. Unfortunately, the formats supported by the viewer are not very useful: The Community Version only displays PDF documents and some image formats, such as PNG files. Simple text files or even ODF documents – or one of the Microsoft formats – are not displayed in the Community Edition or the cheaper versions of Krystal DMS.

If you attempt to view such a document, the DMS throws an error, requiring you to download the file in question. You need the Premium Edition of Krystal with the built-in viewer (Figure 3) to open files in ODF format (LibreOffice or OpenOffice).

Workflow

Krystal DMS offers impressive options that extend far beyond managing

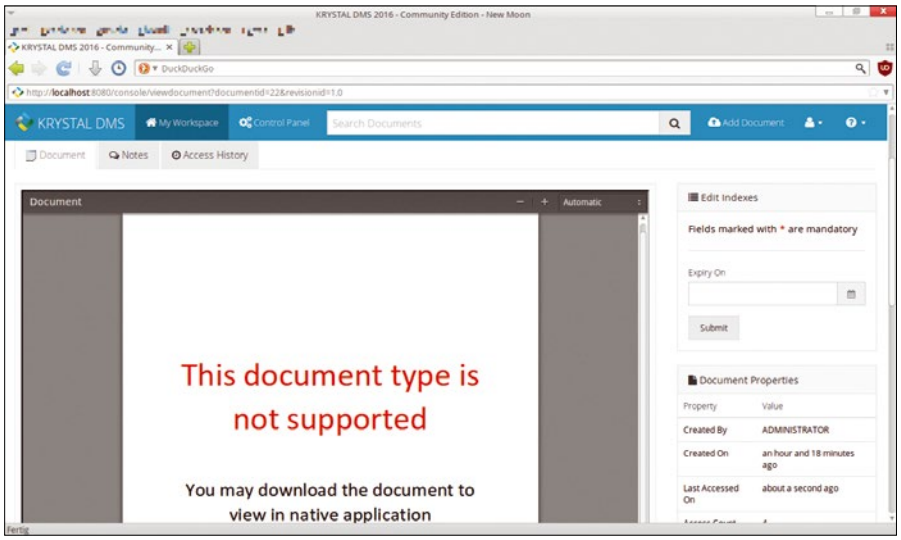


Figure 3: The viewer supports very few formats in the free version of Krystal DMS.

smaller document collections: It indexes documents and captures them automatically with a scanner. An OCR module is included; the program also supports foreign languages, and documents can be provided with expiration dates, which eliminates the need for removing them manually from your databases. However, these functions, as well as the calendar and to-do list, are only available in two versions: Premium and Enterprise.

That said, the Community Edition comes with essential features, such as keywording and appropriate search tools and lets you annotate individual documents. Users do this in the document viewer using the *Bookmark Document*, *Edit Index*, and *Notes* tabs. Additionally, the Community Edition implements a history for each document, which records the latest steps in a specific file.

LogicalDOC

LogicalDOC [7] is firmly established on the market as one of the most widely used DMSs. Several variants of the software exist as client-server applications: In addition to the open source Community Edition [8], there is the Cloud version, as well as the Enterprise and Business versions, which are installed locally. LogicalDOC requires a Java Runtime Environment, as well as a database back end; the recommendation for Linux is MySQL or PostgreSQL.

The manufacturer offers a 30-day trial version, with support available from the website. Neither the manufacturer nor its distribution partners publish either prices or terms and conditions for purchasing the packages. Subscription models including support, and updates are based on the number of users and are usually for one year, as is typical in the industry.

Installation

LogicalDOC is available as a ZIP archive. After registering on the vendor's website, the user receives an approximately 525MB archive as a link to a download page. Packages for virtual environments are available too, as are various add-ons for file viewers, cloud uploads, backup tools, and the integration of an FTP or a CIFS server.

Linux users need to check in advance whether their installation includes Java JDK 8 or newer. You can find out, regardless of your distro, with the `java -version` command. LogicalDOC works both with the OpenJDK environment and with Oracle's original Java package. Additionally, a MySQL server should already be in place on the system.

Ubuntu users also need to make sure that they install the LogicalDOC package with root privileges; type `sudo su` in a terminal and then call the installer with the command:

```
java -jar logicaldoc-installer.jar
```

The routine bundles the package onto your mass storage device (Figure 4) in 13 steps.

Interestingly, the installation routine also queries the paths to third-party applications that display documents as external programs, read text, and convert files. The applications used here are exclusively free software. On most current Linux distributions, the user only needs to adjust the path to the office suite, because LogicalDOC still expects the presence of OpenOffice and offers you the standard installation path for calling the software.

You will generally want to enter the path to the now preferred LibreOffice at this point. LogicalDOC operators can install packages, such as ImageMagick, Ghostscript, Pdftohtml, Tesseract, and OpenSSL, depending on the application scenario, to achieve full functionality.

GUI

LogicalDOC also uses the web browser as the interface. To access the application, you type *http://<IP address of the server>:8080* in the address bar if you are calling the DMS from some other computer on your intranet. On the local computer, access is available by typing *http://localhost:8080*.

After a computer reboot, however, you first need to enable the LogicalDOC server; to do so, pop up a terminal with root privileges and enter:

Figure 4: You can get through the installation dialog for LogicalDOC relatively quickly, despite numerous steps.

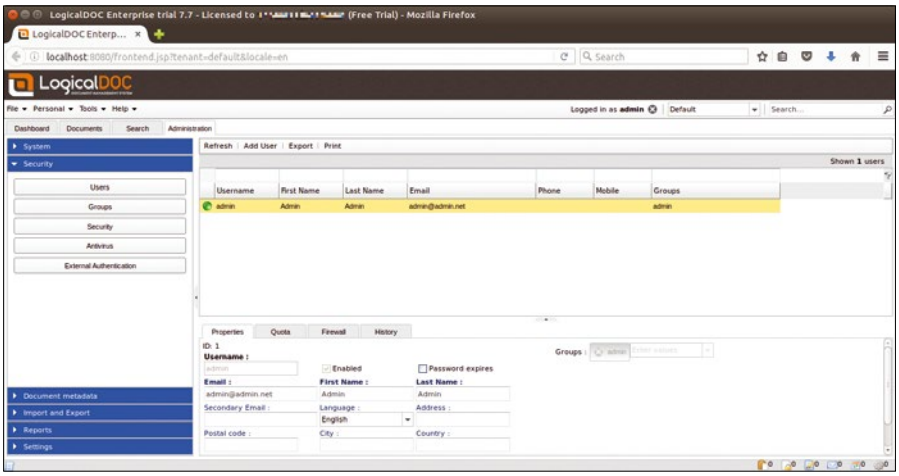


Figure 5: In LogicalDOC, you can create new users in just a few steps.

```
service logicaldoc start
```

In the DMS's login screen, you then authenticate for the first time with *admin:admin*. Additionally, you can change the locale, if needed. A straightforward window with three large display areas and two ribbons arranged horizontally appears in the left-hand panel.

A Question of Settings

In the next step, you get to create the users and groups by clicking at top left on the *Administration* tab in the main window and then clicking the *Security* button. After selecting *Users* in the dropdown menu, a list of users registered with the system appears on the right. The *Add User* button lets you create a new user (Figure 5) and assign the user the necessary rights and appropriate group memberships.

You can create new groups in the same menu and then proceed to assign individual users to the groups. To discover what rights the individual groups and users have and make changes if necessary, click on the current document group in the tree on the left and then open the *Security* tab at bottom right. In the tab, you can check and uncheck options to assign and revoke highly granular group and user rights. Users also inherit the assigned rights; group rights always apply to all members of a group (Figure 6).

In the next step, documents can be imported, scanned, and processed. For that, the DMS offers options the *Documents* tab in the buttonbar at top center in the window. Users can also import complete folders here.

On the left side of the window, the document groups appear in a tree view; below this, the folders are assigned to them in hierarchical order, if needed. Users can right-click on a document group or folder to choose from various management tasks. For example, you can create new folders. With the folders, you also see an empty document table that appears on the right and can now be filled in. At bottom right, several tabs show various properties of the currently selected document.

Search and Ye Shall Find

LogicalDOC has a powerful search function for quickly finding documents; users can access this in the *Search* tab top left in the program window. The software automatically indexes documents it has loaded or scanned based on internal algorithms.

If you type a search term in the input field in the search mask on the program window's left and then click on the magnifying glass icon to the right of it, the DMS searches the current collection and displays a tabular list of the results (Figure 7). Scores show the relevance of each document with respect to the search term as a horizontal bar.

In the results list, you can add more filters, if needed. To perform a more in-depth search, select the *Search in current hits* option and enter a further search term. On the right in the table, the DMS then displays the documents that match all search terms with a new score bar. These can be viewed or edited in a right-click context menu.

LogicalDOC supports users searching for documents with other features in addition to the full-text search: Both tags and various document-specific parameters, including the language setting, also act as search parameters. In the *Folders* selection in the left-hand pane, you can also apply the search function to folder structures. Additionally, you can store your search criteria for repeated searches to avoid having to re-enter all the options.

Dashboard

LogicalDOC also offers a dashboard that gives the user an overview of their activities, details of their used documents, and the workflow. The latter includes interactions with other users on the intranet, and there is also a calendar function and a task manager.

Paperwork

The Paperwork [9] DMS program, released under the GPLv3, is an exception to the rule, because there is only one variant available. The modular application is written in Python and has both a

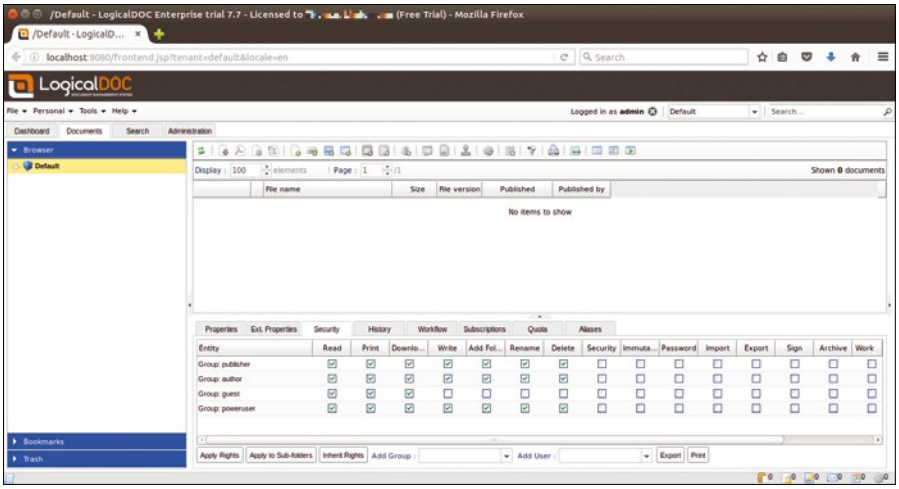


Figure 6: LogicalDOC has a finely tunable rights management system.

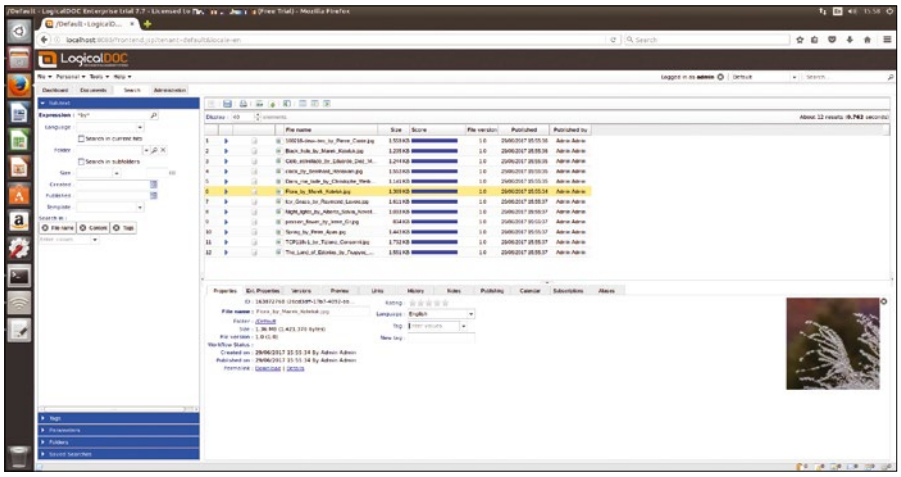


Figure 7: The LogicalDOC search function works very accurately, thanks to sophisticated algorithms.

scanner and an OCR connection; again this module relies on free software. Paperwork is a complete solution for digitizing, tagging, and archiving documents. In terms of requirements, the system is designed for use on single workstations. It requires a scanner that works with Sane [10] and needs Tesseract OCR with the corresponding language modules.

Installation

Paperwork has no built-in installation routine and requires a large number of additional packages. Depending on the Linux distribution, this will sometimes mean expending quite a bit of effort before Paperwork is ready to go. Although the developers provide brief installation instructions for several Linux derivatives on GitHub, they are, however, partly obsolete and therefore not necessarily meaningful. On a freshly installed Ubuntu 16.04, I talked the software into running in a process that involved several steps (see the "Installing Paperwork" box).

If you are missing a starter for the software at the end of the installation, you can launch the program from a terminal by typing `paperwork`. This quickly loads a two-panel window. Scanned documents and documents loaded from your data repository appear on the right, whereas any document-specific attributes, such tags, will appear on the left. First, you will want to call the settings dialog with your scanner switched on, even if this only offers you a few basic options (Figure 8), particularly with respect to the resolution of the scanner.

If your pool of original documents includes many documents with very small fonts, it is advisable to increase the default value to 300dpi. Under certain circumstances, users might want to change the language selection for Tesseract OCR in the selection box.

Usability

Paperwork allows for highly flexible use of documents: When scanning, it will grab a complete stack of originals if you have an automatic document feeder. To do this, press the *Scan* button in the upper right-hand corner of the program window. Paperwork then shows each page in an animation in the window's right pane. In the background, Tesseract is already performing optical character recognition – if you enabled this previously.

The left pane displays thumbnail images of the first page. The `papers` subdirectory, which the software creates in the user's home directory, is used to store the text files; multiple-page documents are stored in a single subdirectory. The text files have a `.words` suffix and can be processed as raw files in any standard editor. But

INSTALLING PAPERWORK

First, you need to install various Python packages on the system:

```
sudo apt install python3-pip python3-setuptools python3-dev
  python3-pil libenchant-dev
```

Then, retrieve Paperwork and store it on your mass storage medium using the Pip package manager for Python modules:

```
sudo pip3 install paperwork
```

Next, use the Paperwork shell to check whether the program can satisfy all its dependencies. The following commands will help you with this:

```
paperwork-shell chkdeps paperwork_backend
paperwork-shell chkdeps paperwork
```

Meet any dependencies still unresolved and, depending on your desktop environment, create a starter for Paperwork in the menu system.

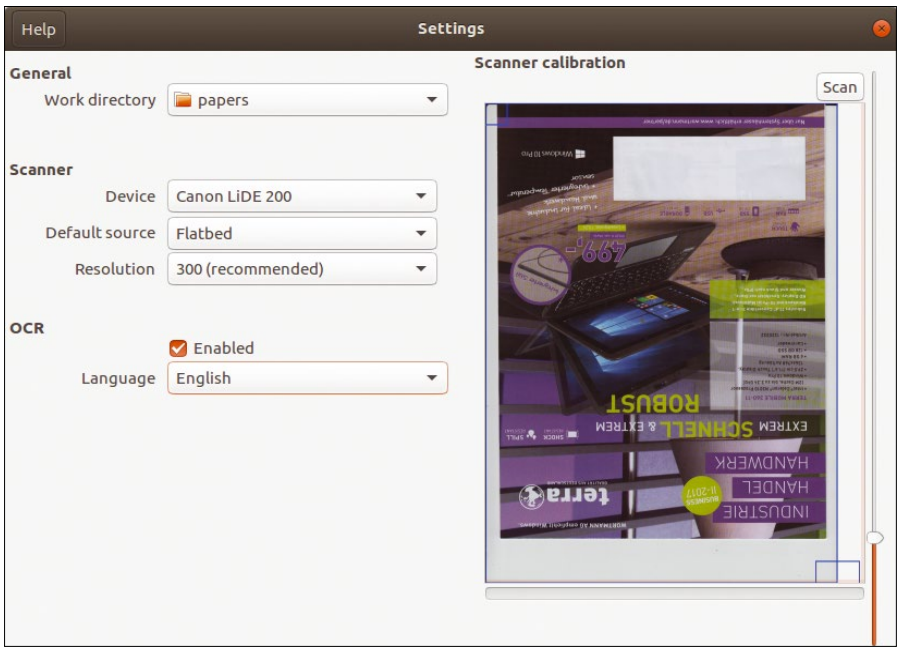


Figure 8: You tweak the Paperwork settings for the scanner in a fairly spartan window.

Figure 9: Manual, but colorful – keywording in Paperwork.

they are primarily used for indexing with keywords.

Paperwork also first runs the OCR program when importing existing files. Note that the software does not support any formats from third-party applications, such as office suites. The exception is the widespread and universal PDF format. If you integrate magazines that are available in this format into the system, the OCR software examines them page by page. Then, Paperwork shows the scaled-down individual pages in a list view in the right windowpane, and the user can magnify these on request.

For print media, which often contains many illustrations and advertisements, it is advisable to turn off predictive text in the *Setup* menu before the OCR step, which not only saves a significant amount of time, it also avoids unusable results, such as those often encountered if the originals use different colors and font sizes.

Indexing and Searching

Once all of your originals have been scanned or loaded, users can assign keywords. To do so, click on the scaled-down document in the program window's left pane and then press the small icon in the right pane. After clicking on the + button, you can define different labels, or keywords. For a better overview, you can assign colors to the labels (Figure 9).

Paperwork always assigns all existing reference terms to documents that you are keywording. To disable the incorrect tags in your document, uncheck and simultaneously enter new ones using the *Additional keywords* input field. After making all specifications, reopen the list display by pressing the back arrow in the *Properties* view. You should now see color indexing for each document.

To find specific data in extensive documents, click on the *Advanced search* button top left in the document and then enter your criteria in a very simple search dialog. You can combine the groups *Date*, *Keyword*, and *Keyword(s)* as needed. Clicking on *Apply* updates the display and lists only the hits that match all search criteria (Figure 10).

An export function is available from the menu button at the top right. You can export the current documents or pages to PNG, JPEG, or PDF files and store them in a directory of your choice. You can also print the current document using the corresponding entry in the same menu.

Referencer

Referencer [11] is a small DMS designed for the Gnome desktop; however, it is not a client-server application, and therefore it is primarily suited to managing smaller document collections on single-user systems. Referencer does not need a database back end; it is found in the repositories of most Linux distributions and installs quickly at the push of a button. Users of other desktops can install and use Gnome Referencer – the dependencies are automatically resolved by the package management tools. For distributions that do not include the program, the source code is available as a `tar.gz` archive [12].

After the install, a launcher should be listed in the Office submenu. Clicking on it opens a simple program window with only one menu and one buttonbar at the top. Below this, you will find an empty space, neatly divided into two windowpanes with another area below it.

The software organizes documents in libraries. To create a collection, drag folders or files selected with the mouse into the right-hand windowpane. Referencer integrates files as links, which offers the advantage of being able to freely choose the file format. A library therefore also houses different file formats, even including multimedia files.

Users should note that Referencer also stores paths in its libraries on the basis of the links to the original documents. It is advisable to put some thought into defining the directory hierarchies before creating the document collection, because Referencer will not

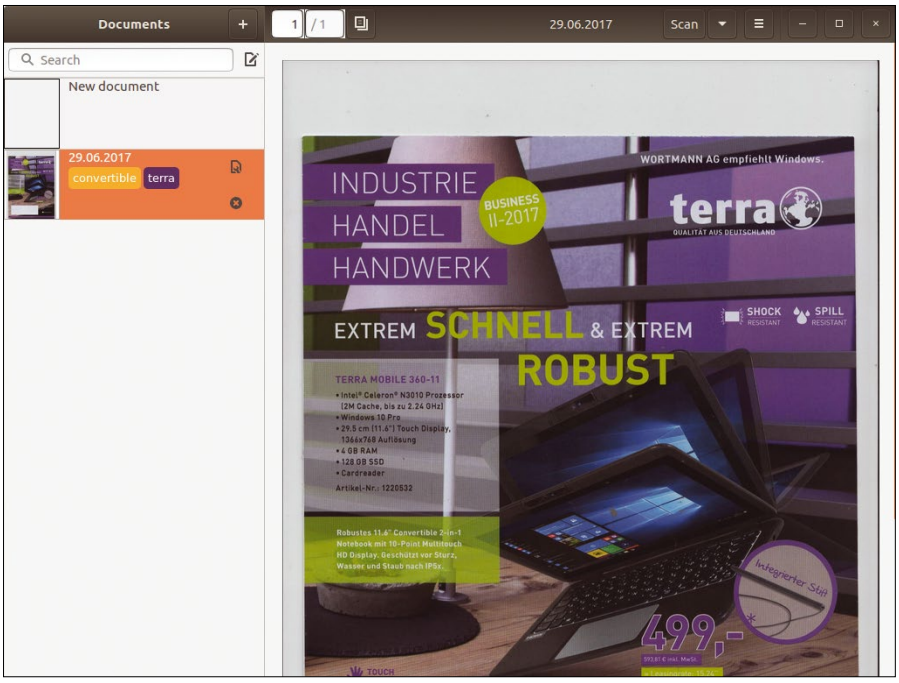


Figure 10: Referenced documents appear with color-coded keywords in Paperwork .

find any documents that you subsequently move to other directories.

Apparently, the software not only lacks a document-tracking mechanism, it also ignores the menu hierarchies when creating libraries and only adds documents from the bottom level of a directory structure in a single operation.

To scan printed originals and for text recognition, Referencer requires third-party programs, because the software does not have its own scanning engine or OCR routine (Figure 11).

Keywords

Once the required documents have been added to the new library, referencing can then begin. To do this, you press the *Create Tag* button to generate a tag manually that you then link to each document. Referencer stores tags in a separate window and transfers them to the left pane. You then select the documents you want to associate with the current tag and call up the context menu by right-clicking. Selecting *Tag* opens the corresponding list, from which you then choose the appropriate tag.

Depending on how many documents you associate with a tag, the font size of the relevant keyword changes in the left window. Documents referenced in this way can be found far more easily, especially in larger collections.

Import and Export

Referencer imports metadata from the commercial DMS EndNote [13], as well as from the free BibTeX literature management system [14]. Also, the tool can import tags from the science database arXiv.org [15], which makes it easier to associate the appropriate metadata with documents in Referencer. Because these databases primarily focus on scientific texts from the fields of mathematics, physics, biology, and computer science, though, the benefits to the small office are limited.

When importing documents, Referencer attempts to retrieve metadata from the database of the Web of Science service [16]. Because you need to register to do this, and registration is not free of charge, error messages tend to appear here. In the Library menu, you can export libraries as BibTeX files to use elsewhere. You can also export any notes you create as HTML files.

In addition to the references, you can also store a comment for each document, if required. The free text box *Notes:* – bottom right in the window – lets you enter and save longer notes using the *Save* function.

Referencer relies exclusively on external programs to display documents. In this way, the program supports far more formats than the internal viewer and will even store file archives in a library. You can then view these with an archiving tool and also unpack if necessary. To view or edit a document, click on the thumbnail view of the file and then on the *Play* button. Clicking on *Open* in the context menu that now appears links up the data with the appropriate application for the file format. In the same menu, you can remove individual files from the library or link them with other keywords.

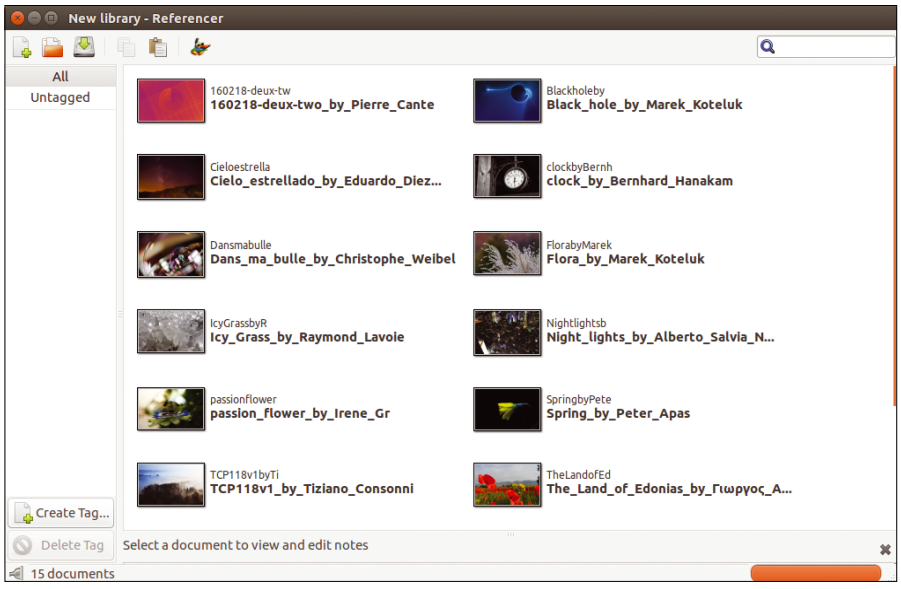


Figure 11: Referencer's very simple interface does not include its own scanning and OCR routines.

Conclusions

The DMSs tested all offered quite impressive functionality, but they focus on different target groups: Whereas Referencer and Paperwork offer good solutions for the small office and home users, Krystal DMS is designed more for medium to large businesses. LogicalDOC is the winner in terms of functionality, but it also requires a longer familiarization period.

Before rolling out a commercial DMS, it is important that you precisely consider which features you need and which you do not. The commercial packages are now so modular, it is difficult to keep track of them, but if you do make a good DMS choice, you can expect significant workload savings in your management and use of documents, which should make it worthwhile introducing such software. ■■■

INFO

[1] OpenKM: *https://www.openkm.com*

[2] Tesseract: *https://github.com/tesseract-ocr/tesseract*

[3] gImageReader: *https://sourceforge.net/projects/gimagereader/*

[4] Krystal DMS: *http://www.krystaldms.in*

[5] Krystal DMS download: *http://www.krystaldms.in/resources/downloads/downloadForm.php?fileid=4*

[6] Krystal DMS versions: *http://www.krystaldms.in/products.php*

[7] LogicalDOC website: *https://www.logicaldoc.com*

[8] LogicalDOC Community Edition: *https://sourceforge.net/projects/logicaldoc/files/sources/*

[9] Paperwork: *https://github.com/jflesch/paperwork/*

[10] Sane: *http://www.sane-project.org*

[11] Gnome Referencer: *https://launchpad.net/referencer*

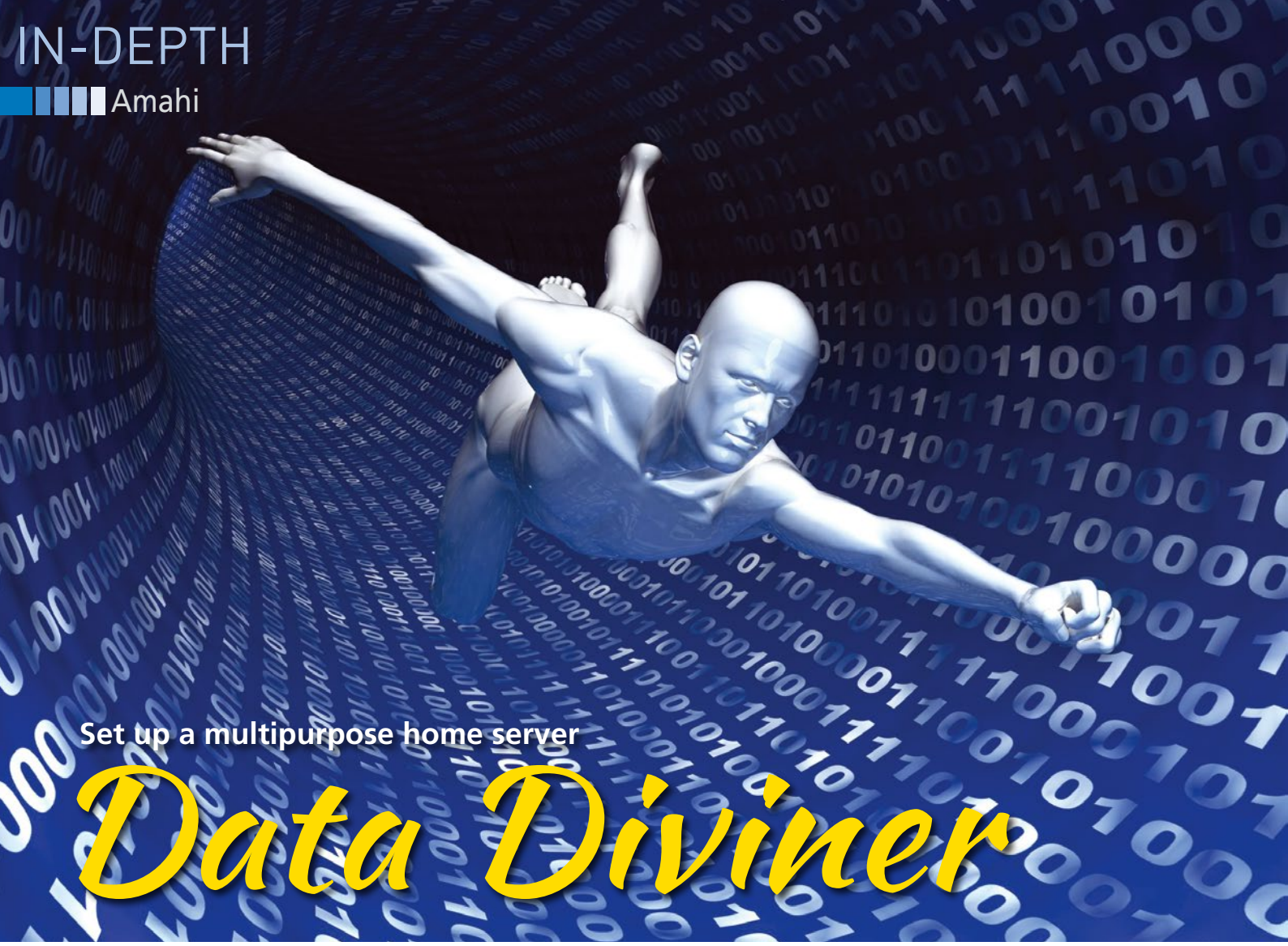
[12] Referencer source code: *http://icculus.org/referencer/*

[13] EndNote: *http://www.adeptscience.de/products/refman/endnote/endnote.html*

[14] BibTeX: *http://www.bibtex.org*

[15] arXiv.org: *https://arxiv.org*

[16] Web of Science: *http://wokinfo.com*



Set up a multipurpose home server

Data Diviner

Install a home server to make your data omnipresent. *By Mayank Sharma*

These days a server isn't something that you'll find only inside a corporate environment. With the proliferation of computing devices inside a typical home, it makes sense to create your own always-on server for your household. You can use your home server as a central media hub to stream videos, music, and pictures to other devices connected to your network. Another popular use for such a server is as a dedicated seed box for downloading and seeding content (legally, of course) such as Linux ISOs or software from the Internet Archive.

Amahi [1] is a free, open source home server solution based around Fedora Linux. It's flexible and customizable, easy to install, and has lots of pluggable apps that you can install with a single click to extend the features of the server to suit your needs. For example, Amahi includes a DLNA server and several streaming servers to broadcast all kinds of multimedia to compatible players and devices. It also includes Greyhole for pooling disks into a unified network storage medium that you can then use to create shares that can be accessed via the Samba

INSTALL FEDORA SERVER 23

Amahi is best set up on a headless server that's always powered on, especially because it's serving essential services like DNS and DHCP. The Amahi developers recommend implementing it on top of a minimal Fedora Server installation. Grab the 64-bit netinstall ISO [2] and burn it onto an optical disk or transfer it onto a USB disk. Boot from the media and begin the installation. Fedora's Anaconda installer uses the hub-and-spoke model and will bring you to the Installation Summary screen from where you'll have to configure the various essential aspects of your particular installation. Anaconda should automatically pick the correct keyboard and time and date settings.

Next up is partitioning. Feel free to partition the disk as you see fit. You can also let the installer partition the disk automatically but make sure you don't use the LVM partitioning scheme as that allots a majority of space to the `/home` partition, which isn't used by Amahi. Instead use the Standard partitioning scheme to make sure the Amahi shares and web apps get the maximum space. Similarly, if you define the partitions manually, make sure the new mount points you add are Standard partitions. Secondly, in the *Software Selection* section you'll have to toggle the *Minimal Install* checkbox. Finally, while the installer copies files to the hard disk, you get the option to define a user and set a password for the root user. Amahi developers advise you to not define a password for the root user and instead add a user and toggle the checkbox to give Administrative permissions to this user.

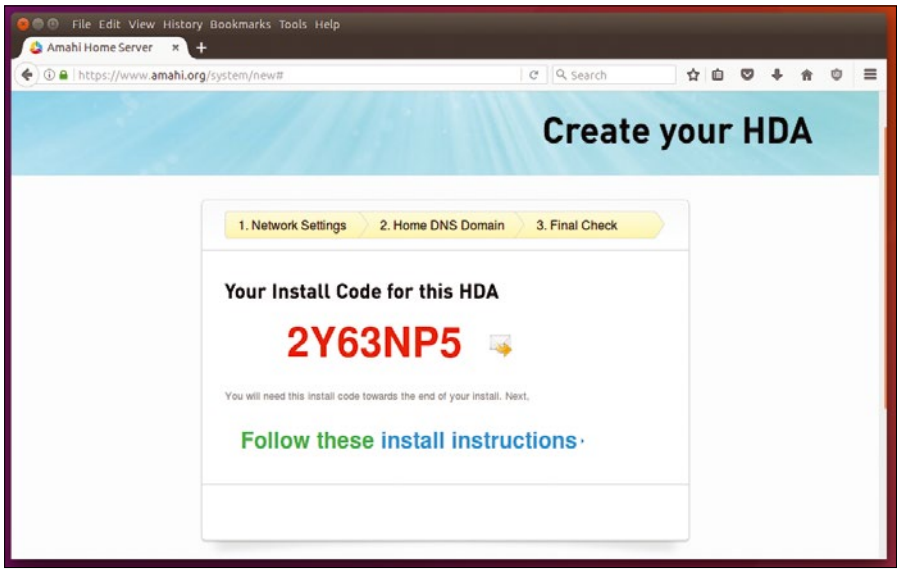


Figure 1: You can also ask Amahi to email you the install code along with install instructions.

protocol and even used as a network backup target. Amahi also comes with a free dynamic DNS name that you can use to access your files from anywhere on the planet.

Amahi has modest hardware requirements, which is why it's often touted as a solution for putting an old unused computer back into active duty. It can manage a small network from a computer with a 1GHz processor and 512MB RAM. Deployments on larger networks where multiple users are shuttling oodles of data running several different apps will require a multicore processor with at least 4GB of RAM and ideally multiple hard disks.

Get Amahi

At the time of writing, the latest stable version of the server is Amahi v9. The developers recommend rolling it out atop a Fedora 23 installation. Before you get started with Amahi, follow the instructions in the "Install Fedora Server 23" box to set up a dedicated machine for the Amahi Server using the Fedora netinstall ISO to install a minimal server.

Once your minimal Fedora 23 server is up and running, switch to any other machine with a functional desktop environment and head to Amahi's website and click the big green *Get Started Now* button to register with the service. The sign-up process is fairly straightforward and, besides the login credentials, also involves picking up a username that will help determine your Dynamic DNS URL.

Once you've registered, log in to the Amahi dashboard on the website and click the *ConfigureYour HDA* button. An HDA or Home Digital Assistant is Amahi's way of referring to your Amahi Linux home server. The configuration process will walk you through a couple of pages requesting various information about your network setup. You'll be asked to enter the gateway address of the network that will host the Amahi server. This is the IP address of the wireless or wired router in your home network that provides the network settings for the rest of the network. Depending on the make and model of your router, the IP address for the gateway is typically 192.168.1.1, 192.168.0.1, or 192.168.0.254. To figure out these settings under Linux, enter the route command in a terminal and make a note of the address listed under the Gateway column for the default route.

Next up, you'll have to enter the fixed IP address that will be used by the Amahi server. Usually it's safe to go with the default suggestion, unless you've already assigned the listed address to another server on your network. For this tutorial, let's assume this to be 192.168.0.14. The third and last setting you'll be prompted for is the local DNS domain name. This is the name for your home domain so you can change it to whatever catches your fancy. Do keep in mind that your network shares and Amahi apps will be accessible via this domain, so make it something meaningful.

Once you've entered the requested information, click the *Create Your HDA Profile* button, which will bring up a page with the necessary information required to setup your Amahi HDA. Make a note of the install code shown on this page (Figure 1).

Roll Out the Server

Now head to your Fedora server, fire up a terminal window, and switch to the super user root with `su -`. The first order of business is to download and install Amahi's repository with:

```
rpm -Uvh http://f23.amahi.org/noarch/
  hda-release-6.9.0-1.noarch.rpm
```

Once the repository has been installed, grab the server with:

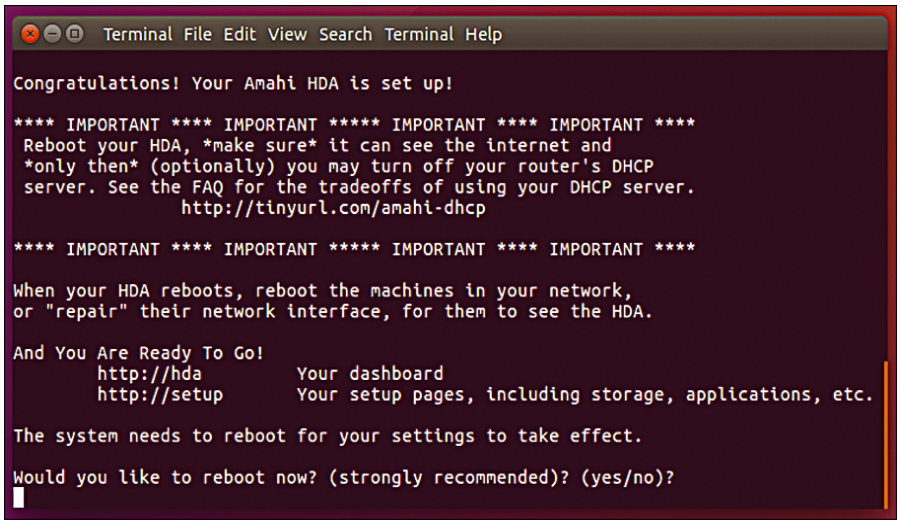


Figure 2: You can install Amahi on a headless server via SSH, which is installed as part of the minimal install selection.

NOT USING AMAHI'S CORE SERVICES

To take advantage of all of Amahi's network conveniences, you need to let it handle the DHCP and DNS services for the entire network. However, it is flexible enough to let you turn these off and instead continue using them from the existing source, which for a majority is the router. There are, however, disadvantages when you prevent Amahi from running these services.

If you continue using your existing DHCP server and only use Amahi for DNS as described in the tutorial, you'll still be able to use a majority of Amahi's network conveniences and functionality across all machines in your network. First, you'll be able to access the Amahi server using the `http://hda` address from all machines. Second, all computers in your network will be able to use the apps installed on the Amahi server using their network-friendly URLs, such as owncloud10.n10.net.

If you don't tweak the DNS settings in your router, you'll still be able to access the Amahi server by pointing to the server's complete IP address, such as 192.168.2.14, instead of the short address. Similarly, the other machines will also be able to access the file server when you point them to the server's URL. Some apps and services like UPnP might also be able to stream to other machines, but their performance isn't guaranteed.

```
dnf -y install hda-ctl
```

When these packages have been download, you can install Amahi using the install code shown earlier with:

```
hda-install <the-install-code>
```

This will configure the Amahi server as per the settings you provided earlier.

That's all there is to it. You don't have to manually edit any configuration files or tweak network settings. Amahi does it all for you automatically. When it's done, simply reboot the server (Figure 2). Once it comes back up, you'll have a fully functional home server that's initially accessible via the static IP address you set up previously (192.168.0.14, in our case). The first time you fire up your Amahi server's web interface, you'll be asked to create a dashboard admin user. It's a safe bet to supply the authentication information for the user you created while installing Fedora server.

By default, Amahi wants to manage your network and hand out IP addresses to all connected machines on your network and resolve websites. This allows all machines on your network to access the Amahi server, the apps running on the server as well as the shares with human readable names instead of IP addresses. However, most users already have a DHCP server on their router. You can, of course, continue using the router's DHCP server and just use Amahi for DNS, which still lets you access the server and the apps with friendly names. However, there are some trade-offs as mentioned in the "Not Using Amahi's Core Services" box.

To continue using your router's DHCP address, fire up the Amahi server's web interface and login. Now head to *Setup* | *Settings* | *Details* and toggle the *Advanced Settings* option. After the advanced settings have been enabled, head to *Network* | *Settings* and disable the DHCP server by unchecking the checkbox adjacent to the option labeled DHCP. The next important task is to ask your local network to use Amahi Server's DNS to access the Amahi apps. The exact process for this step varies from one router to the other. Generally, you open the router's admin page in your browser and look for the page that lists settings for the DHCP server on the router. Among other settings on this page, you can enter the static address of the Amahi server as both the Primary and Secondary DNS servers (Figure 3).

Bear in mind though that after you've tasked Amahi with resolving DNS you'll have to make sure that the server running Amahi is up and running before any client can access the Internet. If the Amahi server goes down, the computers on the network will not be able to resolve websites until the server running Amahi comes back up again. If you ever take down the Amahi server, don't forget to hand over the DNS function back to your router. Once you've set up Amahi's DNS, you can access your server by pointing your web browser to `http://hda` (Figure 4).

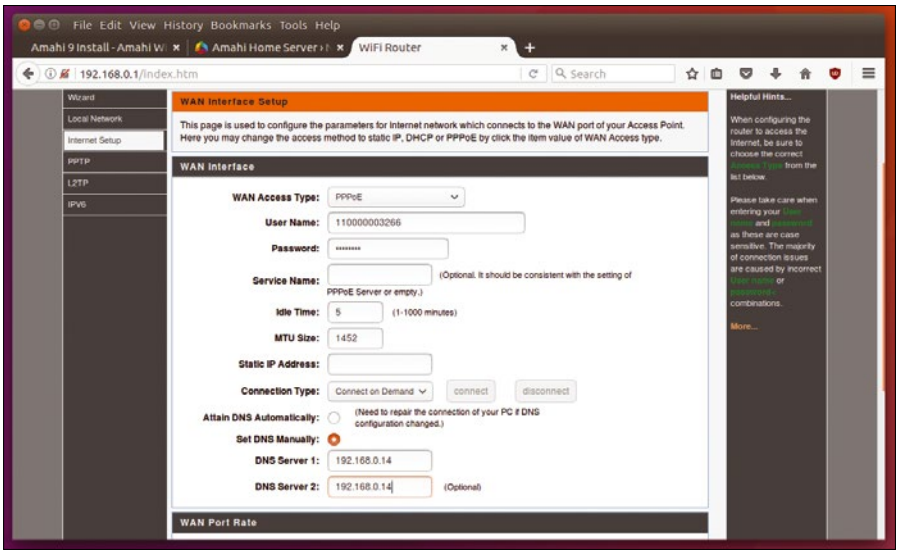


Figure 3: The DNS Server 2 is optional and can be left blank.

Share Files

You can now start configuring the home server as per your requirements. The first order of business is to create users who will interact with the server. At this point the server has only one user that's also the Administrator by default. You should create other users depending on the number of users that are connected to your network and will be using the data in the Amahi server either directly or via its many apps.

To add a user, head to *Setup* | *Users* and click the *New User* button at the bottom of the page. This opens up a brief form and prompts you for full name and authentication information for the user. Once a user has been added, it's displayed in the users table along with the other users. To edit a user's detail, click on their username in the table. You can then edit their details and even toggle the Admin checkbox to make them an administrator on the

Amahi server with the same privileges as that of the first user.

Once you've defined the users, it's time to manage network shares, which is one of Amahi's core functions. By default, Amahi creates a bunch of shared folders (books, movies, music, pictures, etc.) that are accessible to all users. To view and configure them, head to *Setup* | *Shares*.

You can further customize an individual share by clicking it. This brings several options related to that particular share. From here you can reset a share's permissions, control access, and even delete the share entirely. By default, all shares are available to all users. To specify users, uncheck the *All Users* checkbox. This displays a list of users on the server and lets you select which user has read and/or write access to the folder (Figure 5). To create a new share, scroll down and click the *New Share* button. Give it a name, and set it to visible. After it's been created, you can repeat the process described earlier to control access and permissions.

You can now access your shares from other computers in your network. To manipulate the files in the shares from Windows, launch the file manager and enter `\\hda` in the address bar. Similarly, in Linux the Shares are accessible when you enter `smb://hda` in a file manager.

Universal Access

Once you've set up the storage, you can enrich your home server by adding apps. Select the *Apps* option from the toolbar at the top of the dashboard to browse through the list of all supported apps. All apps follow a similar installation procedure. Click on the app to expand it and read about it in detail. Once you're sure you'd like to use it, click the *Install* button, which will download the app. When it's done downloading, Amahi will show you the necessary information you need to use the app including the credentials for the default admin user (Figure 6). One of the best things about these Amahi apps is that they are preconfigured for your network, so you can start using them without any delay.

If you want universal access to your files, you can use the Amahi Anywhere app to remotely browse and stream files

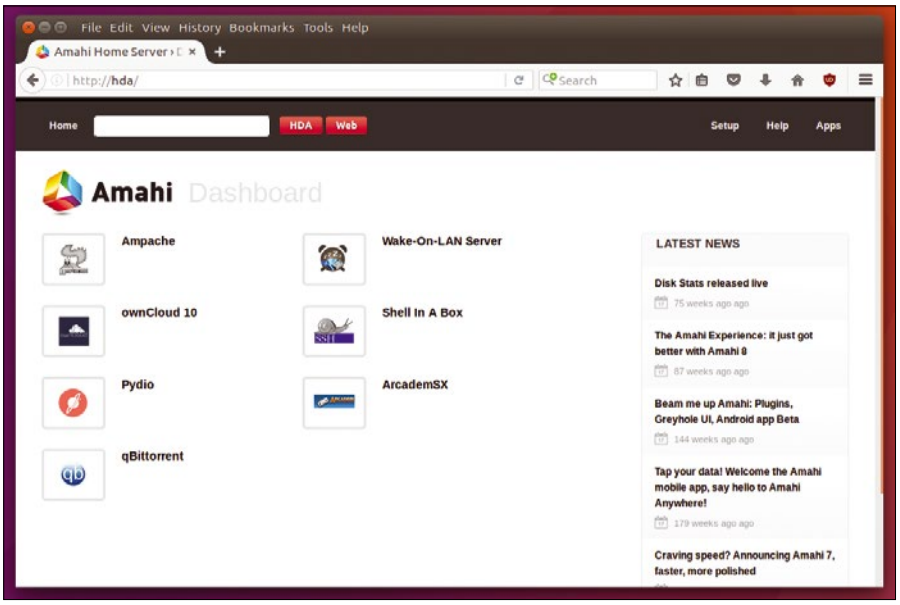


Figure 4: By default, Amahi lists all the installed apps on its simple dashboard, but you can also access them directly via their friendly URLs.

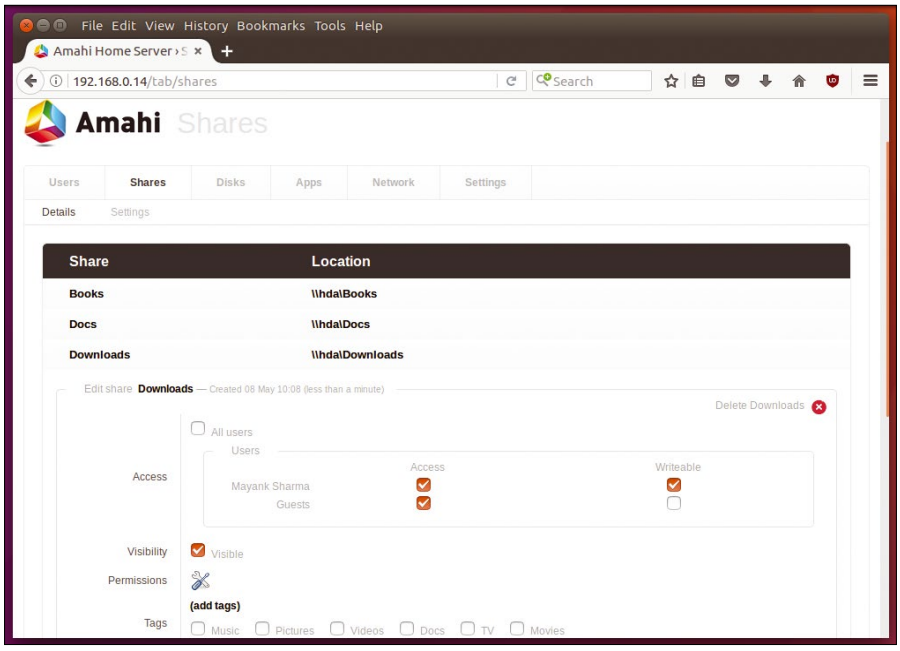


Figure 5: Amahi's access control skills are rudimentary but enough to be effective.

from your server on an Android or iOS device. First, install the Amahi Anywhere app on your Amahi server from under the list of apps displayed in *Setup* | *Apps*. Next, head to the Google Play Store or the Apple App Store on the mobile device and install the freely available Amahi app. Once installed, fire up the app and use the credentials for accessing the Amahi online dashboard to log in and browse the files on your Amahi HDA.

Use as NAS

If you have multiple hard drives in your server, you can use the Disk Wizard to make the Amahi server aware of them. Again, head to *Setup* | *Apps* and first install the Disk Wizard app. Shut down the server and plug in the additional drives if you haven't already. Then, power on the server and head to *Setup* | *Disks*. The page will list all the drives connected to your Amahi server under the *Devices* tab. Switch to the *Partitions* tab to view how the space inside the drives is used and divided into partitions.

Next, click on the *Add* button to append a newly connected drive to the pool of drives managed by Amahi. Select the

additional drive and click the *Next* button. Here toggle the button to format the drive and select a filesystem. It's best to go with the default option, which is ext4, unless you have a reason for favoring a particular filesystem. Besides ext4, it supports ext3, NTFS, and FAT32 filesystems. In the next screen, toggle the option to mount the drive automatically and give it a label for easier identification. Now review the settings before pressing the *Apply* button. Depending on the size of the disk and the processing power of the server, Amahi will take some time to format and add the new drive. When it's done, you can use the *Continue with another device* button to repeat the process and add more drives.

Network Backup Target

Because the computers on your network see the Amahi server as a NAS server, you can use it to save backups from all devices connected to the network. The Amahi NAS server supports various protocols including Samba, WebDAV, and SSH that you can use to backup your data. Although you can use any of these protocols for maximum security, we'll use SSH. It's one of the few protocols that's supported by a variety of backup tools including the ones that are used by default in several distributions such as Deja Dup in Ubuntu. Other popular backup tools that support the use of SSH as a transport protocol include BackupPC, Back In Time, Duplicati, luckyBackup, and others [3].

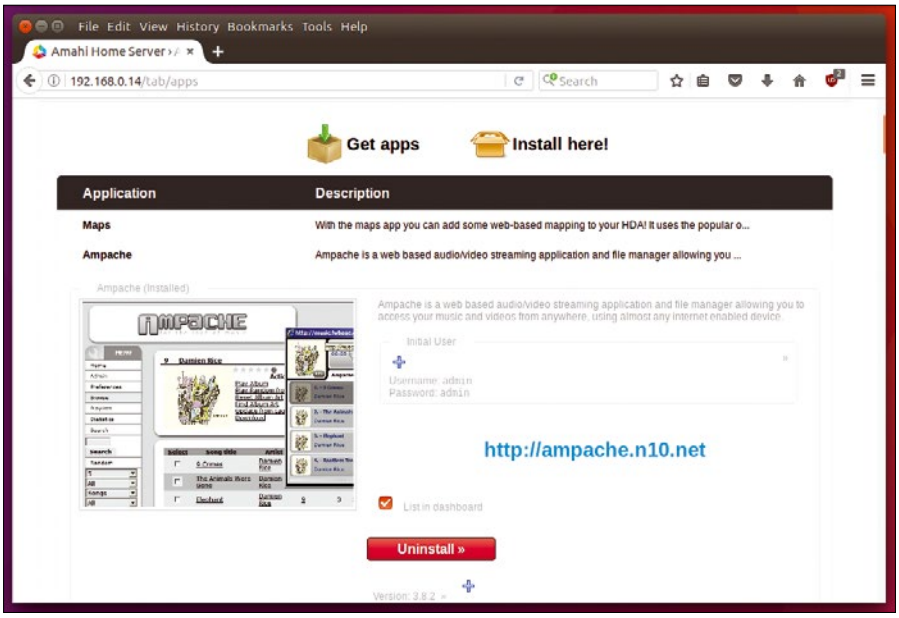


Figure 6: Remember to change the password of the default admin user for all apps.

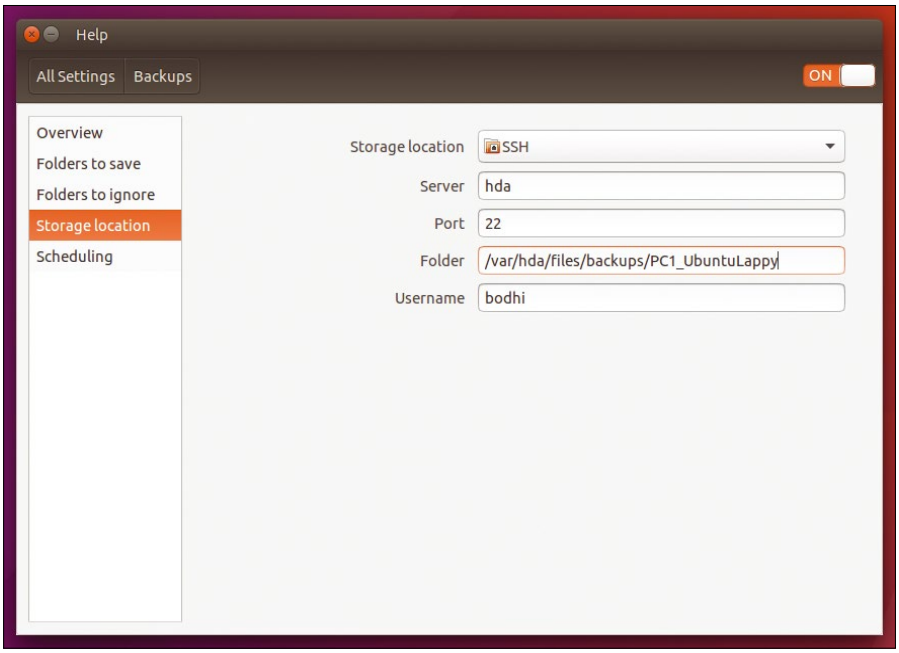


Figure 7: SSH isn't just secure but is often recommended as the best protocol for incremental backups.

Although the process for configuring a backup tool to transfer the backups over SSH varies from one tool to another, they all usually ask for the same information. When defining a backup target, you'll have to select SSH from the list of supported protocols. Next, you'll be asked for the username of a user on the SSH server. Although this can be any user on the Amahi server, you must make sure they have write access to the Amahi Share in which you want to house the backup. By default, the Amahi administrator, which is the user we defined while installing Fedora Server, has read-write permissions over all the Shares.

Once you've specified the username, you'll be asked for the complete path to the folder that will house the backups. The Shares on the Amahi server are defined under the `/var/hda/files` folder. A good strategy is to keep backups from different computers inside a separate subfolder inside a backup meta folder, such as `/var/hda/files/backups/PC1_UbuntuLappy`, `/var/hda/files/backups/PC2_FedoraVM`, and so on (Figure 7).

That's usually all there is to it. Some tools will also let you save the password for the username, whereas others will prompt for it when they establish an SSH connected to the Amahi server. Some tools like Deja Dup also let you keep the backups under lock-and-key and will ask for an additional password. If you choose to head down this route, remember that you'll need this password to restore your files from the backup as well.

Sync Files

Another popular use of the Amahi server is to sync data from a mobile device. For example, if you take a lot of photographs from your mobile phone, you can configure it to automatically upload the photographs from your phone to an Amahi Share. The Amahi Sync app from the developers of the server is designed for this very purpose. However the app, unlike the server, is a paid app and is available

only to Pro ($7.95/month) and Ninja ($15.95) users [4].

The premium accounts offer other benefits as well, but if you're just interested in the sync facility, you can get it for free using the ownCloud 10 and Pydio apps. Both of these apps are available for free in the Amahi app store and come with the ability to sync data. To begin the process of syncing files, log into the Amahi server and click the Apps link in the top bar. Browse the list and install either the Pydio or ownCloud app [5]. Once the app is installed, make a note of their respective default login credentials. Now log into the app and create a folder that will house the synced images. Let's use ownCloud 10, which by default displays a handful of folders. Click on the + icon at the top to create a new folder where you want to keep the uploaded images.

Next head to the Play Store in your Android device and install the Synchronize Ultimate app [6]. The freely available ad-supported version of the app lets you sync data between your Android device and several data hosting servers including Pydio and ownCloud. Fire up the app and tap the *Remote accounts* option to connect the app with the ownCloud installation on the Amahi server. Tap on the + icon to bring up a list of supported services. Next, you can browse the list and tap the ownCloud option. The app will then ask you for the ownCloud host and authentication information. The host is the short-name for the ownCloud service running atop Amahi. After entering the information, tap the floppy icon at the top to save the details and return to the main menu.

Now tap on *Profiles* to define the sync task. Again, tap on the + icon to set up the profile. First, is the *General* tab, which prompts you for the name of the profile. Next is the *Sync Type*, which defines the direction of the synchronization. The default left-to-right option will sync data from the source defined in the left tab to the destination defined under the right tab. There's also a right-to-left sync option and a two-ways bidirectional sync that will make sure the content under the source and destination folders are mirrors of each other.

For our task, we'll first define the left-to-right profile with the left tab being our Android phone and right tab being the ownCloud folder on Amahi. After defining the general parameters, switch to the *Left Side* tab and use the Account pull-down menu to select the *Internal* option. Tap the *Browse* button to navigate to the folder that stores the images you've clicked, then switch to the *Right Side* tab, and again bring up the Account pull-down list. This time, tap the *ownCloud* option.

When you now tap the *Browse* button, the app will establish a connection to the ownCloud instance on the Amahi server and fetch a list of folders on the remote server. Navigate the filesystem and tap on the folder you created earlier to house the images. Tap on the floppy icon to save the changes and return to the page that lists the sync profiles. You'll find a Play icon adjacent to the profile you've just created. When you tap the icon, the app will connect to the remote ownCloud installation and save all the images from your phone to the Amahi server (Figure 8).

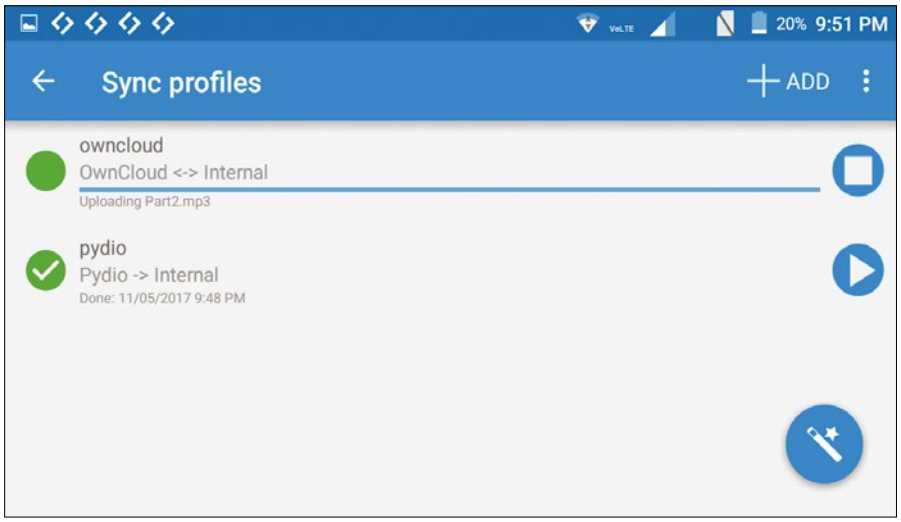


Figure 8: The ad-supported version of Synchronize Ultimate lets you create a maximum of two sync profiles.

Explore the app and create other types of tasks to keep a folder synchronized with the data on the Amahi server. We've only looked at a fraction of the things you can do with your Amahi home server. You can explore and flesh out the server with several other apps (see the "Other Useful Apps" box) to put the server to creative use. ■■■

OTHER USEFUL APPS

Besides the ones already mentioned, several other apps can enhance the productivity of your Amahi home server. Disk Stats visualizes the disks connected to Amahi. If you download Linux distributions via torrents, install the qBittorrent app to manage your downloads using its excellent web interface. For a bit of nostalgia, you can install the ArcademSX Game Pack 1 & 2, which are collections of classic arcade games, such as Pacman, Mario Bros 2, Sonic the Hedgehog, Treasure Chain and more.

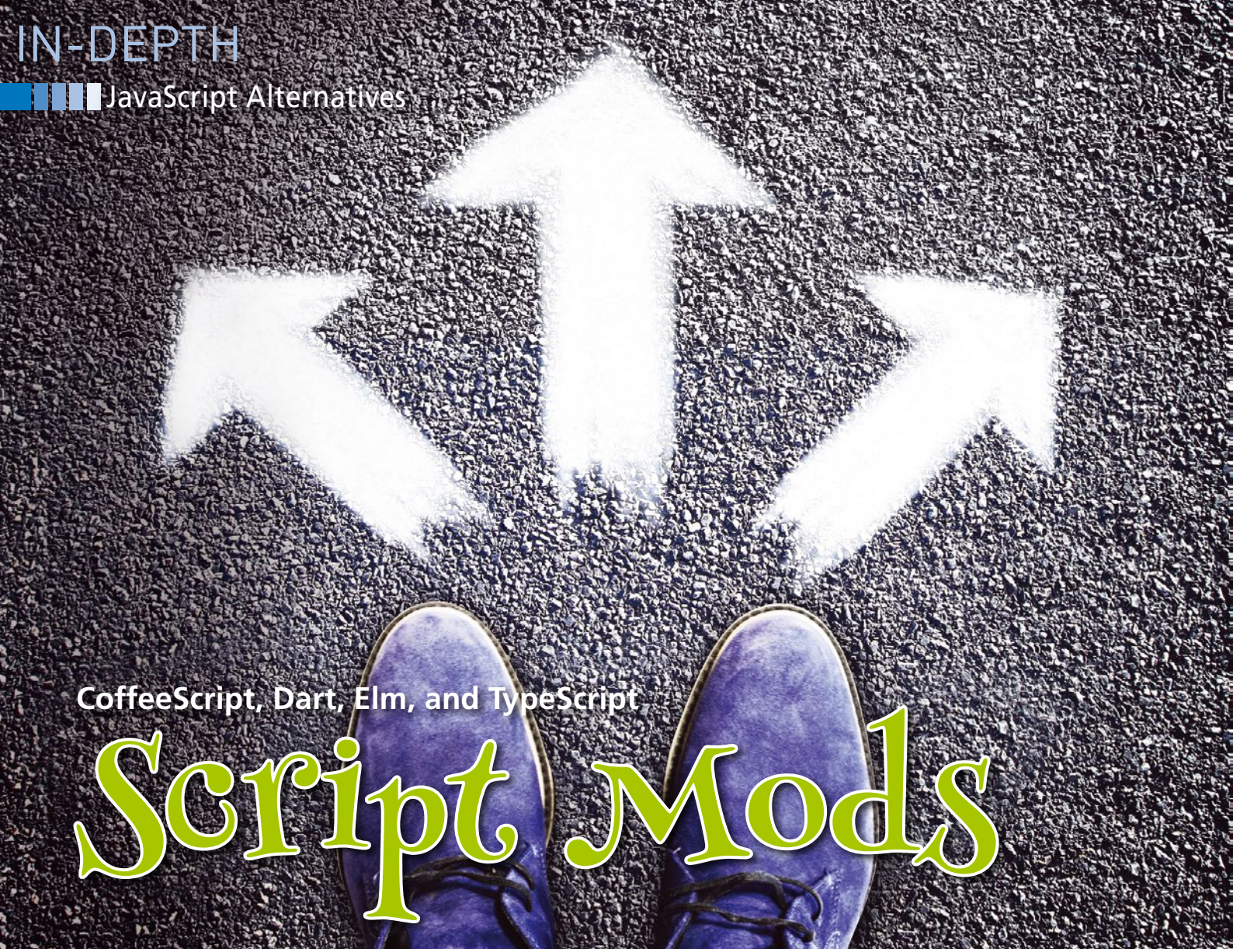
Advanced users should install the HDA Troubleshooting Tools app, which adds the `ifconfig`, `nslookup`, and `netstat` command-line utilities to the Amahi server to help troubleshoot networking issues. If you are running Amahi on a headless server, install the Shell-In-A-Box app, which is a browser-based terminal emulator and saves you the effort of firing up a terminal and initiating a SSH connection for remote administration. Finally, if you are using Amahi to dole out DHCP addresses to the computers in your network, you can use the Wake-On-LAN server app to remotely boot and power down any connected computer on your network.

INFO

[1] Amahi: *www.amahi.org*
[2] Download Fedora 23 ISO: *https://tinyurl.com/f23iso*
[3] Backup tools: *https://tinyurl.com/linux-backup-tools*
[4] Amahi Premium plans: *https://www.amahi.org/plans*
[5] ownCloud for Amahi: *https://tinyurl.com/amahi-owncloud*
[6] Synchronize Ultimate: *https://tinyurl.com/sync-ultimate*

AUTHOR

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



CoffeeScript, Dart, Elm, and TypeScript

Script Mods

JavaScript is the stuff of which many interactive web clients is made, but it comes with a fair amount of historical ballast. The creators of four alternative scripting languages seek to ditch the ballast. *By Tim Schürmann*

Browser-based interactive applications have helped the recent comeback of JavaScript, although the scripting language first saw the light of day in the mid-1990s. Over the course of time, JavaScript's makers added more and more new constructs to what was initially a fairly simple scripting language. One prime example is prototype-based object orientation, hated by many developers and often misunderstood and unused.

Not until 2015 was the scripting language, by then standardized as ECMAScript, given optional class-based object orientation. However, compared with Java, C++, and others, it still lacks certain features [1]; moreover, neither JavaScript nor ECMAScript offers type checking, which continually results in exception errors in practice.

For this reason, several scripting languages want to replace JavaScript – or at least simplify it in terms of programming. Some of the most widespread and popular examples are CoffeeScript, Google's Dart, Elm, and Microsoft's TypeScript. In a comparison, I describe what distinguishes the four candidates and in which areas they are superior to JavaScript.

CoffeeScript

Jeremy Ashkenas published the first version of CoffeeScript [2] in 2009. His scripting language does not simply reinvent the wheel; rather, it supplements JavaScript with additional constructs so that developers can continue to use existing JavaScript libraries such as jQuery [3]. When this issue went to press, CoffeeScript had reached version 1.12.4, which only supported ECMAScript (ES) 5 and some parts of ES2015. The upcoming CoffeeScript 2.0.0 will implement more elements from ES2015 – in particular, the classes it introduces.

The compiler is available under the MIT license and requires a JavaScript run-time environment (e.g., Node.js). CoffeeScript plugins exist for many text editors, such as Emacs and Gedit. If you want to try CoffeeScript first, you can do so from an online editor [4] on the project site (Figure 1).

A compiler provided by the CoffeeScript team translates CoffeeScript code into the JavaScript equivalent, which in turn runs in any browser. The JavaScript code is said to run faster than an equivalent written by hand.

CoffeeScript also lets you use shorthand notation. Among other things, you can leave out the `var` before variable

declarations and the semicolon at the end of an expression. Comments begin with a hashtag (#), as in shell scripts, whereas two slashes (//) cause integer division. Longer strings and regular expressions can run across multiple lines. CoffeeScript also lets you chop up arrays:

```
start = countdown[0..2]
```

In this case, start only contains the first three items of countdown[]. You can create functions with the -> operator and set the parameters to default values:

```
mult = (x = 1) -> x * x
```

Rather than curly braces, indentations mark statement sections, as in Python. Listing 1 defines a Dot object in this way with two properties and one method, draw(). The splat operator (...) lets you pass any number of parameters to a function. The ? checks whether a variable exists.

The pattern <result> = <value> if <condition> exists for simple conditions. Because the JavaScript comparison operator (==) often leads to misinterpretations, the CoffeeScript compiler automatically translates it to ===. Additionally, many Boolean and comparison operators have synonyms; for example, on and off are identical to true and false. Chained comparisons quickly test whether a variable lies in a particular range:

```
inpicture = 0 > Dot.x > 640
```

The scripting language also offers a Ruby-style switch-case construct.

Besides this, CoffeeScript extends for loops to include comprehensions, which help the code iterate through the elements of an array in a particularly elegant way. At the same time, they are expressions and can be assigned and returned:

```
drive = (a) -> alert(a)
drive auto for auto ➲
  in ['bmw', 'vw',  'skoda'] ➲
  when auto is not 'skoda'
```

The compiler also attempts to transform statements into expressions. Functions always return their last value, which means you can often omit return. A while loop also contains an array with the result of one iteration of a loop. In the example, countdown contains an array with the numbers 9 to 0:

```
counter = 10
countdown = while counter > 0
   counter = counter - 1
```

CoffeeScript 1.12.4 introduced a simple class concept (Listing 2). Here, @ is the short form of this; super (line 8) invokes the function currently running in the parent class. The upcoming CoffeeScript 2.0.0 compiles the classes to create their counterparts from ES2015.

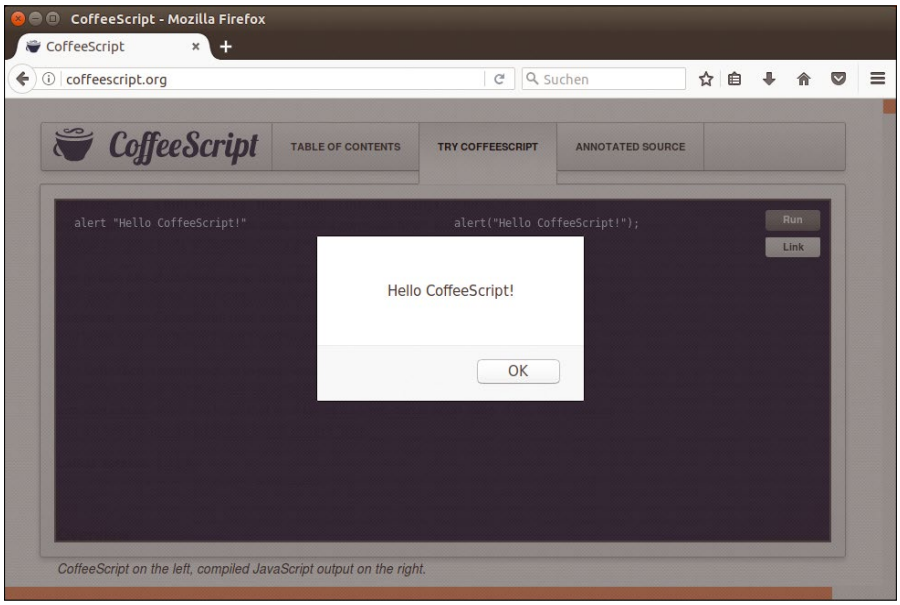


Figure 1: CoffeeScript website developers can type source code on the left in the *Try CoffeeScript* tab and then press *Run* to execute. The compiled JavaScript code appears on the right.

LISTING 1: Simple Object in CoffeeScript

```
01 Dot =
02    x: 1
03    y: 2
04    color: "red"
05    draw: () -> alert(this.x + "," + this.y + "," + this.color);
06    paint: (a, b...) ->
07       exists = true if this.color?
08       if exists == true
09          this.color=a
10       else alert("No color");
11
12 Dot.paint("green");
13 Dot.draw();
```

CoffeeScript 1.12.4 already supported generator functions and tagged template literals from ES2015. Template literals incorporate the contents of variables into text:

```
greeting = "Hello #{name}"
```

LISTING 2: CoffeeScript Class Definition

```
01 class Dot
02    constructor: (@x,  @y) ->
03
04    draw: () -> alert(@x + "," + @y);
05
06 class Rectangle extends Dot
07    constructor: (x, y, @b,  @h) ->
08       super x,y
09    draw: () -> alert(@x + ";" + @y + ";" + @b + ";" + @h);
10
11
12 r = new Rectangle 1,2,3,4
13 r.draw();
```

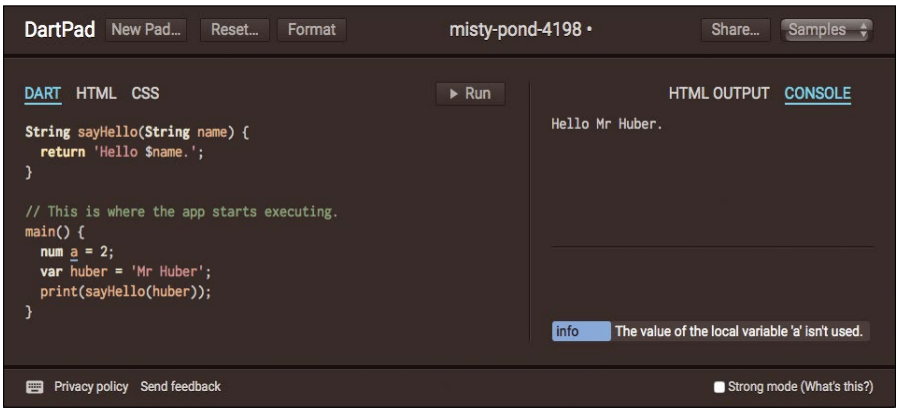


Figure 2: You can easily try out Dart scripts in DartPad. At bottom right, you can see the compiler notes.

In addition to the compiler, the CoffeeScript package includes a simple build system that works much like Make or Rake. Its core is a `Cakefile`, in which you define the various tasks for the compiler. These can then be executed using the `cake` tool. You then feed a markdown document with the `.litcoffee` extension to the CoffeeScript compiler. The compiler interprets all the indented blocks in the document as CoffeeScript code and ignores the rest.

Dart

Google first presented its Dart [5] scripting language to the general public in 2011. Either a Dart virtual machine (VM) executes programs written in Dart, or a compiler converts them into JavaScript code. The Dart SDK provided by Google includes the Dart VM and compiler; it is available under a BSD-style license and includes other useful tools. For example, `dartanalyzer` analyzes the Dart scripts fed to it and points out potential pitfalls. Additionally, a Chromium browser comes with an integrated Dart VM [6] dubbed Dartium.

Plugins help IDEs and editors such as Atom [7] and Emacs [8] learn the scripting language; WebStorm [9] even supports Dart out of the box. If you do not want to install the SDK, you can get started with the DartPad [10] online editor shown in Figure 2. According to Google, Dart is especially suited for programming larger applications. ECMA has now standardized Dart [11].

Each Dart program must have a `main()` function as its entry point. Figure 2 on the left shows an example of a simple script that defines a `sayHello()` function, which `main()` calls. If you indicate the variable type, as in this example, a type test is performed later on. In addition to floating-point numbers (`double`) are integers (`int`). Both are subtypes of `num`, which supports the basic addition (`+`), subtraction (`-`), multiplication (`*`), and division (`/`) operations.

Dart also supports strings (`String`) and Boolean values (`bool`). A long string can be spread across multiple lines. Strings use UTF-16 encoding by default. If you want a variable to store arbitrary values, you declared it with `var`, as in JavaScript. Dart uses the `$` notation to add variable content to a string, and `${exp}` lets you integrate an entire expression (`exp`) into the text.

For short functions like `sayHello()`, you can use shorthand notation:

```
String sayHello(String name) => ➲
  'Hello  $name.';
```

The `enum` type lets you create a custom data type. In the following example, variables of type `Color` assume one of three values: `Color.Red`, `Color.Yellow`, or `Color.Blue`:

```
enum Color {Red, Yellow, Blue}
```

Dart only supports the `==` comparison operator, with shorthand notations for conditions:

```
var result = condition? Expr1 : expr2
```

or, even shorter,

```
var result = expr1 ?? expr2
```

In the latter case, if `expr1` is not `null`, Dart returns its value; otherwise, it returns the result of `expr2`.

If Dart expects a Boolean value, only `true` is evaluated as true. Unlike JavaScript code, the following thus returns nothing:

```
var smith = 'Mr Smith';
If (smith) print('Hi!');
```

In addition to arrays – referred to as lists – Dart has maps, which store key-value pairs. In addition to the `for` loops known from JavaScript, programmers can use `foreach()` to iterate over the elements of an object and `for … in` to iterate through lists and other iterable objects. To leave a `while` loop, you use `break`; `continue` immediately starts the next iteration of the loop. Both keywords also occur in `switch-case` constructs.

In Dart, everything is an object. Functions store variables, which means that variables can also be used as arguments for other functions. As in CoffeeScript, you can denote parameters as optional and define default values. Dart supports anonymous (lambda) functions and closures. Types can be tested at run time with the keywords `as`, `is`, and `is!` (i.e., *is not*); for example:

```
(p as Person).name = 'Henry';
```

Listing 3 shows an example of a class definition: The constructor in Dart has the same name as the class. The example also uses shorthand notation that directly assigns the values passed to `x` and `y`. Alternatively, the variables can be listed, wherein the assignment occurs before the constructor body is executed:

```
Dot(num a, num b) : x=a, y=b { ... }
```

If needed, you can name constructors and thus highlight their purpose, as shown in Listing 3 in `Dot.onXAxis()`. The attached `: this(x,0)` (line 6) calls the nameless constructor. Unlike CoffeeScript, Elm, and TypeScript, you can overload some operators (e.g., `+`). In the listing, this ability is used to add two more dots later.

Inheritance is implemented with `extends`, and `super` accesses the parent class. All methods and properties are public. If their names start with an underscore (e.g., `_coverage` in this example), they are automatically only visible within the class (`private`).

The `Square` class in Listing 3 demonstrates an abbreviated notation for `get` and `set` methods. Additionally, you can create constant objects that are immutable in the course of the program. Properties and

methods marked as `static` are shared by all objects in a class.

Factory constructors are used to ensure that only a single or a specific object of the class exists; mixins let you combine multiple classes in a single class:

```
class Car extends Engine with
  Body { //... }
```

In Listing 3, if you replaced `extends` with `implements`, `Square` would inherit the interface from `Dot`, but not the implementations. Dart can derive classes from those classes tagged as `abstract` but cannot create any objects.

If necessary, you can initiate access to non-existing methods in Dart. To do so, you only need to implement the appropriate class method:

```
noSuchMethod(Invocation mirror)
```

If the class provides a method named `call()`, both its object and its function can be called.

Derived classes can override the methods of the parent class with their own versions. To do this, you need to tag them with `@override`:

```
class Rectangle extends Dot {
        @override
        void draw() { ... }
}
```

Besides `@override`, Dart currently supports other annotations – known as metadata. Dart developers should no longer use methods tagged as `@deprecated`. Finally, objects can be produced from JSON data:

```
var dot = JSON.decode('{"x":1, "y":2}');
```

Dart also supports generics. When declaring a class or function, it is not clear which values one of its objects will process later; thus, you need to use a placeholder (e.g., `T` in the example that follows). You only define the type when creating the object; `extends` (in the angle brackets) lets you assign types to specific (sub)classes:

```
class Car<T extends Daimler> {
        T drive(T model) { ... };
}
var mb = new Car<Mercedes>();
```

LISTING 3: Class Definition Example in Dart

```
01 class Dot {
02         num x;
03         num y = 12;
04         num _coverage = 1;
05         Dot(this.x, this.y);
06         Dot.onXAxis(num x) : this(x, 0);
07         Dot operator +(Dot p) {
08                 return new Dot(x + p.x, y + p.y);
09         }
10 }
11
12 class Square extends Dot {
13         num width;
14         Square(x, y, b) : super(x,y) {this.width=b;}
15         num get right => this.x + this.width;
16         set right(num w)  => this.width = w - this.x;
17 }
18
19 main() {
20         var a = new Dot(1,2);
21         var b = new Dot(3,4);
22         var c = a + b;
23
24         var d = new Square(1,2,3);
25         d.right=20;
26 }
```

Dart is oriented on Java when it comes to handling exceptions; for example, you use `throw` to create an object and `catch` to field it. Dart provides prebuilt `error` and `exception` classes for error handling, from which you can derive your own classes.

Dart supports asynchronous programming: A function tagged `async` returns immediately before Dart has run the statements contained in the function's body. Similarly, a function call tagged `await` waits until the asynchronous function has finished its work. The future API will offer a concept similar to `Promise` objects in ES2015.

Dart code can be outsourced to libraries, if needed, with each library using its own namespace. All functions and variables that start with an underscore in a library are visible and can be used only within the corresponding library. The `import` statement integrates libraries, which can reside on other servers, into a Dart script.

Dart even supports lazy loading, in which the Dart program does not load the library until it actually needs it. Existing JavaScript libraries can only be addressed via an API.

Elm

Evan Czaplicki created Elm [12] in 2012; the Elm Software Foundation is now responsible for development. In contrast to the other alternatives, Elm is a functional programming language. Although a compiler translates the code written in Elm to JavaScript code, existing JavaScript libraries can only be used with a special interface.

The compiler is available under a BSD-style license and is written in Haskell. It not only points out errors, it also delivers solutions (friendly error messages). As an alternative to the compiler, you can use a special console (a REPL, or Read-Eval-Print Loop shell), that directly executes the Elm code you type. Plugins are available for several popular editors, such as Atom, IntelliJ IDEA [13], and Emacs. If you are interested, you can try out Elm in an online editor [14].

Elm itself only supports a few constructs and keywords. When you define a variable, you can also specify the type. Elm distinguishes between Booleans, integers, floating-point numbers, characters, and strings:

```
counter: int
counter = 42
```

The compiler automatically infers the type of the variable and warns of problems. The `++` operator lets Elm developers weld strings together. The language distinguishes between integer division (`/`) and floating point division (`//`). Comments start with `--`. Lists are similar to JavaScript arrays:

```
colors = ["Red", "Blue", "Yellow"]
```

Lists can be manipulated with functions from a supplied library. For example, `List.sort colors` sorts the `colors` list. Tuples, such as `("Henry", 32)` contain a fixed number of random values. Records take several variables; the dot operator points to the value of a variable:

```
dot = { x = 1, y = 2 }
dot.x
```

Alternatively, you can use `.x dot` to tell Elm to retrieve the variable `x` from the `dot` record. To modify the values in a record, use `{dot | x = 3}`. All record manipulations are non-destructive: Elm does not change the value of `x` in the `dot` record but generates a completely new record. The compiler ensures efficient use of memory. In functions, you use the record elements directly:

```
dist {x,y} = sqrt (x^2 + y^2)
```

Finally, union types correspond to `enum` in Dart. In the following example, the new `Color` data type is created, from which variables can assume the value `red`, `yellow`, or `blue`.

```
type Color = Red | Yellow | Blue
```

Functions can be defined without the usual brackets, which are also missing in calls. You only need to use spaces to separate several parameters:

```
square n = n * n
```

If you want to avoid calling nested functions, you can use the pipe operator (`|`) to write them one after another. Elm supports anonymous, or lambda, functions, such as: `\n -> n/2`. The `\` is followed by the name of the parameter, and the arrow is followed by the function part. Conditions are also short and sweet:

```
isPositive number = if number > 0 then ➘
  "Positive" else "Not positive"
```

Alternatively, you can use `case … of` (as shown in Figure 3), and you can limit the visibility of variables to a specific expression with `let … in`. This essentially describes the language scope. The built-in functions let you design a complete web application and support access to HTML code. They consistently use a model-view-controller concept, which is referred to as the Elm architecture (Figure 3).

TypeScript

Microsoft has also worked for several years on a JavaScript alternative named TypeScript [15]. JavaScript libraries like jQuery can still be used in TypeScript, which also adds proprietary constructs to the latest version of ECMAScript, much like CoffeeScript. Today, elements from TypeScript are regularly fed back into the official ECMAScript standard. Microsoft offers a compiler for TypeScript [16] that translates TypeScript programs into ECMAScript. Like the alternative scripting languages, you can test TypeScript programs in an online editor – the Playground editor (Figure 4) [17] in this case.

As in Dart, you need to set the variable type; the compiler then performs a static type check. In the following example, the second line defines a function; the return type follows the name:

```
var name: string = "Tim";
function output: ➘
  void (a: string) { console.log(a); }
```

The `void()` return type tells the compiler that the function returns nothing. The special return type `never` defines functions that do not terminate, as is the case with exceptions. In addition to strings, TypeScript supports Boolean values (`boolean`) and floating-point numbers (`number`).

If the type of a variable is unknown, you explicitly assign the `any` type, which tells the compiler to switch off type checking. Conversely, you can check the type explicitly two ways:

```
var length: ➘
  any = (<string>firstname).length;
var length: ➘
  any = (firstname as string).length;
```

In functions, you can allow several alternative types for a single parameter. The following example passes in either a number or a string:

```
function foo(value: string | number) ➘
  { ... }
```

You can also dictate the exact values that the string is allowed to contain (string literal types), which is useful to enforce certain settings in web applications. Like Dart, TypeScript supports the enumerated type `enum`:

```
enum Color {Red, Yellow, Blue};
```

Tuples are an alternative. In the following example, `workers` stores two values: `John` and `34`. This pair of values can be accessed with the use of an index, as with an array:

```
var workers: var workers:
workers = ["John", 34].
workers[1] = 35;
```

Classes are defined as in ES2015. On request, the TypeScript compiler translates them to older ECMAScript. If a property

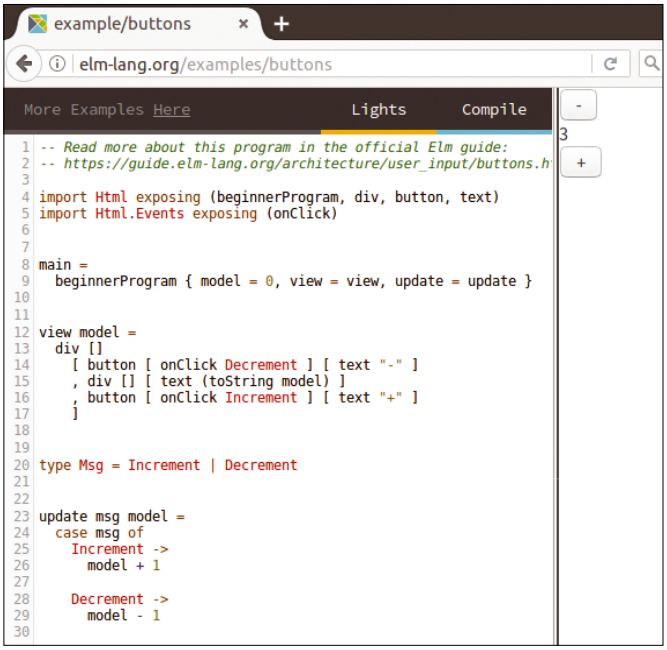


Figure 3: Elm provides several ready-made examples in the online editor. For example, the buttons example selected here creates the two buttons on the right side, with which users can change the number displayed.

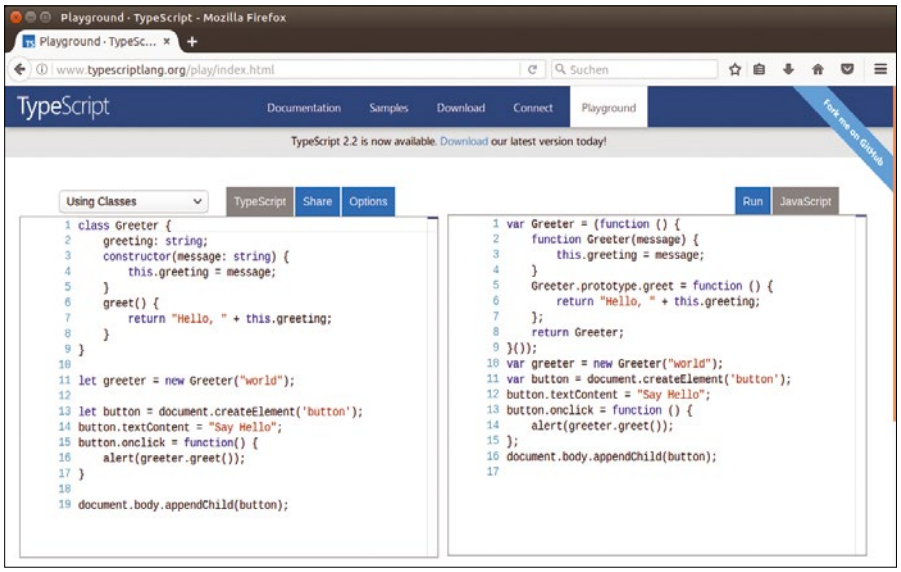


Figure 4: TypeScript examples are available in the Playground editor. The *Using Classes* example demonstrates the handling of classes.

or a method is preceded by the `private` keyword, only the methods from the class are allowed to access it. In the case of `protected`, however, all derived classes also have access.

The `static` keyword tells TypeScript to create properties that all objects of a class share. TypeScript adds notation for getter and setter methods:

```
class Auto {
   protected _model: string;
   get model(): string {
     return this. _model; }}
```

TypeScript adds interfaces to the class concept that describe only the structure or the interface of an object (e.g., Listing 4). A property that is marked with a question mark is optional. The `extends` keyword lets an interface extend another interface or class; for classes, the interface only adopts the methods and properties.

In TypeScript, two types are compatible with each other if their internal structure is the same. Unlike other languages, then, you do not need to implement the interface explicitly. It is sufficient if an object provides the interface described. Listing 4 shows an example in which `sayHello()` correctly processes an `Employee`. An interface describes functions as well as classes. In addition to interfaces are abstract classes, which only let you derive other classes.

In the case of objects created with `const`, you can also change the properties of the object. This is only prevented by the `readonly` keyword preceding the corresponding property. Indexable types transform an interface into an array or a dictionary.

LISTING 4: Example of Interfaces in TypeScript

```
01 interface Person {
02         firstname: string;
03         familyname?: string;
04 }
05
06 interface Student extends Person {
07         matriculationno: number;
08 }
09
10 function sayHello(person : Person) {
11         console.log("Hello " + person.firstname);
12 }
13
14 class Employee {
15         name: string;
16         constructor(public firstname, public surname) {
17                 this.name = firstname + " " + surnam;
18         }
19 }
20
21 var john = new Employee ("John", "Miller");
22 sayHello(john);
```

In the following example, the `Employees` interface comprises several names:

```
interface Employees {}
        [index: number]: string;
}
```

Access is as in an array. Here, the index is a number, and the stored values are text (`string`). You replace `number` with `string` to access the stored values by way of terms (e.g., `zip["Lawrence"]`).

In some cases, TypeScript groups several declarations to form a single declaration (declaration merging). In the following example, the compiler automatically merges the two interface declarations:

```
interface Dot { x: number; }
interface Dot { y: number; }
let p: Dot = {x: 1, y: 2};
```

TypeScript also offers generics, with notation similar to Java or Dart. Mixins compile several classes to create a new one, and intersection types allow combining of types. For example, `Person & Employee & Student` is a person, and an employee, and a student.

TypeScript supports ECMAScript modules and produces code that is suitable for modular solutions from Node.js (CommonJS), RequireJS (AMD), Isomorphic (UMD), and SystemJS. TypeScript encapsulates variables, classes, and other elements in its own namespace on request, reducing the risk that an imported function uses the same name as your own function.

If you enjoy experiments, you can already use decorators in the current version. They give a class, property, or method additional properties. The example assigns the decorator `@sealed` to the class `foo`:

```
@sealed
class Foo { ... }
```

What this modification actually does is determined in a private function. The TypeScript compiler supports the JSX specification [18] and points out typical JavaScript errors. Plugins integrate TypeScript with multiple build tools such as Maven. Plugins exist for Atom, Eclipse, and many other editors and IDEs. The compiler itself is written in TypeScript and needs Node.js to execute.

The tools developed by Microsoft and the compiler itself are available under the Apache 2.0 license; the language specification is under the Open Web Foundation Final Specification Agreement Version 1.0 (OWF 1.0) [19] open license.

Conclusions

The choice between CoffeeScript, Dart, Elm, and TypeScript depends on the project in question, your requirements, and your own preferences. CoffeeScript is slightly out of touch with ECMAScript and does not offer type checking. The advantage of this scripting language is thus limited to Python-style code formatting with tabs and the ability to mix Markdown with CoffeeScript.

Dart can be run in a special VM and offers modern object-oriented concepts, such as mixins, but it is oriented more on Java and C than on JavaScript.

Elm specifically targets fans of functional programming who can develop web applications using familiar patterns of thought. Thanks to automatic type deduction, run-time errors are also rare in Elm.

TypeScript is now used as a playground for future versions of ECMAScript. Because it is based on JavaScript, making the move is easy, and typing ensures fewer run-time errors. Dart and TypeScript let you mix typed and typeless variables, so despite compiler support, a small error source remains. ■■■

INFO

[1] ECMAScript specification: *https://github.com/tc39/ecma262/*
[2] CoffeeScript: *http://coffeescript.org*
[3] jQuery: *https://jquery.com*
[4] CoffeeScript online editor: *http://coffeescript.org*
[5] Dart: *https://www.dartlang.org*
[6] Dartium: *https://webdev.dartlang.org/tools/dartium*
[7] Atom editor: *https://atom.io*
[8] Emacs: *https://www.gnu.org/software/emacs/*
[9] WebStorm: *https://www.jetbrains.com/webstorm/*
[10] DartPad: *https://dartpad.dartlang.org*
[11] Dart ECMA specification: *http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-408.pdf*
[12] Elm: *http://elm-lang.org*
[13] IntelliJ IDEA: *https://www.jetbrains.com/idea/*
[14] Elm editor: *http://elm-lang.org/try*
[15] TypeScript: *http://typescriptlang.org*
[16] TypeScript tools on GitHub: *https://github.com/Microsoft/Typescript*
[17] TypeScript Playground: *http://www.typescriptlang.org/play/index.html*
[18] JSX: *http://www.typescriptlang.org/docs/handbook/jsx.html*
[19] OWF 1.0: *https://github.com/Microsoft/Typescript/blob/master/doc/spec.md*

The sys admin's daily grind: Enlightened libcoap

Ikea Illuminati

Charly did a spot of shopping in a furniture store and came out with a smart lighting system that he has now automated with a Linux PC: Read on for further enlightenment.

By Charly Kühnast

Trådlös is the Swedish word for "wireless" and thus throws light on the name of the Trådfri smart lighting system. In the simplest case, you combine a lamp with an actuator: A remote control, a dimmer, and a motion detector are offered.

If you also want to use an Android or Apple smartphone to control the light source's brightness – and in many cases also modulate the light temperature from cold to warm white – you need a gateway, which is a small control box that connects to your Ethernet network and sends commands to the lighting systems. ZigBee Light Link [1] acts as the protocol in this case, but customers also can use compatible lamps with E14, E27, and GU10 sockets and LED panels that can be hung on the wall or built in to furniture.

That's all quite nice, but, of course, I do not want to control everything with an app. Instead, I want the lighting system to adapt dynamically to the ambient light, such as gradually becoming brighter at the onset of dusk. The Linux PC that evaluates the data from my photovoltaic system knows when it turns dark (otherwise, a simple brightness sensor will do the trick), and it is precisely this Linux PC to which I want to pass the Ikea gateway commands so that it can pass them on to the lights.

I do so with *libcoap* [2], a C implementation of the machine-to-machine Constrained Application Protocol (CoAP). The installation requires the commands in Listing 1. Doing this gave me the `coap-client` program, which I used to pass the parameters to the gateway that controls the lights.

Wrapped for Pleasure!

Because the syntax of the program is pretty horrific, I used a Python wrapper [3] that makes working with libcoap pure pleasure. The entry

```
sudo pip install tqdm
git clone https://github.com/➥
  sandyjmacdonald/ikea-smartlight
cd ikea-smartlight
```

installs it. I then dropped a file with the IP address of the gateway and its security code, which you will find on a label glued underneath the housing, into the current directory:

```
[tradfri]
hubip = 10.0.0.42
securityid = laskuhgoACBUWHG
```

I now have access to simple control and status commands. Figure 1 is the output from `tradfri-status.py`, showing that it has discovered two light sources that are combined into a group. Each lamp and the group have an ID that I can use for control purposes. For example, to switch all lights in the present group 133164 to 75 percent brightness, I just need the following command:

```
tradfri-groups.py -g 133164 ➥
  -a brightness -v75
```

By December 13, the Swedish St. Lucia celebration, I will definitely have programmed a suitable candle simulation for my living room. ■■■

```
pi@pi433 /usr/local/ikea-smartlight $ ./tradfri-status.py
[ ] Tradfri: acquiring all Tradfri devices, please wait ...
Tradfri lightbulbs: 100%|████████████████████| 3/3 [00:00<00:00, 16.72 lightbulb/s]
Tradfri groups: 100%|████████████████████████| 1/1 [00:00<00:00, 16.60 group/s]
[+] Tradfri: device information gathered
============================================================

bulb ID 65537, name: TRADFRI bulb E14 WS opal 400lm, brightness: 191, state: on
bulb ID 65538, name: TRADFRI bulb E27 opal 1000lm, brightness: 191, state: on

group ID: 133164, name: TRADFRI group, state: on
pi@pi433 /usr/local/ikea-smartlight $
```

Figure 1: The `tradfri-status.py` script reports two lamps in one group and their IDs.

LISTING 1: Installing libcoap

```
sudo apt-get install libtool autoconf automake buid-essential
git clone --recursive https://github.com/obgm/libcoap.git
cd libcoap
git checkout dtls
git submodule update --init --recursive
./autogen.sh
```

CHARLY KÜHNAST

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

INFO

[1] ZigBee Light Link: *http://www.zigbee.org/zigbee-for-developers/applicationstandards/zigbee-light-link/*

[2] libcoap: *https://libcoap.net*

[3] Ikea-smartlight wrapper: *https://github.com/sandyjmacdonald/ikea-smartlight*

Reading driving data via the on-board diagnostic port

Connected Car or Bust

A connector plugged into the diagnostic port of Mike Schilli's cars sends current information such as speed, acceleration, and fuel economy via the mobile phone network to a cloud service. An app and a programmable API read out the data and provide stunning visualizations. *By Mike Schilli*

I find it absurd to invest half a year's salary on a new car. Instead of more expensive German craftsmanship, I opt for second-hand Japanese cars with screaming VTEC engines from the 1990s. Alas, this type of car won't get you the electronic bells and whistles nowadays commonly associated with the "connected car." In my adopted home of San Francisco, a startup company called Automatic has developed a connector to the car's on-board diagnostic (OBD) port [1] (Figure 1) that continually reads data from the car's on-board computer and feeds the data either to the driver's cell phone or

MIKE SCHILLI

Mike Schilli works as a software engineer in the San Francisco Bay area of California. In his column, launched back in 1997, he focuses on short projects in Perl and various other languages. You can contact Mike at *mschilli@perlmeister.com*.

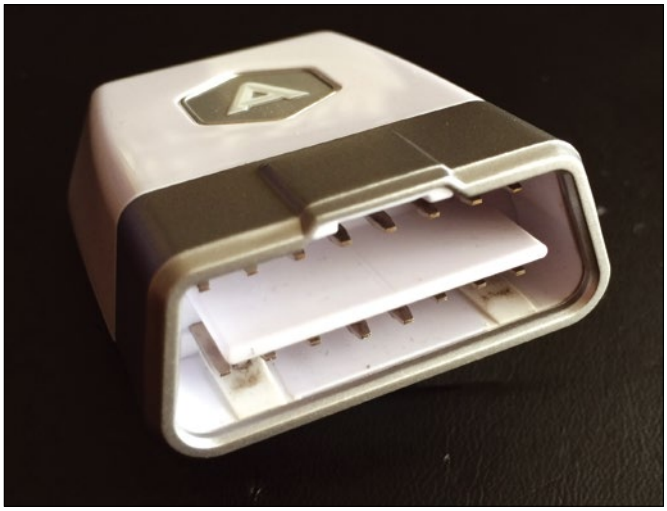


Figure 1: The Automatic adapter is plugged into the car's OBD-II port and wirelessly transmits the engine data over the 3G network.

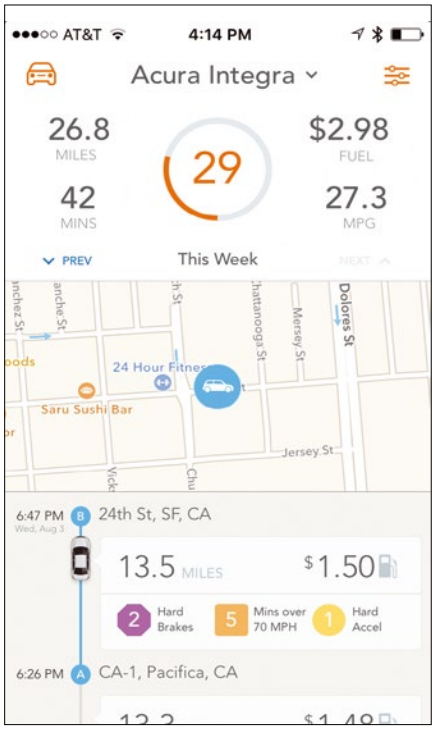


Figure 2: The Automatic app on the phone gives this spirited driver a meager score of 29 out of 100 when it comes to fuel efficiency.

straight to an account in the cloud via the 3G mobile network.

The data is collated on Automatic's servers. From this data, the driver can then check out their driving route, how often they stepped on the gas like a madman or hit the brakes in the same way, how much gasoline the car consumed, and what the trip cost.

Figure 2 shows that the Integra GSR under my care achieved a fuel economy of 27.3 miles per gallon. The low rating of only 29 out of a maximum of 100 points for economical driving could thus be greatly improved.

Even for Fleets

If someone has several cars, like me, each needs its own Automatic connector. The OBD-II port is often found under the dash in the footwell area, usually on the driver side, but sometimes also on the front passenger side (Figure 3). Once registered, the adapter then integrates seamlessly with various family cell phones on which the Automatic app is installed.

The adapter comes in two versions: Automatic Lite, a less expensive one that connects itself with the driver's cell phone via Bluetooth, and Automatic Pro, which has its own GPS unit, 3G mobile network card, and transmits data directly to the cloud. I've had the El Cheapo version for a while, but found it to work unreliably. At least 20 percent of the time, the adapter would fail to find the phone and not record any data at all. The Pro version is much better; it still misses the occasional trip, but it's probably 99 percent accurate.

Upon arrival at the destination, the adapter notices that the car is no longer moving; it then saves the current address as the destination of the trip and automatically uploads the data via the Internet to the server, without the driver even noticing.

You can display the trip data later via the app or on request in the browser on a desktop computer (Figures 4 and 5). Automatic also provides an API that returns machine-readable data via JSON to

Figure 3: The OBD-II port for plugging in the adapter is usually below the dashboard.

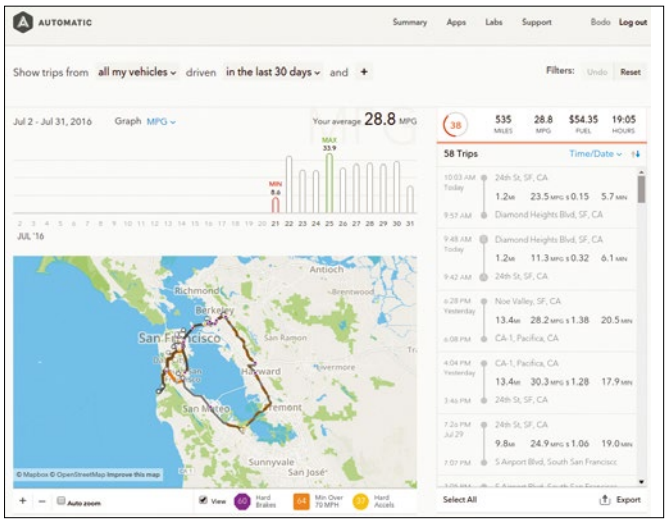


Figure 4: On the Automatic website, the user can see when they drove what car, where they drove to, how much fuel their car consumed, and how their performance was rated.

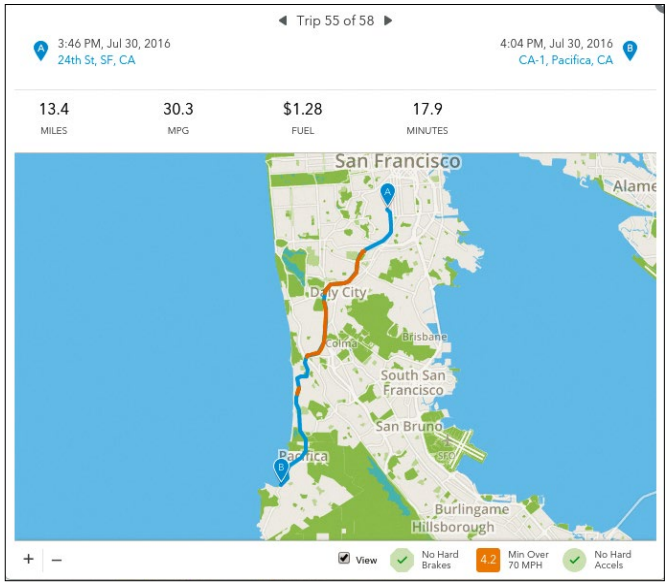


Figure 5: The drive to the beach in Pacifica took 18 minutes for a distance of 13.4 miles and cost me $1.28 for gas.

```
$ ./wheresmycar
      Honda-Fit-2011: 1-99 S Hill Ct, Oakland, CA 94618, USA
  Acura-Integra-1998: 3782 24th St, San Francisco, CA 94114, USA
$
```

Figure 6: A Perl script fetches the car data from the Automatic server and determines where the user's vehicles are parked.

registered users for downstream script processing and viewing.

Now, Where Did I Park?

We urbanites tend to crawl around the block endlessly until we finally find a free parking space. But later on, an important question often arises: "Now, where did I park?" Thanks to the Automatic connector, the answer is easy as pie; after all, in the Automatic universe, the car must be parked at the geographical end of a trip, unless it happens to be driving around somewhere in the meantime.

You can retrieve the data with the help of the Automatic API, for example, by using a Perl script at the command line. This requires the user first to register the application on the Automatic developer site at *https://developer.automatic.com*. After an OAuth token dance, you are then assigned an access token with which an API client, such as the one in the script in Listing 1 [2], can retrieve private data from the server on behalf of its user.

Listing 1 first uses the API call to `vehicle` to pick up the descriptions of all of the user's registered cars in lines 10 and 11. The `for` loop from line 13 then iterates through the fleet and calls the `trip` API method for each vehicle in lines 17 to 19 with the ID of the current vehicle. The API calls deliver data in JSON format to the client, and the `from_json()` function exported from the CPAN JSON module then converts the data to Perl's internal format, in which the code can poke around by accessing hashes or arrays.

The output in Figure 6 shows that the Acura Integra GSR, a model typically driven by belligerent urban youth, is now parked on 24th Street in San Francisco, whereas the Honda Fit, the commuter vehicle used by my better half, is currently parked in Oakland, where my wife works.

Logging a Trip

The resulting data is stored under the `results` key as an array, and, in the case of the call to `trip`, each of the elements contains a series of trip objects that in turn contain the coordinates of a trip, with information on the speed, the number of acceleration and braking maneuvers, and even the street addresses of the starting point and trip destination. The trips appear in reverse chronological order; thus, the first trip in the list is the last one to have been completed.

The destination stated under the `end_address` entry is the address at which the vehicle was last parked. The only task remaining now is to distinguish between the different vehicles in the fleet and output the brand, the model, and the year of manufacture of the automobile in question. (My garage actually used to accommodate two Integras of different vintages a while back.) Armed with this

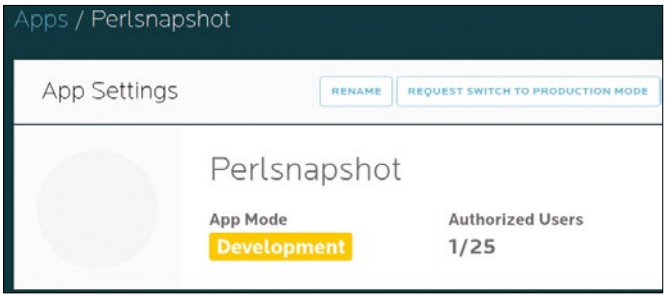


Figure 7: Before API access, you need to register the script with Automatic.

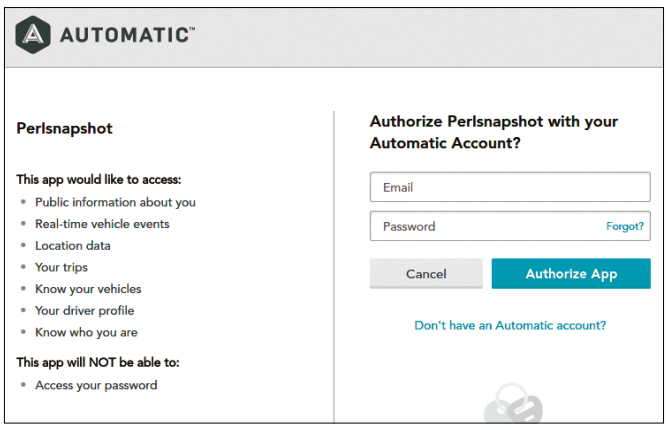


Figure 8: The locally launched web server takes the user to the login window of the Automatic website in order to authorize the Perlsnapshot application.

LISTING 1: wheresmycar

```
01 #!/usr/local/bin/perl -w
02 use strict;
03 use OAuth::Cmdline::Automatic;
04 use JSON qw( from_json );
05
06 my $oa = OAuth::Cmdline::Automatic->new();
07 $oa->raise_error( 1 );
08
09 my $vehicles =
10   from_json $oa->http_get(
11     "https://api.automatic.com/vehicle" );
12
13 for my $car (
14     @{ $vehicles->{ results } } ) {
15
16   my $trips =
17     from_json $oa->http_get(
18         "https://api.automatic.com/trip",
19         [ vehicle => $car->{ id } ]
20     );
21
22   for my $trip (
23       @{ $trips->{ results } } ) {
24     printf "%20s: %s\n",
25       "$car->{ make }-$car->{ model }-" .
26       "$car->{ year }",
27       $trip->{ end_address }->{ name };
28     last;
29   }
30 }
```

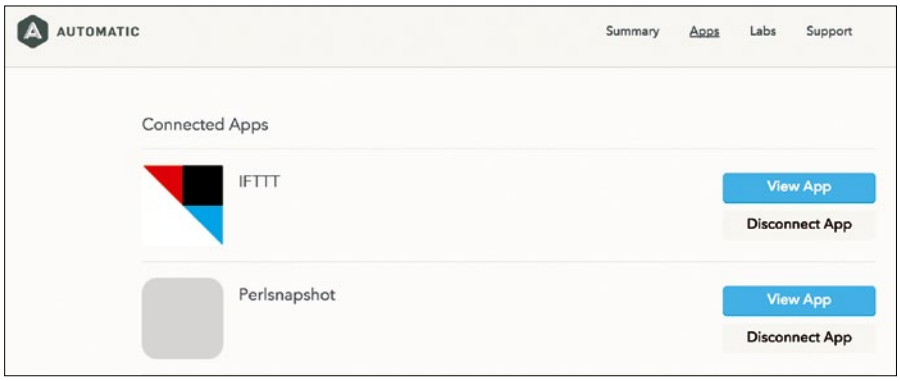


Figure 9: The Perlsnapshot app has been successfully integrated with the account.

information, plus the `name` field containing the trip destination, it is then clear where the respective car is located. The `last` command in line 28 stops processing the trip data after the first record; the `for` loop in line 13 starts a new round with the next car in the fleet.

Dancing with OAuth

How can a user authorize a script to pick up private data from the Automatic server on their behalf? Version 0.05 or newer of the CPAN OAuth::Cmdline module offers both helpers for providers such as Google Drive or Spotify, and an interface for Automatic.com. To guide the user through the token dance in a web browser, users first need to register the application on the Automatic developer site (Figure 7).

Under `My Apps`, press the + button and enter the following URL in the *OAuth redirect URL* field next to the name of the app (Perlsnapshot in this case)

```
http://localhost:8082/callback
```

so that the Automatic server, after completing the work, will guide the command-line client back to its own web server and store the access token it has received locally on the hard disk.

Automatic insists on release checking commercial applications before commissioning, but in test mode you can manage up to 25 user accounts without this inquisition. Automatic confirms the registration of the app with two keys: the client ID and the client secret. You need to drop both into the `.automatic.yml` file in YAML format in your home directory:

```
~/.automatic.yml
client_id: <XXXXX>
client_secret: <YYYY>
```

In the CPAN OAuth::Cmdline module's `eg` directory, you will find a script named `automatic-token-init`; when launched, it outputs a message indicating that it has started a web server on port 8082 of the `localhost`. If you then point your browser at *http://localhost:8082*, it shows you a link titled *Login on Automatic*, which takes you to the login page of the Automatic server at the push of a button (Figure 8).

Access Approved

Once you have logged in and agreed to let the new application, whose name is displayed on the site, retrieve user data (Figure 9), the Automatic server returns to the web server controlled by the init script, which then stores some other parameters, such as the access token in the `~/.automatic.yml` file, allowing scripts to pick up driving data in the future without requiring the user to enter a password.

The CPAN module provides the `http_get()` method for retrieving authenticated web requests. Behind the scenes, this then retrieves the access token from the YAML file and then neatly adds it to the request header, convincing the server to release the user's private data without much ado.

A detail on the fringe: Unlike other OAuth implementations, Automatic does not offer a refresh token; once issued, an access token remains valid for a whole year, after which the user has to do the token dance once again.

Painting by Numbers

The trip data recorded by the Automatic app contains other informational tidbits. The *path* field contains a cryptic string that logs the vehicle's current location every few seconds during a trip as the latitude and longitude values in what is known as polyline encoding [3]:

```
qzjeFb|hjVDkBh@gAY]hBiDrAwBt@_A...
```

Using the CPAN Algorithm::GooglePolylineEncoding module, you can quickly decipher the string of characters:

```
(37.74393, -122.43922),
  (37.74369, -122.43832), ...
```

The first value specifies the longitude, and the second value specifies the latitude of geographic locations through which the car moved.

During a 10-minute test drive to my go-to place for freshly baked rolls on weekends (the supermarket in San Francisco's Diamond Heights neighborhood),

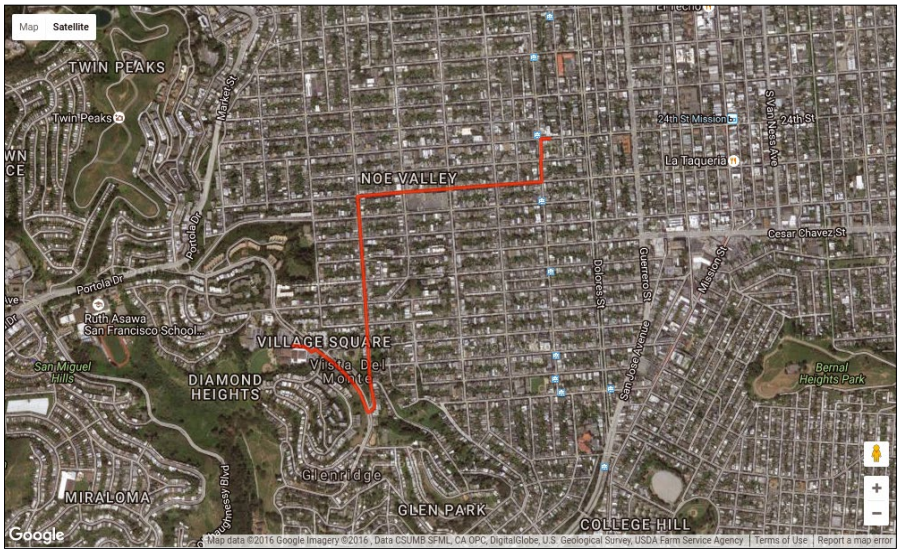


Figure 10: Morning drive for freshly baked rolls in San Francisco, drawn using the Google Maps API.

LISTING 2: trip-path

```
01 #!/usr/local/bin/perl -w
02 use strict;
03 use OAuth::Cmdline::Automatic;
04 use JSON qw( from_json to_json );
05 use YAML qw( Dump );
06 use Algorithm::GooglePolylineEncoding;
07
08 my $oa = OAuth::Cmdline::Automatic->new();
09 $oa->raise_error( 1 );
10
11 my $trips = from_json $oa->http_get(
12   "https://api.automatic.com/trip", [] );
13
14 my @locs = Algorithm::GooglePolylineEncoding::decode_polyline(
15     $trips->{ results }->[ 0 ]->{ path } );
16
17 print Dump ( \@locs );
```

a total of 38 data points were collected; Listing 2 decodes these and prepares them for visualization.

But how can you transfer the measured values to a map like the one shown in Figure 10? The Google Maps API provides a convenient interface; I actually used it in a previous column [4], where I introduced the `map-draw` script, which accepts latitude/longitude points in YAML format and produces JavaScript from them. It can also tell the browser to draw these points on a map via the Google Maps API.

Listing 2 thus only needs to produce YAML with latitude/longitude pairs. You can then call

```
./trip-path | ./map-draw >map.html
```

to create an HTML page, which you can easily load in your browser with *file:///…map.html* in the URL field to view an interactive map of the route.

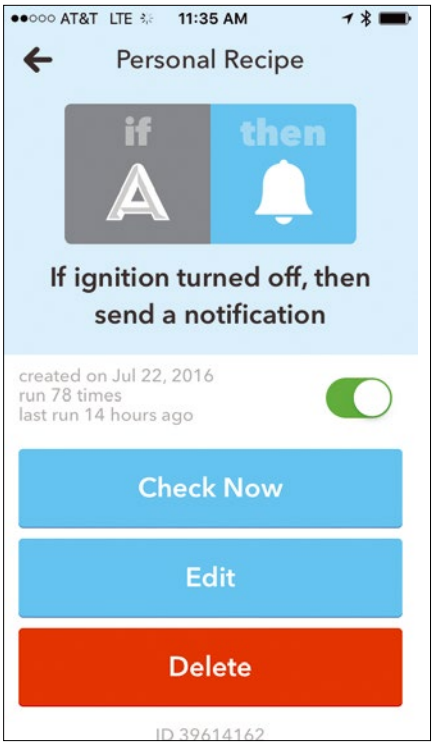


Figure 11: Integrated with IFTTT, the Automatic adapter sends push notifications to your cell phone when your car's engine is started.

Also in Real Time

In addition to the REST API that is used in the scripts, Automatic also offers a Real-time Event API, either via websockets or webhooks, that notifies applications practically instantly when specific events occur.

Upon starting your car, at the end of a trip, when a preset speed value is exceeded, on sharp braking, or when an indicator light comes on, the web service can send a message to the listening app, which it will then display on the lock screen of your locked cellphone and cause the phone to vibrate or play a sound if so desired.

The IFTTT web service discussed in a previous installment of this column [5] for the Wemo remote switch also makes integration easy in the case of the car connector. Instead of API integration, you need to register the app with the Automatic channel; you are then guided through the token dance – as with the CPAN module used previously – and prompted to grant IFTTT access to the Automatic data.

Figure 11 shows the IFTTT recipe that sends a short message to the user's cell phone whenever a driver switches on the ignition of the car.

Cost and Outlook

The Automatic Pro costs around $130 and includes five years of prepaid mobile network service in the US. The app is freely available, and there are no additional costs for logging and displaying the data. The Automatic Pro worked about 99 percent reliably during the test drives; it sometimes takes a while for a trip to be uploaded to the server if Automatic's servers are struggling or the adapter loses its mobile network connection at the end of a trip.

The adapter might even help in the case of a stolen vehicle – that is, if the culprit doesn't notice and unplug it first. There are already about two dozen more apps for iPhone or Android, and, thanks to the easy-to-use API, it should not be long until a whole ecosystem of other practical applications becomes available. ■■■

INFO

[1] "Automatic Pro review: Be a better driver with this driving data tracker in your car" by Glen Fleishman, *Macworld*: *http://www.macworld.com/article/3116113/car-tech/automatic-pro-review-be-a-better-driver-with-this-driving-data-tracker-in-your-car.html*

[2] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/magazine/202/*

[3] Encoded polyline algorithm format: *https://developers.google.com/maps/documentation/utilities/polylinealgorithm*

[4] "Climbing Aid" by Mike Schilli, *Linux Magazine*, issue 185, April 2016, *http://www.linux-magazine.com/Issues/2016/185/Perl-GPS-Data*

[5] "Click-Clack" by Mike Schilli, *Linux Magazine*, issue 189, August 2016, *http://www.linux-magazine.com/Issues/2016/189/Perl-IFTTT-Home-Automation*

REAL SOLUTIONS FOR REAL NETWORKS

FREE CD or DVD in Every Issue!

Each issue delivers technical solutions to the real-world problems you face every day.

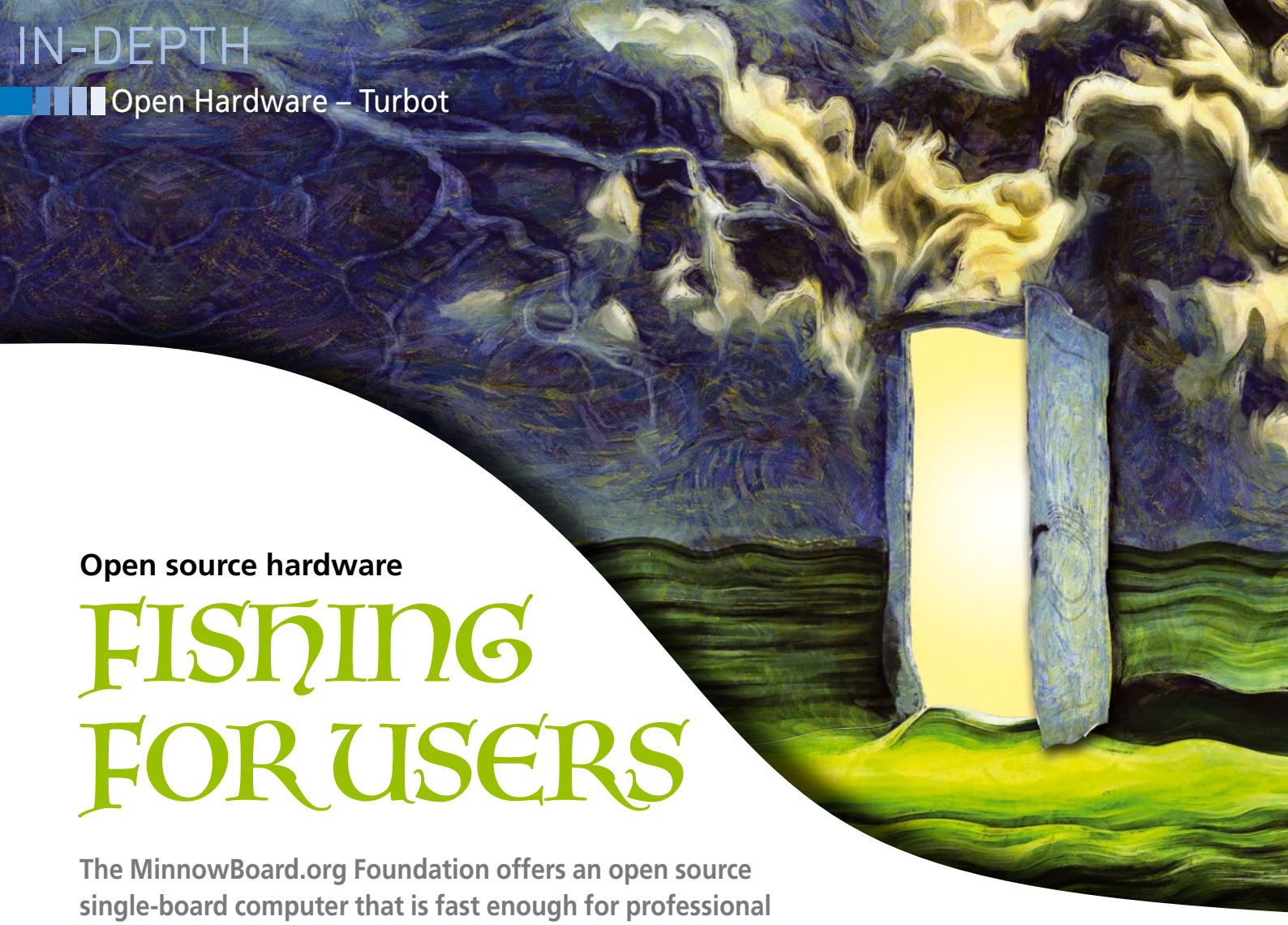
Learn the latest techniques for better:

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

on Windows, Linux, Solaris, and popular varieties of Unix.

6 issues per year!

ORDER ONLINE AT: shop.linuxnewmedia.com



Open source hardware

FISHING FOR USERS

The MinnowBoard.org Foundation offers an open source single-board computer that is fast enough for professional use, but accessible for all user levels. *By Bruce Byfield*

One of the main obstacles to open hardware is that the tools to make the tools are rarely available. Although many small motherboards exist, the specifications are not completely free-licensed in some cases. When motherboards are sourced, vendors often substitute them for a cheaper version, because they are more attuned to price than software freedom. However, the situation is slowly improving: witness the MinnowBoard.org Foundation's [1] recently released single-board Turbot computer [2], which includes a full set of specifications and free software from the firmware up (Figure 1).

BRUCE BYFIELD

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*

Located in Beaverton, Oregon, the MinnowBoard.org Foundation is a non-profit organization specializing in embedded computing on Intel architecture. The Foundation supports the Open Source Hardware Association [3] and makes its designs available for studying, modifying, distributing, and selling hardware based on that design. With an intended audience of both hobbyists and professionals, the Foundation also develops detailed tutorials that make its products broadly accessible.

In the past, the MinnowBoard.org Foundation released specifications and tutorials under Creative Commons By Attribution-Share-Alike 3.0 (CC BY-SA 3.0) license [4], which states that derivative products must acknowledge the original and be released under the same license. That remains MinnowBoard.org's preferred license, but in a statement released by the Foundation, they add that "some developers were hindered by the need to 'share alike' their added innovations.

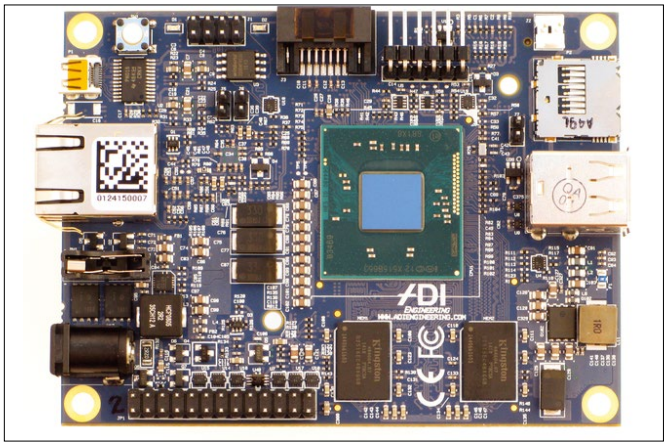


Figure 1: The Turbot, the MinnowBoard.org Foundation's latest release, is open source from top to bottom.

We recognized that there are developers that end up with a product idea that they want to productize, and we wanted to allow that avenue as well. As such, we have adjusted our default licensing to CC BY 4.0," which requires only the acknowledgment of original works [5].

Available in two- and four-core versions, the Turbot is the Foundation's third board, succeeding the MinnowBoard v1 [6] and building on MinnowBoard MAX [7]. In fact, it uses the same form factor, user connections, connector locations, and mounting holes as the MinnowBoard MAX and runs an updated version of the same software. Its main difference is the use of the Intel® Atom™ E3826 processor, which offers lower energy consumption and increases in code and graphics speed, as well as other general improvements in speed [2].

Supported operating systems include Debian, Windows 10 IoT, Android, Ubuntu, and Yocto Project, a distribution for building custom installations on embedded systems [8]. Free firmware supported includes free BIOSs, such as coreboot, SageBIOS, and TianoCore. The boards are manufactured by ADI Engineering, a division of Silicom Connectivity Solutions [9], and the complete technical specifications are published on GitHub [10].

In addition to the basic board, the Turbot can also add extension boards, or lures, as the Foundation calls them in an extension of the fish metaphor [11]. The half dozen lures currently available include boards designed for debugging, prototyping, and breadboard prototyping – that is, for temporarily arranging components during development. Other lures that are in development but not yet in production include boards for flying drones and extending Ethernet and USB capabilities. All these lures – many of which have marine names – should go a long way toward making prototyping quicker and more efficient.

The Foundation is planning on adding case studies to its website, but in the meantime, declines to be more specific than saying that its boards "are in use at Fortune 100 companies, universities, embedded in infrastructure devices, in routers, [and] test bed[s] for firmware development." The website adds that "the Intel® Data Plane Development Kit (DPDK) in a box development kit includes a MinnowBoard Turbot. You'll see the MinnowBoard Turbot being used in demos in almost every embedded and IoT focused event in the US and Europe because it is a versatile, low-cost, PC-like capable embedded board."

Tutorials Make the Difference

Boards are released every day, and although the Turbot seems outstanding in its specifications, by itself, the Turbot is newsworthy chiefly for its use of free licenses.

However, what is truly unusual is the care with which the Turbot is documented. With most boards, users are lucky to get a folded piece of paper giving the specifications and perhaps a link to slightly more detailed information online. The assumption is that anyone buying the boards knows enough to parse the specs and put them to use by themselves. By contrast, the MinnowBoard.org Foundation takes its educational purposes as seriously as its designs.

To start, the online details about the Turbot are illustrated by a diagram with mouseovers (Figure 2). Click a component, and a window opens with a brief explanation of what it does, along with several links that offer more detailed information.

The tutorials start by listing the hardware needed to power up the board, accompanied by a diagram of the board with everything connected (Figure 3) and an explanation of what to expect from the UEFI shell when booting for the first time, plus how to interact with it. The tutorials continue by showing how to install Ubuntu 16.04 (a long-term support release) on the board, as well as how to update the firmware. The tutorials end with the board equivalent of a "Hello, world!" script – a description of how to write a Bash script to make the board's lights blink.

Experienced users might complain that the tutorials are too basic. However, all levels of users should at least scan the tutorials to ensure that their expectations are in sync with the board's design. By the time users are finished, they should have a thorough understanding of what the board does and how to interact with it.

Evidently, the MinnowBoard.org Foundation understands that it is better to over-document than to assume what users know. By making this effort, the Foundation has made the Turbot and its predecessors accessible in a way that few pieces of hardware can claim.

Landing a Turbot

Turbot boards are just starting to appear in commercial outlets. As I write, they are available in limited quantities at

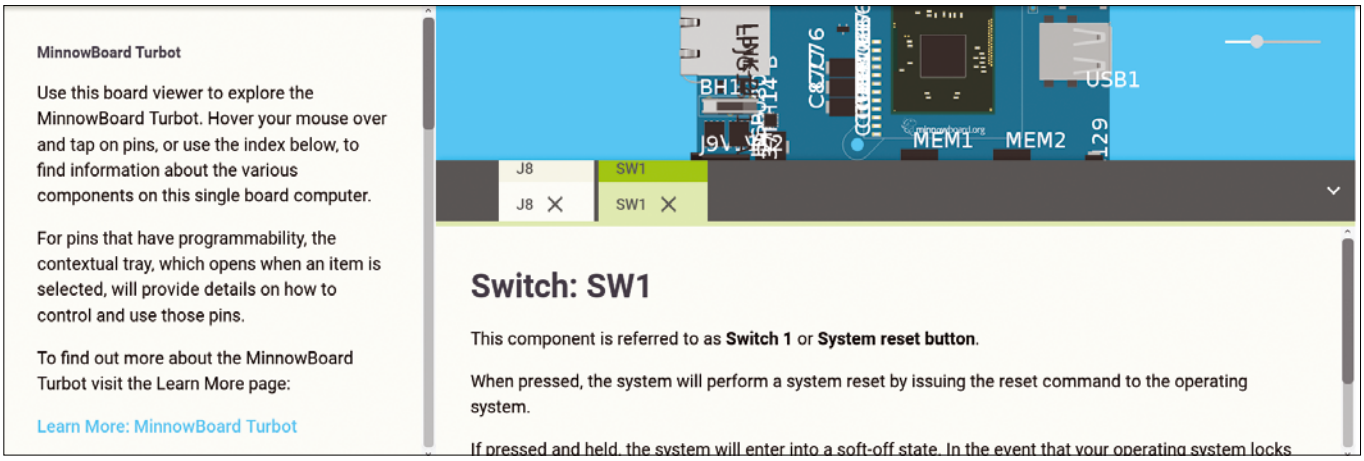


Figure 2: The Turbot's interactive diagram makes few assumptions about users' knowledge.

prices just under $200. One user on Amazon.com complained that the USB port was underpowered, but in general, the reception is positive.

Despite the competition among single boards, the Turbot has several advantages. Since it is intended for commercial use, and not just hobbyists, it is relatively high-powered. Unlike the Raspberry Pi, whose GPU remains proprietary, the Turbot is completely open source, although it can be used with two Windows versions. However, most of all, the MinnowBoard.org Foundation has made the effort to make the board accessible – and therefore usable – for all levels of users.

The Foundation expects to release another board by the end of 2017, but its features remain undisclosed. But if the approach to this design rivals that of the Turbot, the Foundation might have helped bootstrap open hardware simply by paying attention to its users. ■■■

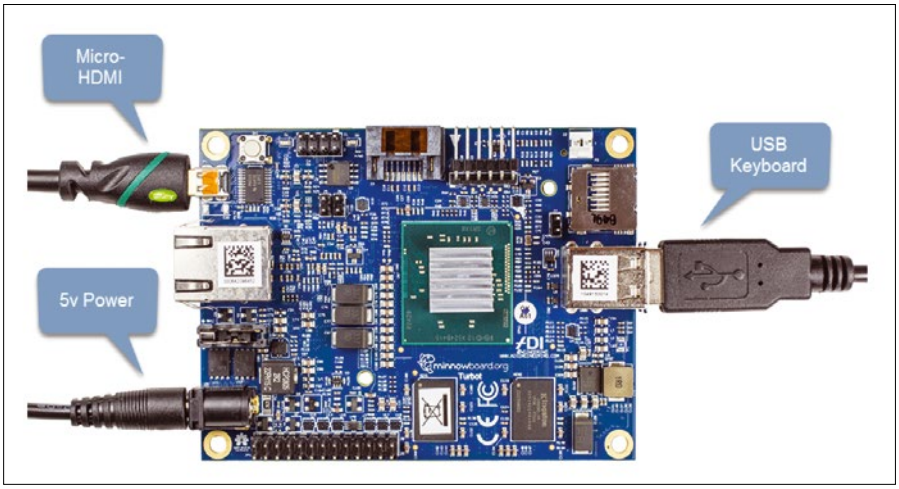


Figure 3: The Turbot's tutorials get users up and running in a matter of minutes.

INFO

[1] MinnowBoard.org Foundation: *https://minnowboard.org/*

[2] Turbot: *https://minnowboard.org/learn-more*

[3] Open Source Hardware Association: *https://www.oshwa.org/*

[4] CC BY-SA 3.0: *https://creativecommons.org/licenses/by-sa/3.0/*

[5] Previous MinnowBoard.org boards: *https://minnowboard.org/legacy-boards*

[6] MinnowBoard v1: *https://minnowboard.org/minnowboard-v1*

[7] MinnowBoard MAX: *https://techcrunch.com/2014/04/03/intel-releases-99-minnowboard-max-an-open-source-single-board-computer/*

[8] Yocto Project: *https://www.yoctoproject.org/*

[9] ADI Engineering: *http://www.adiengineering.com/*

[10] Turbot specifications: *https://github.com/MinnowBoard-org/design-files*

[11] Expansion boards (lures): *https://minnowboard.org/lures*

Command-line search engine

GOOGLER

With a few customizations, Googler plus a text-based browser offers faster, more accurate searching than a traditional web browser. *By Bruce Byfield*

These days, doing a web search from the command line may seem like an anachronism. However, Googler [1] can be useful in a number of ways. It provides a headless search engine that can be called within a script. When combined with a text-based browser, it gives you access to the Internet. Most important of all, Googler offers the same search options and presentation of results as the Google home page, but with the use of a few aliases, in fewer keystrokes.

Googler is available in many distributions and runs in the most common command shells. However, you can get the latest stable version by running the command `googler -u` as soon as you install. To obtain the latest, possibly unstable, version, you can add `--include-git` to the command as well. Alternatively, if the latest version seems more of a risk than you are willing to take, you can compile Googler yourself with the command:

```
make disable-self-upgrade
```

Additionally, if you plan on using Googler with a text-based browser, such as ELinks, Links, Lynx, or w3m, set the *Browser* environment variable with:

```
export BROWSER=[BROWSER]
```

To use a text-based browser temporarily, enter:

```
BROWSER=[BROWSER] googler query
```

Options

Googler uses the standard command structure `COMMAND OPTIONS STRING`. As in Google itself, the search string should be enclosed in double quotation marks (") when you want to search for an exact phrase. The majority of options are divided into two categories: search filters and customization of search results.

Search filters are similar to those available on the Google home page. The difference is that, while many search filters are available on Google only after you make the initial search, you can add search filters directly into the `googler` command and combine them more easily. Even if you are using `googler` without shortcuts (see below), the number of characters entered with `googler` is about equal to the number of clicks required on the Google home page, which means that a Googler search is somewhat faster once you have memorized the search filters.

As you filter a search, you probably want to begin with

```
--count=NUMBER (-n)
```

to set the number of results per page. (The entry in the parentheses is the short form of the option.) If you are previously familiar with the results, or you hope to sidestep Google's placement of personalized results at the top of the first page, you can specify the first result to display:

```
start=NUMBER (-s)
```

Using `--count` and `--start` together displays results starting with the one specified by `--start`. Other search filters include

BRUCE BYFIELD

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

Figure 1: The first two color-coded results of a search for the Gentium font.

```
tld=TOP LEVEL DOMAIN (-c)
```

to search in a specific version of Google – for example, *google.ca* for Canada rather than *google.com*. A two-character abbreviation is used for each country [2]. Similarly

```
--lang=LANGUAGE (-l)
```

returns only results in the language and locale specified using a two-character abbreviation for the language, separated from the two-character locale code – for example, `en-uk` is the English language as spoken in the United Kingdom [3]. In the same way,

```
--times= UNIT NUMBER (-t)
```

specifies how long ago the results were posted, with options for the units `h` (hours), `d` (days), `w` (weeks), `m` (months), and `y` (years); therefore, `-t=w3` would look for results posted in the last three weeks.

Search filters can also be used to specify a site. The option

```
--news (-N)
```

specifically searches Google News, and

```
site=URL (-S)
```

searches only the defined site. You can also enter

```
--first (--lucky, -j)
```

to limit output to the first result found – the equivalent of the Google *I'm Feeling Lucky* button.

Another set of options specifies the colors used to display results (Figure 1). By default, six characters are used to define colors. In order, they represent the indices, titles, URLs, metadata (for searches in Google News), abstracts, and prompts. Each can be assigned a basic, bright, or bold color option (Table 1) using the option:

```
--colors= SIX LETTER STRING (-c)
```

To use a set of colors permanently, set the environment variable `GOOGLER_COLORS`. These codes are entered with:

```
--colors= SIX LETTER STRING (-c)
```

Normal colors (`y`) and bold colors (`Y`) can also reverse foreground and background. If you prefer to have no colors, use:

```
--nocolor (C)
```

Navigating Results

When results display, you can open a result in your default web browser by entering its number at the bottom of the list. Keyboard shortcuts – which Googler help refers to as "omniprompt keys" – are available to help you navigate through the list (Table 2). Selections open in the web browser, but you must return to the results to navigate.

Shortcuts

Unlike most applications, Googler does not have a configuration file. However, you can store preferences for items, such as text-based browsers and color schemes, to save typing.

Also, Googler has several ways of storing searches. None of these choices are necessary, but you might explore them to see which suits you.

To start, Googler has autocompletion files for bash [4], fish [5], and zsh [6]. Another option that is installed with Googler is the googler @t add-on, which includes the `googler_at` [7] configuration file of aliases.

`googler_at` contains hundreds of abbreviated aliases (Figure 2), ranging from social media sites like Facebook and the Linux Kernel Mailing List to resources such as the Debian package search page and the Ubuntu man pages. To use `googler_at`, enable it by entering the command:

```
source googler_at
```

Although the message displays that the file cannot be found, you can then use the file's aliases. For example, entering

```
@w debian
```

searches Wikipedia for references to Debian. You can add aliases using the standard `alias` command. For example, the command:

```
alias m='googler -n 20 -c en -l us'
```

returns results in pages of 20 in US English. These aliases require some memorization, but, for ones you use regularly, they can greatly increase Googler's speed. Just be sure that none of Googler's aliases conflict with ones that you have already made.

Troubleshooting Notes

Googler does have some limitations. For example, searching for a single result may return nothing because the Google web service omits some results depending on your geolocation. In the same way, if the expected number of results are not shown, that is probably because you entered a larger number of results than actually exists. Nor can you get any

TABLE 1: Color Options

a	Black
b	Red
c	Green
d	Yellow
e	Blue
f	Magenta
g	Cyan
h	White
I-P	Bright versions of the regular colors above
A-H	Bold versions of the regular colors
y	Reverse normal colors
Y	Reverse bold colors

TABLE 2: Googler Omniprompts

`n`	Next search result
`p`	Previous search result
`index`	Return to complete list of search results
`f`	Jump to the first page of results, if any
`o [INDEX|RANGE|a]`	Move to indices, a range of results, or all results
`g [KEYWORDS]`	Do a search for keywords within obtained results
`q`, `^D`, double Enter	Exit Googler
`?`	Show omniprompt help
`*`	Start a new search using the original options

results from Google News if the country is set to Denmark (`dk`), Finland (`fi`), or Iceland (`is`), none of which have a news service.

Sometimes, problems can be solved by using

```
--exact (-x)
```

which disables Google's spelling autocorrection. On a server, you may also need to specify a proxy with:

```
--proxy=PROXY (-p)
```

However, if these solutions fail, you may find that Googler will not work as expected.

However, when Googler is customized with suitable aliases, you may find that it is a more flexible application than Google in a web browser. Like many command-line applications, Googler requires some memorization, but once you know the options you use most often, Googler becomes a match for a web browser or even surpasses it for speed and accuracy. ■■■

INFO

[1] Googler: *https://github.com/jarun/googler*

[2] Top-level domain abbreviations: *https://en.wikipedia.org/wiki/List_of_Google_domains*

[3] Language and local abbreviations: *http://www.science.co.il/language/Locale-codes.php*

[4] Bash autocompletion file: *https://github.com/jarun/googler/blob/master/auto-completion/bash/googler-completion.bash*

[5] Fish autocompletion file: *https://github.com/jarun/googler/blob/master/auto-completion/fish/googler.fish*

[6] zsh autocompletion file: *https://github.com/jarun/googler/blob/master/auto-completion/zsh/_googler*

[7] `googler_at`: *https://github.com/jarun/googler/blob/master/auto-completion/googler_at/googler_at*


```
1  # googler @t alias list
2  # Author: Arun Prakash Jana
3  # email: engineerarun@gmail.com
4  #
5  # To request key addition or removal upstream, please drop an email.
6
7      # A
8
9  # Amazon.com
10   alias @a='googler -w amazon.com'
11 # AlternativeTo
12   alias @alt='googler -w alternativeto.net'
13 # Android Developers
14   alias @android='googler -w developer.android.com'
15 # ARM Information Center
16   alias @arm='googler -w infocenter.arm.com'
```

Figure 2: `googler_at` **comes with a file of ready-to-use aliases, to which you can also add your own.**

Qt5-based image viewer PhotoQt

Doubtful Candidate

The lean PhotoQt tries to join the ranks of modern image viewers, but it's still not very stable.

By Karsten Günther

Various image viewers are available under Linux, but if you want to view RAW images, the choice becomes considerably restricted. You are left with veterans like Geeqie [1] and Shotwell [2]. Both offer far more than merely displaying the many different formats – each in their own way.

TABLE 1: Shortcuts

Shortcut	Function
O	Open image/directory
Home/End	Show first/last image
Left arrow	Show previous image
Right arrow	Show next image
+/-	Scale display
0	Set default resolution
Ctrl+F	Filter
R/L	Rotate left/right
Ctrl+H	Flip image horizontally
Ctrl+V	Flip image vertically
F2	Rename image file
Del	Delete image file
Ctrl+C	Copy image file
Ctrl+M	Move image file
E	Display settings
Ctrl+E	Show Exif info
M	Start slideshow; show Exif info
Shift+M	Start slideshow in the current directory

PhotoQt [3] – a relatively young project – is now trying to gatecrash the veteran party. Its first release, version number 1.2, dates back to January 2015. This article is based on the current 1.5.1 version. Many repositories already contain the software, and the installation does not typically cause any problems. On the homepage, you will find instructions for many distributions. On Ubuntu version 15.04, you can do:

```
$ sudo apt-add-repository ↵
  ppa:lumas/photoqt
$ sudo apt update
$ sudo apt install photoqt
```

to set up PhotoQt from the PPA for the project.

Requirements

If you use an image viewer for a larger number of images, some features are useful or even essential. These in particular include different ways to browse or sort the images. The program needs to

Figure 1: PhotoQt presents itself as simple and functional at startup time. The program's semitransparent interface offers little added value in practice.

Figure 2: The menu and other Windows remain hidden and appear only when you move the mouse pointer to the edges of the screen.

be able to evaluate the metadata contained in the images, that is, the Exif tags. Additionally, it should ideally offer an option for including reviews or keywords in the search.

In the case of RAW images, access to embedded thumbnails is a big help, and it accelerates the display in many cases. The RAW formats also contain metadata. It is particularly useful if you also have the option of using the raw data for the preview.

As the PhotoQt developer Lukas Spies states on the homepage, his interests are strongly focused on using the graphics processing unit (GPU). Today, this is often the most powerful processing unit in a computer.

Features

PhotoQt relies on modern technology with a combination of Qt5/QML. The program supports many formats through the use of GraphicsMagick and LibRaw. However it is pretty lame at times, such as when loading the thumbnail. And, PhotoQt only supports images in 32-bit color depth.

The program has functions for rotating, mirroring, and enlarging the pictures. You can directly rename, copy, move, and delete the image files. For many actions, keyboard shortcuts are available (Table 1); you can modify these if necessary and add your own.

The wallpaper function did not work consistently in the test. A slideshow dis-

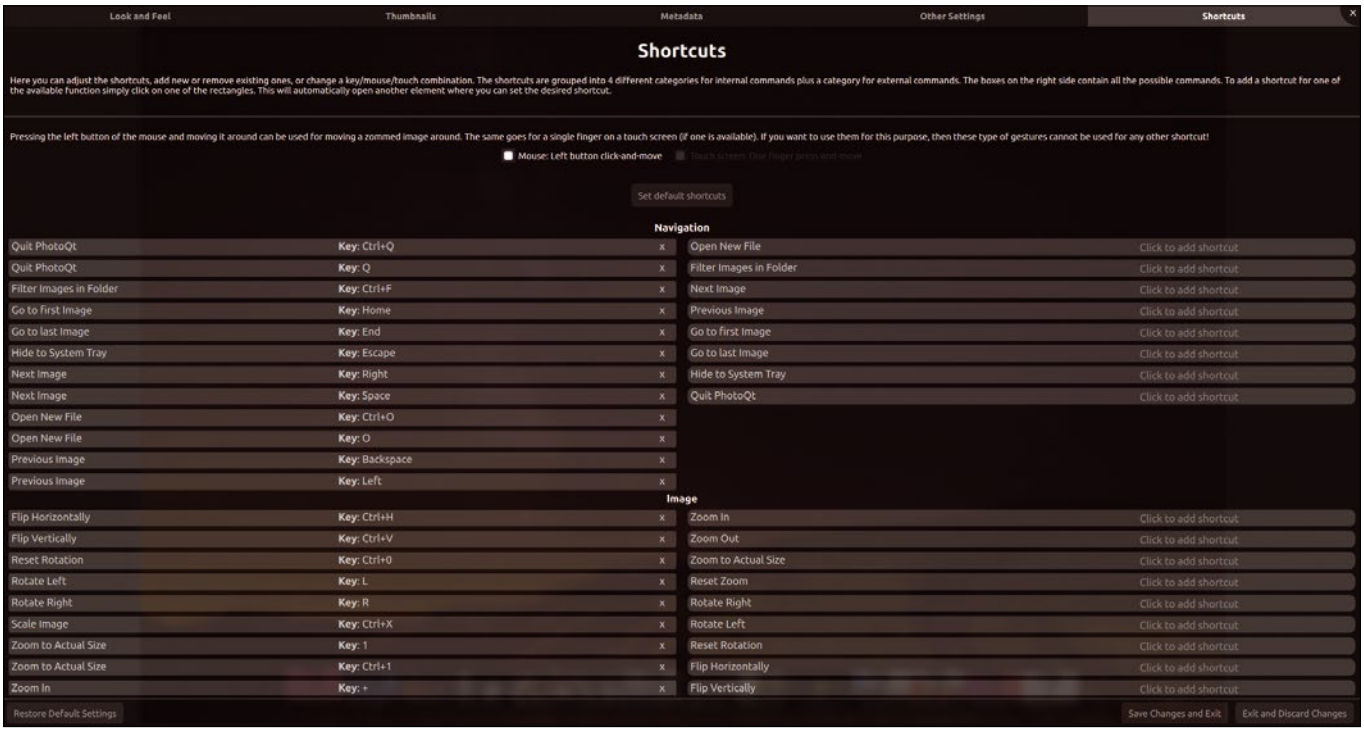


Figure 3: Below *hortcuts* users can customize the mappings for key and mouse events. You select additional actions in the right column.

plays a series of images without the need to switch manually. If necessary, the program can add the Exif information for the displayed images.

PhotoQt allows many customizations, more than many other viewers. It caches thumbnails of the loaded images. If you need quick access to the program, you can drop it into the system tray. Some rudimentary options let you control the behavior of the software when launched.

Hands On

By default, PhotoQt launches in full-screen mode and loads the images from the current directory. Alternatively, there is a mode with window decorations, and a several options that let you set the behavior of the program. Thanks to the semitransparent interface, you still see the desktop despite displaying images. This is modern but has no practical value (Figure 1). At the center, animated bars visualize loading the images.

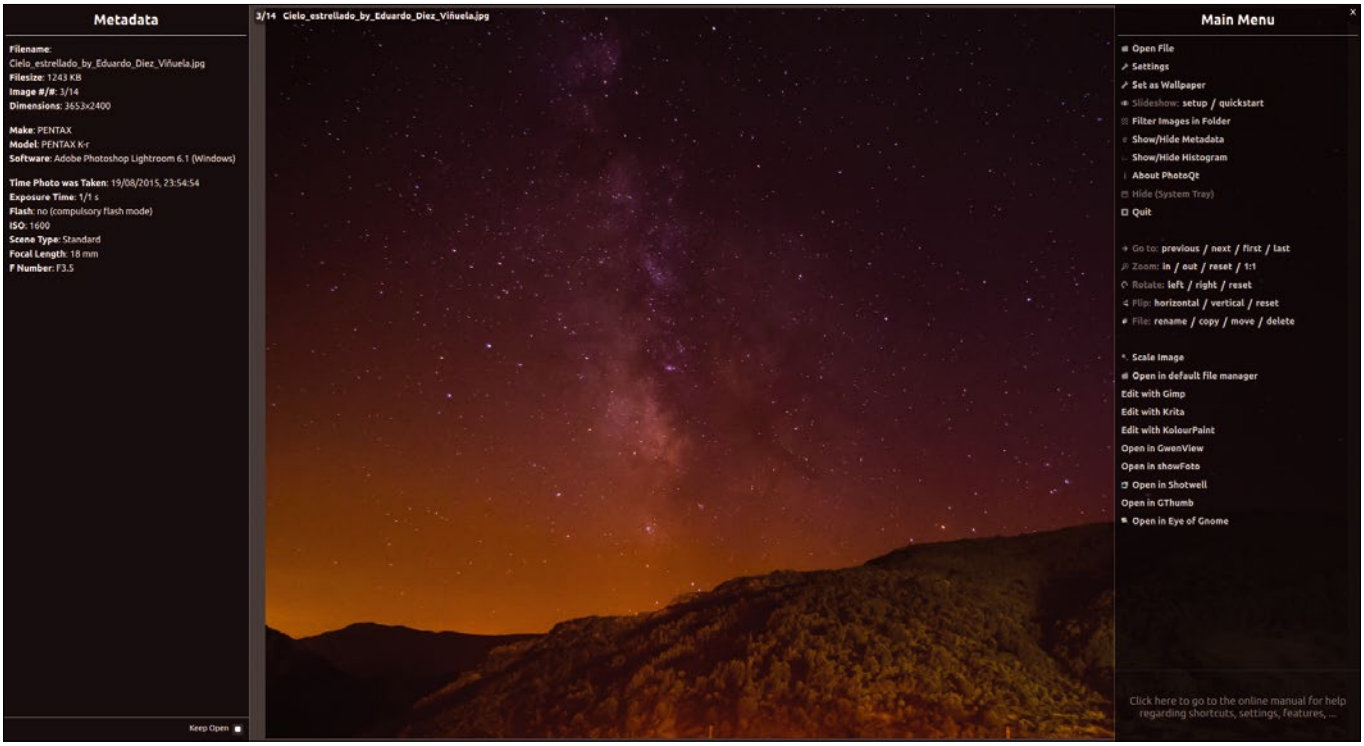


Figure 4: Although the program usually displays the important metadata from the EXIF tags, it exhibits some weaknesses especially with RAW formats.

By default, there only a few ways to toggle between images. For example, it is not possible to toggle between images with the mouse wheel. However, you can customize the controls to suit your needs. To do so, use the menu that opens up when you move the mouse pointer to the right edge of the screen (Figure 2). Below *Settings* is a comprehensive dialog with several tabs with which you can configure nearly all the aspects of the program.

Below *Shortcuts*, you can map keys and mouse events to functions (Figure 3). To change the images with the mouse wheel, first select the *Next image* function and then perform the desired action. Then, repeat the whole thing for the *Previous image* function. Then, select *Save changes and close* to quit the dialog.

If you were looking for a context menu that is quickly accessible with special functions in PhotoQt, you can stop now; this tends to make working with the program unnecessarily complicated. The online guide provides information about the most important settings [4] among other things. However, the document completely ignores the filters.

PhotoQt can display the metadata for images (Figure 4). Essentially the software is limited to the Exif information. The corresponding function still showed some weaknesses with most of the (raw) formats, delivering only a *File format is not supported* message; however, if necessary, you can display a histogram.

PhotoQt transfers the images to Gimp for processing – this is probably not the right way to handle RAW images. A RAW converter would be the better choice. However, Gimp does let you automatically call a converter (now including Darktable).

The Viewer uses a bar at the bottom of the window for the thumbnails (Figure 5). However, it only appears when you move the mouse pointer to the bottom. You can use the mouse to browse the images in the bar and click to select the one you want.

However, the main window does not always show the image under the mouse pointer in the preview bar, but the one that is highlighted in the bar – this is somewhat confusing at first. Caching also apparently doesn't work when displaying RAW images; the result being that image changes can take a very long time.

Conclusions

PhotoQt makes a good impression at first glance. In terms of controls, it does not differ significantly from predecessors like Geeqie. Nonetheless, the software lacks important functions for managing the images, which significantly restricts its use especially in larger image collections, even though the filters offer an interesting approach.

All told, PhotoQt still does not implement the concept of having a single application for touchscreens, laptops, and desktop PCs in an ideal way. For example, the absence of context menus forces many unnecessary mouse movements, which has an impact on effectiveness depending on the size of the desktop. But, it is still definitely worth keeping an eye on the project's development. ■■■

INFO

[1] Geeqie: *http://geeqie.sourceforge.net/*

[2] Shotwell: *https://en.wikipedia.org/wiki/Shotwell_(software)*

[3] PhotoQt: *http://www.photoqt.org*

[4] PhotoQt manual: *http://www.photoqt.org/man*

Figure 5: The bar with the preview images appears as soon as you move the mouse pointer to the bottom of the screen.

Subscribe now!

Don't miss a single issue of the magazine that delivers the in-depth technical solutions you'll use everyday!

GET IT NOW!
SAVE TIME ON DELIVERY WITH OUR PDF EDITION

shop.linuxnewmedia.com/subs

Ben Everard

Understanding modern Linux systems is both difficult and important. If you're working in technology, sooner or later, you'll hit a "weird" bug: the sort of problem that seems to come out of nowhere and, by all rational thought, shouldn't exist, yet somehow does and tends to be caused by something deep within the internals of the system. Personally, I take quite a perverse pleasure in these problems because they really push your skills and understanding.

When facing these problems, your key weapons are almost always old-school, low-level tools for probing the system. This month, both Valentine Sinitsyn and Mike Saunders take a look at some of these tools. In Core Tech, Valentine investigates nmap which is, in my opinion, the best network analysis tool. Mike, meanwhile, pokes the very fabric of executable files themselves. These are the sorts of skills you need to master if you really want to become a Linux expert.

These days, there's a good chance that if you're working on a large computer system with these sorts of ghostly problems, it is analyzing large amounts of data. Simon Phipps and I take a look at this problem from different angles. I investigate Spark, one of the leading tools for processing Big Data, while Simon takes a look at the regulatory and moral issues that these systems unleash.

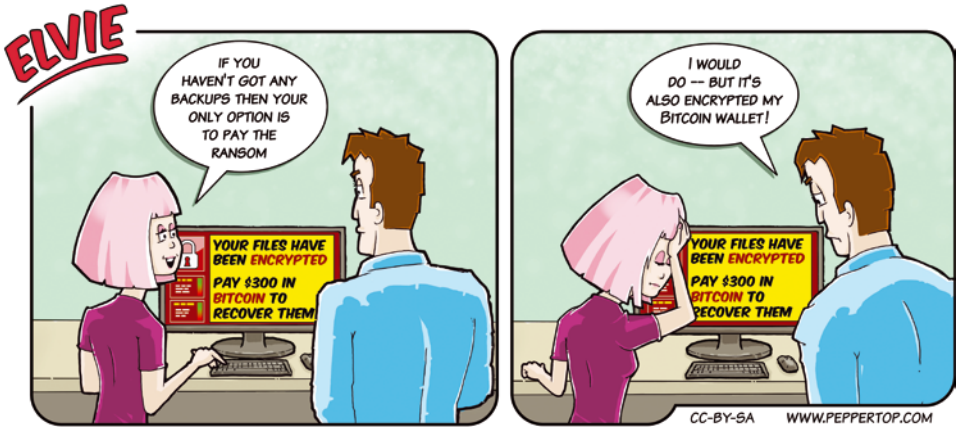
Of course, you don't always have to get your hands quite that dirty. If you prefer your Linux to be a little less industrial, Graham is here as always with his pick of the best new Linux software to make your computing a bit more entertaining, and Mike introduces Cherry Tree, which can help you get your life in order.

– Ben Everard

Andrew Gregory

Graham Morrison

Mike Saunders



LINUXVOICE

NEWSANALYSIS

The Linux Voice view on what's going on in the world of Free Software.

Opinion

Butter, Triangulation, and Data Protection

Data protection laws are not just about keeping personal information safe; they are also about stopping the derivation of personal insights from seemingly harmless information. BY SIMON PHIPPS

Simon Phipps is a board member of the Open Source Initiative, the Open Rights Group, and The Document Foundation (makers of LibreOffice).

At the end of May 2018, the new General Data Protection Regulation (GDPR) will come into effect in Europe. It creates a whole set of new responsibilities that are causing concern for businesses across the EU. It has effects outside Europe as well, because it will control the way businesses located in Europe can share data across borders, both within their company and with other companies.

While businesses are complaining about the new bureaucratic burden the GDPR creates, some privacy activists think it offers an absolute minimum level of protection in the emerging meshed society. This is not necessarily because of the way obviously confidential information is stored and used.

It seems obvious why we should be concerned about big chunks of personal data, but why should we care about protecting small details such as our date of birth, parents' names, postal code, and so on? Why does it matter when we're asked for them by someone with no need to know them?

The reason is, personal info can be used for *triangulation* to breathe meaning into otherwise anonymous data. Since they are likely to appear as a field in every database table that records information about us, they can readily be used as shared keys to allow data manipulation. What does that mean? Here's a (currently completely fictional) example to explain.

You are at the checkout in your local supermarket. They scan the pack of butter you want to buy along with the rest of your shopping, and then at the end of the packing as you prepare to pay, you hand over your supermarket loyalty card. The assistant looks at the screen, then reaches for the voucher printer, and pulls a form from it. He places it on the counter and gives you a pen. "Here, sign this." You look at it in surprise. It is a liability waiver, with your name at the top. The text simply says "I absolve the store of all consequences for my purchase of butter."

How did this happen? The store doesn't know your health status; they just know it's in their interest to get that waiver signed. Something like this happened:

- Their insurance company has used your name, postal code, and year of birth – effectively a unique identifier – as a "shared key" to identify you.
- They have used this shared key to ask holders of health records, past purchases at other stores, data from advertising tracking on web sites, and other information about you to each give a classification (without actually sharing the raw data), and they received back data about the data ("metadata") that they've then kept in a new database.
- At the time of purchase, they have fed this metadata into an actuarial model to see if anything in your basket triggers any indication of elevated risk.
- Because it did, and because people with similar profiles have sought class actions against contributors to health issues, they flagged you to this store as a litigation risk.
- The store gets a discount on their liability insurance if they get waivers from all at-risk customers, hence the waiver form.
- The store also gets a reduction on their insurance if they will provide metadata about your loyalty card purchases.

This is triangulation. No individual data item discloses private information you really care about, but gathered together, it can be deduced and used without consent. Triangulation is how anonymous data can be de-anonymized, and why assurances that your data has been anonymized and is thus safe to share should be treated with skepticism. The principle of least privilege (check it out on Wikipedia!) should inform us everywhere in our lives and is why we should support data protection laws like the EU's new GDPR. ■■■

Their insurance company has used your name, postal code, and year of birth – effectively a unique identifier – as a "shared key" to identify you.

Market Failure

BY ANDREW GREGORY

Free software isn't free – and we'll all end up paying

The downside of Free Software: Because it's free, the market doesn't work in the same way as for most commercial products. Some software, even the essential plumbing of the Internet on which we all depend, becomes an externality, and that makes it a problem.

What's an externality? It's a cost that you bear, or an advantage that you enjoy, that you haven't paid for. If a factory pollutes a river and the fishermen downstream find that their catch declines, that's an example of a negative externality affecting the fishermen.

The Free Software economy abounds with positive externalities, but the one that's got me thinking this month is GnuPG, the encryption software used by everyone and everything. It's fundamental to the Internet, in that Linux distros running 90-odd percent of the world's web servers use it to verify software updates. Without GnuPG, the Internet would need to come up with another way of keeping its servers up to date, which would cost money and effort. The money it saves – the trust it engenders – would be worth billions of dollars if only we could quantify it.

But, we can't count the benefits of GnuPG to the people who use it, and that means that nobody pays. That's forced the GnuPG team to start a crowdfunding campaign to raise enough money to pay for three full-time developers. It's crazy, and (after the Heartbleed SSL exploit, caused partly by underfunding of another crucial security tool) we should be worried.

Some companies, such as Red Hat, are making a massive success of Free Software. But for some projects, the market is failing. This is where a government would step in (in the case of the polluted river, with environmental protection laws). In the absence of such a supranational regulatory body for Free Software, however, it's up to us all to put our hand in our pocket and support GnuPG. Amazon, over to you. ■■■

Delve into ELF
Binary Magic

Discover what goes on inside executable files, how to reverse-engineer them, and how to make them as small as possible.

BY MIKE SAUNDERS

Back in the good old days, you could leave your door unlocked at night, music made sense, and writing computer programs was simply a case of putting some CPU instructions in the right order. Today, we have a mammoth range of libraries, toolkits, abstraction layers, and other things that make writing large programs easier – but it's increasingly difficult to understand what the CPU is actually doing. Open up LibreOffice, for example, and type a dot (period) character. What exactly happens here? How many CPU instructions are being executed between your finger hitting the key and that dot appearing on the screen?

Now, we don't want to sound like old codgers who think that everything should be written in assembly language. There's a reason why we have these layers of abstraction, to make software safer, easier to understand, and more portable. But sometimes it's good to go low-level and interact more closely with the CPU and operating system, to better understand what's going on. So, in this article, we'll get down and dirty with CPU instructions, the ELF executable format, and reverse-engineering binary files so you can see what they do.

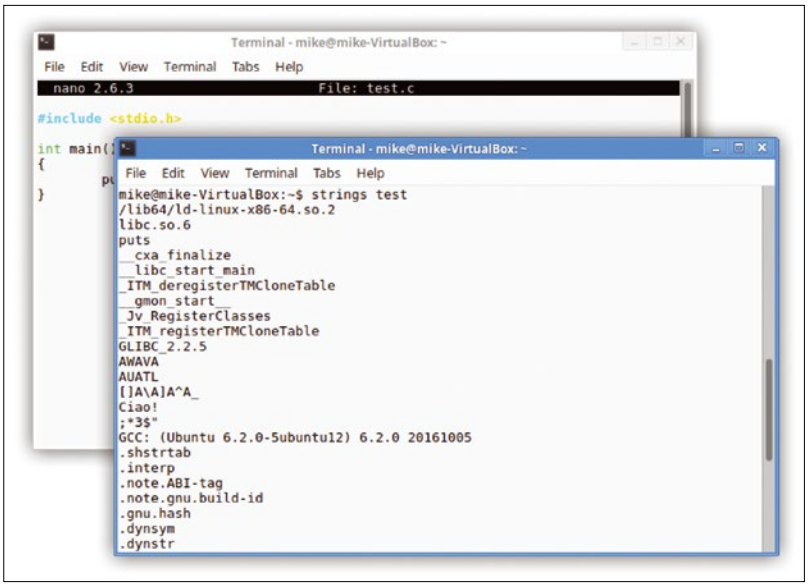


Figure 1: After compiling our simple C program, we can see the extra data that GCC puts inside the executable file.

I Can C Clearly Now

Let's start by writing a very simple C program. Put this into a file called `test.c` in your home directory:

```
#include <stdio.h>

int main()
{
        puts("Ciao!");
}
```

Now compile it in a terminal and then run it, using the following commands:

```
gcc test.c -o test
./test
```

As you'd expect, our "test" program simply prints the word "Ciao" on the screen, using the standard C library's `puts` (put string) routine – no surprises there. But enter `ls -l test`, and you'll notice something odd: The program is around 8KB in size! Sure, that may sound trivial in today's world of terabyte hard drives, but 8KB is actually huge for a program so simple. (Consider that space exploration classic Elite, back in 1984, was squeezed into 22KB of RAM [1]. That included a whole galaxy to explore, 3D spacecraft, missions, trading, and more. And yet our "Ciao" program is a third of the size.)

Well, this "test" executable includes some information generated by the compiler that we can use for debugging purposes. Let's remove that:

```
strip test
```

Now do `ls -l test` again, and you'll see that it's slightly smaller – down to 6KB. But that still feels overly large. The Commodore 64's operating system and support routines (aka "KERNAL") fit into 8KB, so we must be able to do better.

When we start poking around inside our "test" binary executable file, however, we see that most of the data inside it has nothing to do with the

printing bit. Run this command to see what kind of ASCII (text) data is stored inside the file:

```
strings test
```

You'll see results like in Figure 1. There are lots of text strings there generated by GCC that are of no importance to us, but if we look closely we can see the "Ciao" string somewhere among all the gobbledygook. Let's see exactly what kind of file `test` is, using the command `file test`. Your results will look something like this:

```
test: ELF 64-bit LSB shared object, x86-64,
version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2,
for GNU/Linux 2.6.32, BuildID[sha1]=
83b8ef795a8d78af706be36db142d9d64aca2307, stripped
```

Wow, that's quite a bit of info. The most important part is "ELF": This means that the file is in "Executable and Linkable Format," which is the standard format on all GNU/Linux systems. But what exactly is so special about this format? What does it do?

What's in an ELF?

Well, back in the days of early 8-bit and 16-bit computers and `.com` files on MS-DOS, executable files were simply bundles of CPU instructions and data. The operating system loaded them into a position in RAM and handed over execution to that position. There was no checking of the executable code beforehand, nor was there a distinct separation of code and data. It was a free for all, so to speak.

In modern operating systems, the situation is different. Multiple programs are run simultaneously, they can be loaded into many different places in RAM, and they are made up of multiple sections. In ELF files, like our `test` program, there's a "header" section that doesn't include executable code but provides the operating system with information about the program (e.g., the data we see in the `file` command above).

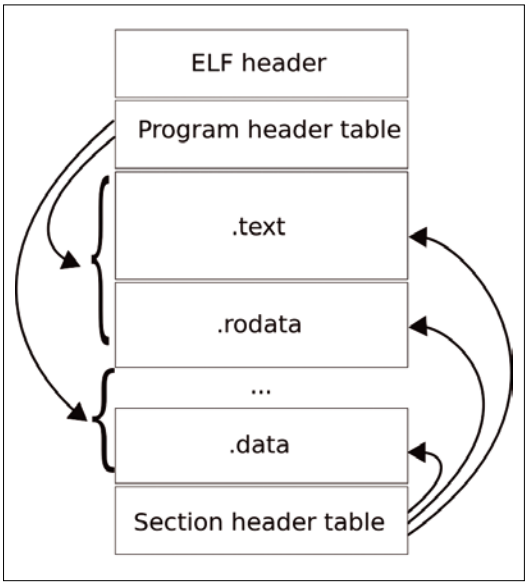


Figure 2: Here's a basic outline of the sections included in an ELF file (Image: Wikipedia).

Following this, the ELF file contains other sections: one for executable code, one for read-only data (like the "Ciao" string), and one for data that can be changed. Keeping these separate is an important security measure, as the operating system then knows which things can be changed, and which cannot. You don't want a compromised program able to start modifying its own code on the fly, for instance. A simplified view of ELF file structure is shown in Figure 2.

So, that's one reason why our "test" file is larger than we'd expect – but there are others. We can "disassemble" the executable file to show the assembly language that corresponds to the CPU instructions like so:

```
objdump -d test
```

This produces a lot of output, so you may want to pipe it through `less` to view it: `obdjump -d | less`. As you scroll around, look for the `.text` section, which actually contains the main code of our program, despite its name. If you've never seen assembly language before, you may be surprised at how many instructions are there, just to print a single word. Are they all necessary?

The answer is no. GCC and other parts of the compiler tool chain add a lot of boilerplate and setup code that's useful but not strictly needed. Most of this is added later in the compilation process. To see the raw assembly language that GCC initially generates from our C code, run this command:

```
gcc -S test.c
```

This generates a file called `test.s` – have a look inside it, and you'll see results like in Figure 3. The parts on the left, beginning with the dot characters and no indentation, are labels that point to specific parts of the code. You can see that the `.LC0` label points to our "Ciao" string, while the main program code begins at `.LFB0`.

Various assembly language instructions set up the program and memory, but the two that do the work of putting "Ciao" on the screen are these:

```
leaq    .LC0(%rip), %rdi
call    puts@PLT
```

We won't go into the specifics of assembly language right now, but in a nutshell: This code executes (calls) the standard C library's `puts` routine,

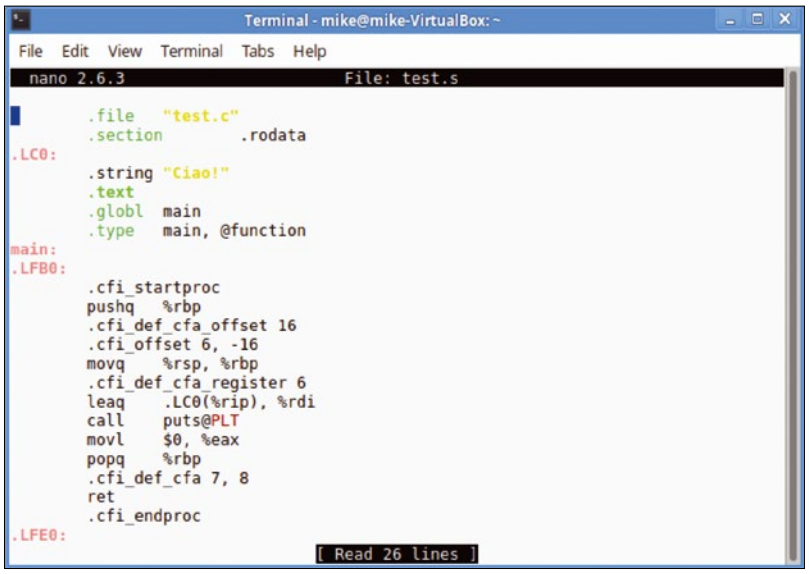


Figure 3: GCC can show us the assembly language instructions it generates from our C code.

giving it the location of the text string to print. Once the C library has done its work, it hands control back to our program, which does a bit of cleaning up before it ends with the `ret` instruction (return – basically, give control back to the operating system).

You Can Go Your Own Way

So, we've poked around inside an executable generated from a C file and done some reverse-engineering on it; now let's look at making the program as small as possible. One thing we want to do is remove our dependency on the GNU C library (*glibc*). Running this command

```
ldd test
```

shows the libraries on which `test` depends – and one of them is *libc.so.6*. The output of `ldd` shows where that library is on your system (it's probably a symlink to another file), so with `ls -l` followed by the full filename you can see how big it is. On our system, the C library weighs in at 1.8MB, but if you're running a super-sized Gentoo setup with optimizations galore, you may have shrunk it down a bit. In any case, it's a hefty dependency that we'd like to get rid of.

But, how do we print a message on the screen, without using `puts`, `printf`, or other common routines from the C library? Well, we can actually get the kernel to do the work for us. The Linux kernel includes a bunch of system calls for doing crucial tasks: opening and closing files, starting processes, and basic input and output. Many standard C library routines act as fancy wrappers around these system calls, adding extra features and checks to reduce bugs, which is why few programs interact directly with the kernel. But we're going to do it!

We'll write a short assembly language program that does the exact same job as `test.c` created earlier. (Note that we're using 32-bit x86 assembly language code here, so it'll work on 32-bit and 64-bit Intel/AMD PCs, but not on other architectures like the Raspberry Pi.) To convert the assembly code into an executable, we'll use the NASM assembler, so install it from your distro's package manager or – on Ubuntu-based distributions – enter the following:

```
sudo apt-get install nasm
```

Then, enter the following into a text editor and save it as `test2.asm` (Figure 4):

```
section .data
        msg db "Ciao!", 10

section .text
global _start
_start:
        mov ecx, msg
        mov edx, 6
        mov ebx, 1
        mov eax, 4
        int 0x80

        mov eax, 1
        int 0x80
```

Assemble it into a binary executable file (`test2`) and run it using these commands:

```
nasm -f elf test2.asm
ld -m elf_i386 -s -o test2 test2.o
./test2
```

Et voilà – "Ciao!" is printed on the screen, just like with the C program we created at the start of this tutorial. But this program is very different, in that it uses a kernel system call to display the text on the screen.

At the start, we set up a "data" section, which contains our "Ciao!" text string. This is put next to a label called `msg`, which identifies exactly where the string can be found. Note that we end our string with the number 10, which is the ASCII character for a line feed (like pressing enter – see the online ASCII chart [2] for a reference).

So with our string prepared, we can start writing CPU instructions and talk to the Linux kernel. You may recall that the "text" section is the one that contains code, rather confusingly, so we start with this section. We immediately create another label called `_start`, which points to the beginning of the code – this is used by the operating system to determine exactly where in the file the program begins.

Next up, we need to populate some registers with important data. Registers are a bit like variables, in that they can store many different values, but they are actually memory storage spaces built in to the CPU. Working with them is extremely fast, compared to regular RAM, but there is a very limited set of registers.

Anyway, before we tell the Linux kernel to display the string, we need to provide it with some information. First, we put the location of the string into the `ecx` register. (NASM instructions go from right to left, so the first `mov` here means move – actually copy – the value of `msg` into the `ecx` register.)

Second, we need to determine how many characters in the string we want the kernel to print. With the exclamation point and trailing line feed (10) character, that's six characters in total, so we put that in the `edx` register. Then we put 1 and 4 into the `ebx` and `eax` registers, respectively, which tell the kernel which specific system call to use (in our case, `write`) and where to print the text (stdout).

With all the registers set up, the `int 0x80` instruction does the magic of "interrupting" our program and handing control over to the Linux kernel. The kernel looks at the `eax` register and thinks: "Aha, the calling program wants me to run the 'write' system call. Let's see what's in the other registers, to find out where the string is, how long it is, and where I should display it."

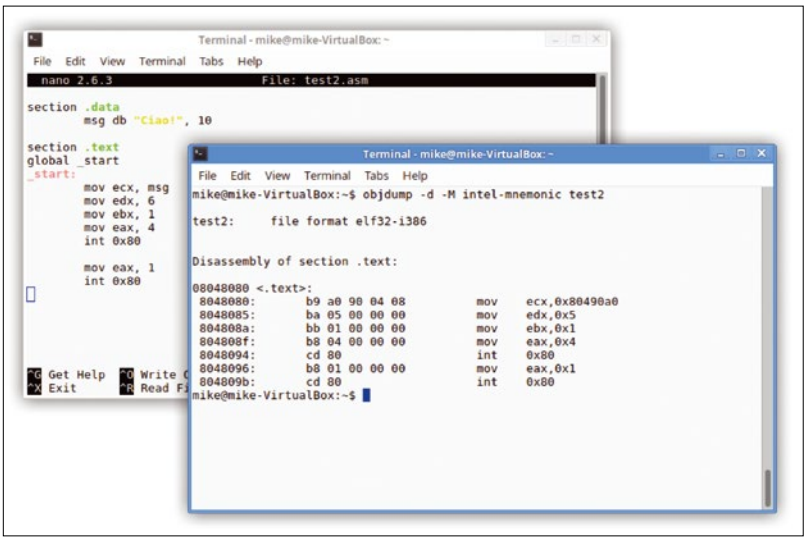


Figure 4: Reverse-engineer your own binary! Run `objdump -d -M mnemonic-intel test2`, and you'll see the same instructions as in `test2.asm`.

Once the kernel has done its work, it hands execution back to our program. Then we put 1 into the `eax` register and call the kernel again – this time the 1 value tells the kernel to safely terminate our program. And that's it!

Now run `ls -l test2`, and you'll see that the executable is down to around 350 bytes! That's way, way smaller than the C equivalent. We've still created a valid ELF executable file, but there's none of the extra startup and cleanup code added by the C compiler, nor are we using a C library.

And guess what? It's possible to make this executable even smaller! This involves some rather advanced tricks and hacks, but if this tutorial has whetted your appetite for minimalism, check out the fascinating "Creating Really Teensy ELF Executables for Linux" guide by Brian Raiter [3]. (See the "ELF and ARM" box for more information.) ■■■

ELF and ARM

Although we focused on x86 assembly language in the latter part of this tutorial, ELF files are not constrained by any particular CPU architecture. If you have a Raspberry Pi and go into the `/bin` directory, for instance, and run `file` on a few of the executables there, you'll notice that they're also ELF files – but for the ARM architecture.

ARM is arguably a much more elegant and better designed instruction set than x86. The latter has a more limited set of registers (in 32-bit mode), and some instructions can only be performed on certain registers. There's also baggage everywhere, due to backwards compatibility over many decades. So, if you really want to get into assembly language, we recommend going with ARM first.

Sure, it's not architecture used by most desktop and laptop PCs, but it's absolutely everywhere – in smartphones, embedded devices, and of course the Raspberry Pi. We ran a tutorial on ARM assembly previously that you can find on the website [4]. One especially fun ARM device to play around with and write assembly code for is the Nintendo Game Boy Advance. It's a fairly simple machine compared to the Pi, but you can do a lot with it. Parater [5] has a pretty good outline of the essentials, covering common ARM CPU instructions and how to interface with the Game Boy Advance's hardware.

Info

[1] Elite: *http://news.bbc.co.uk/2/hi/technology/8261272.stm*

[2] ASCII chart: *http://asciichart.com*

[3] ELF Executables for Linux: *http://www.muppetlabs.com/~breadbox/software/tiny/teensy.html*

[4] ARM assembly tutorial: *https://www.linuxvoice.com/creative-commons-issues/*

[5] Parater: *https://patater.com/gbaguy/gbaasm.htm*

FAQ

Flathub

A distro-agnostic software repository set to take the Linux world by storm.

BY BEN EVERARD

Q Let me guess, Flathub's the center of Terry Pratchett's *Discworld*?

A Nope. For starters, at the center of *Discworld* lies a great set of mountains. Hardly the sort of thing to be called flat. Flathub (Figure 1) is a repository for applications packaged in Flatpak.

Q Ok, let's go down a level. What's Flatpak? I assume Ikea hasn't branched out into software distribution.

A No, they haven't. At a very basic level, Flatpak is a way of sandboxing applications so that they're more independent from the rest of the system.

Q What do you mean, independent?

A Ok, let's look at it this way. Currently, most Linux systems install software in packages. Any package can require that many other packages be installed, and those packages can require further packages, etc.?

Q Yeah, I've always wondered why we need to do this in Linux, but it's not done in Mac OS or Windows.

A Almost all software is built on libraries that provide common functionality. For example, the Gnome libraries provide tools for building windows and user interfaces. A bit of Gnome software uses these libraries. There are basically two ways of doing this. One is to have a single version of the library on the system that every bit of software that needs the library uses, whereas the alternative is to have each bit of software come bundled together with everything it needs.

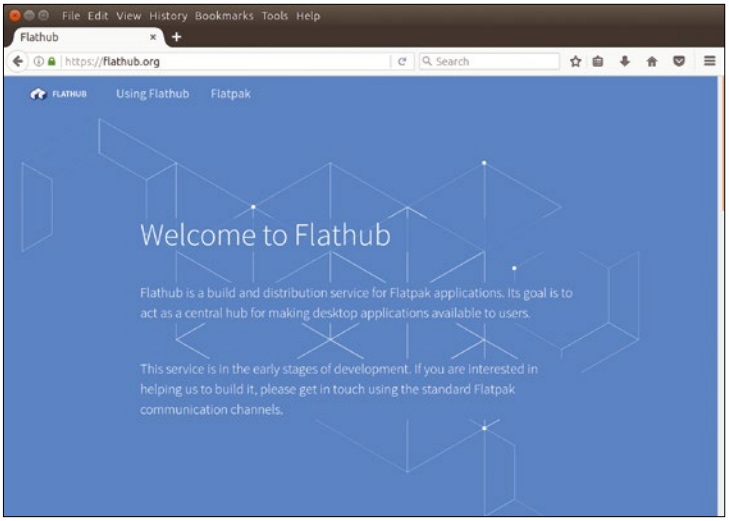


Figure 1: The Flathub website [1] is still a little sparse due to the software being new, but eventually it will become the place to find out what's going on with the project.

Q That second way seems really wasteful – why have many copies of something when you only need one?

A Yep. For years, Linux has traditionally used the first approach, and it does have advantages. It uses less space, and if there's a bug fix, you only have to fix the library once. However, it does also have a problem. Bits of software often depend on a specific version of a library (or at least have minimum and maximum version numbers they can work with). These libraries themselves can also require specific versions of other libraries, and these other libraries...well it goes on and on. Essentially, the software in a Linux distribution is a complex web of interdependent version numbers. Normally, this isn't something you have to worry about because the distro maintainers deal with the problem for you. If you grab something from your package manager, it'll make sure that everything is in appropriate versions. However, if you install something from outside the package manager, you're on perilous footing. Not only do you have to make sure that it was built for your particular distro, it can also break if you upgrade other parts of your system.

Q So why not just install everything from your package manager?

A If you can get everything you want from your package manager, then great; stick with that. However there are a few occasions where this doesn't work out. One example is if you want the latest version of a bit of software and it's not yet in your repository (in other words, if you don't use Arch). Another example is when the software simply isn't available for your distro; maybe it's obscure, or maybe it's brand-spanking-new. The final reason is for

software that the distro maintainers can't recompile or ship (yes, we're talking about proprietary software here).

Flatpak makes it possible to ship software without getting tangled in the web of interdependent software in a Linux system.

Q Hang on, is all this Flatpak stuff just a way to get non-free software onto my system?

A No. It can be used for that, but really it's just a method to make it easier for developers to ship Linux versions of software. Let's flip our viewpoint for a minute. Rather than being a Linux user, suppose you're a software developer writing code that can be compiled on Windows, Mac OS, and Linux.

Q Ok, I'm picturing me at a standing desk in Shoreditch, London with a carefully groomed moustache tapping away on my MacBook air.

A That's a wildly unrepresentative image of software development, but never mind, we can go with it.

You know that Windows will make up the majority of your user base, so you're also happy to spend a bit of time supporting them even though (as you decided), you're developing on a Mac. However, there are only a few versions of Windows you really need to worry about (five if you go all the way back to XP, which even Microsoft only supports if it's giving them really bad publicity). That's five different versions you need to compile and test that make up the majority of your user base. A further version or two for Mac OS.

Then it comes to Linux. Obviously you'll support Ubuntu because that's the most popular Linux distro (there are five current versions of Ubuntu), but if you only support Ubuntu, you'll get complaints from people using other distros. There are two current Fedora releases, two of openSUSE, and countless others. Not only are there all these, but they change, and you have to keep making sure your software is updated to work with newer versions of libraries on all this software. And this is without considering the issues associated with supporting rolling releases such as Arch or openSUSE Tumbleweed.

The end result is that, unless you really care about Linux, you just won't bother because it's a lot of hassle and you're not going to get many users out of it.

Q Garh, that sounds like a hassle. I'd much rather be sipping a flat white in a coffee shop with exposed brick walls than dealing with all that software packaging.

A I don't know where you've got your impression of a software developer from, but it's quite alien to the programmers I've met. But still, packaging software isn't fun, and packaging it for lots of different Linux distros to get a small number of users is not a useful way of spending time.

Wouldn't it be so much easier to just have one way to package it and one way of distributing it that went to all Linux distros?

Q Yes. I would want that so much.

A That's what Flatpak and Flathub are: a way of packaging (Flatpak) and distributing (Flathub) software to all Linux distributions in one go.

Q Ok then, how does it work?

A The simple answer is that it bundles everything into a single package – all the libraries and everything else – and this is run in a standalone way. There are some quite advanced things going on behind the scenes (Figure 2), but the basic concept is really straight forward.

Q Are there any advantages besides making it easier to package stuff?

A Yep. With everything decoupled, you can install whatever version of whatever you want – including multiple versions of software that depend on different versions of libraries.

Also, the way it works has some security advantages because each application can be far more isolated from the rest of the system in terms of what it can access.

However, while both of these benefits are useful, we think that by far the biggest advantage is that it'll bring more software to more distros and make it easier to keep up with the latest versions of software.

Q Awesome! How do I get started?

A Well, the starting point is Flatpak, which you'll need to install through your package manager (there are more instructions at the project website [2]). Once you've got Flatpak up and running, you can install software, but it's far easier to get it via the central repository, Flathub. Try this single-line command:

```
flatpak remote-add ➥
--if-not-exists flathub ➥
https://flathub.org/repo/➥
flathub.flatpakrepo
```

Once that's run, you can grab software with the Flatpak `install <package>` command. ■■■

Info

[1] Flathub: *http://flathub.org/*

[2] Getting started with Flatpak: *http://flatpak.org/getting.html*

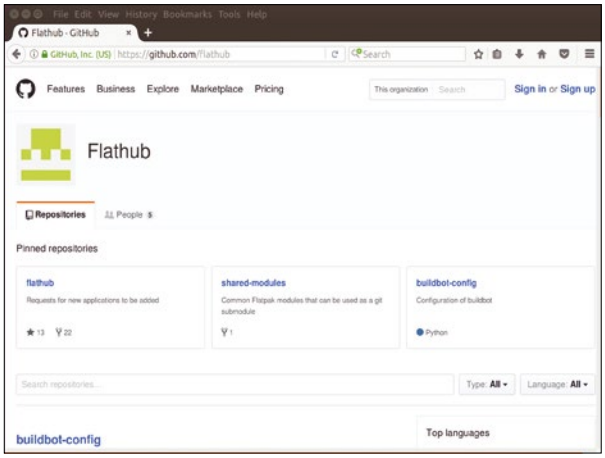


Figure 2: Software in Flathub is managed through GitLab repositories. Want to add a new piece of software? Submit a pull request.

Valentine Sinitsyn works in a cloud infrastructure team and teaches students completely unrelated subjects. He also has a KDE Developer account he's never really used.

CORE TECHNOLOGY

Network scanning may carry a negative connotation, but it doesn't mean you shouldn't look for weak spots in your network.

BY VALENTINE SINITSYN

Network Scanning

Imagine you are administering a small office or home network. Perhaps you want to know what hosts in this subnet are currently online, or which service that Internet of Things (IoT) device keeps open to the world. Network scanners are tools built to do just that.

Even if a host is properly secured and has unused ports closed, a network scanner may tell quite a lot about it. There are slight discrepancies in how popular operating systems (OSs) implement network protocols such as TCP. A tool that knows these nuances can make an educated guess about which OS the host runs. This is known as OS fingerprinting, and many network scanners implement it as well. Sometimes, it can even give you an uptime estimate!

As you guessed already, this Core Tech is about network scanning. Before we dive in, a usual word of warning: As with many technologies, network scanning can be used for good and for evil. Many network attacks begin with it, so it is deemed illegal in some provider and corporate networks. Never scan a network you don't really own unless you have permission to do so. When in doubt, a purpose-built scan target, *scanme.nmap.org*, is a good choice.

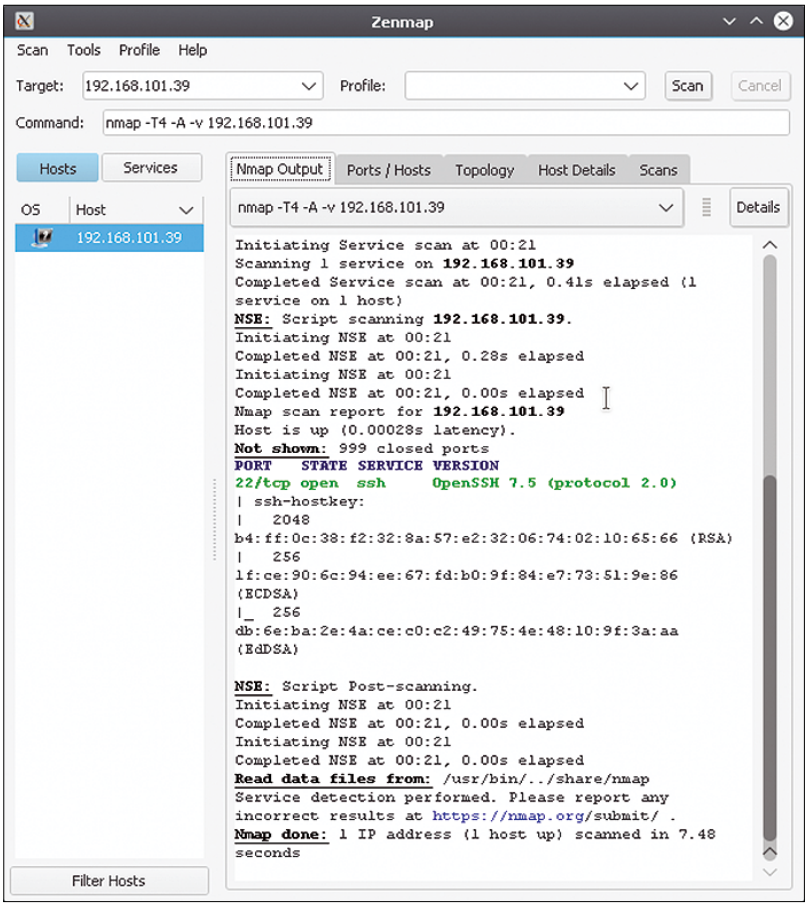


Figure 1: Zenmap wraps Nmap goodies in a simple yet feature-rich Gtk+ user interface.

Host Discovery

As usual, Linux doesn't come up short of network scanners, and many are free as in speech. Of these, Nmap [1] is perhaps the most ubiquitous. Nmap stands for Network Mapper, and it is (naturally) a command-line tool (actually, a set of tools). For those of us not looking for hacker brownie points, a GUI called Zenmap (Figure 1) is also available.

Nmap should be already in your package manager, so you don't need to compile it from the sources. Many of the operations it performs require raw sockets or are otherwise privileged, so you typically run `nmap` via `sudo`.

Imagine you connect to some IP network and want to know which hosts are online. What should you do? What springs to mind first is to ping each host in the subnet in turn and look for replies. This won't work well for large /8 networks (16M hosts each), but for a typical /24 (256 hosts) or smaller, it's a matter of minutes or seconds. This can be done faster if you ping multiple hosts in parallel. This technique is known as a ping scan, and of course Nmap implements it for you:

```
sudo nmap -sn 192.168.0.0/24
```

Figure 2 shows the result. Here, we scan a complete subnet, but Nmap understands many target specifications. It could also be a single host for which you provide either an IP address or DNS name. Although not very useful for host discovery, this target is very common in port scanning, which we will cover next. You can also use IP address ranges: 192.168.0.35-40 or 192.168.0.1,2. Moreover, it is possible to exclude certain targets with `--exclude`:

```
sudo nmap -sn 192.168.0.0/24 ➥
  --exclude 192.168.0.1-10
```

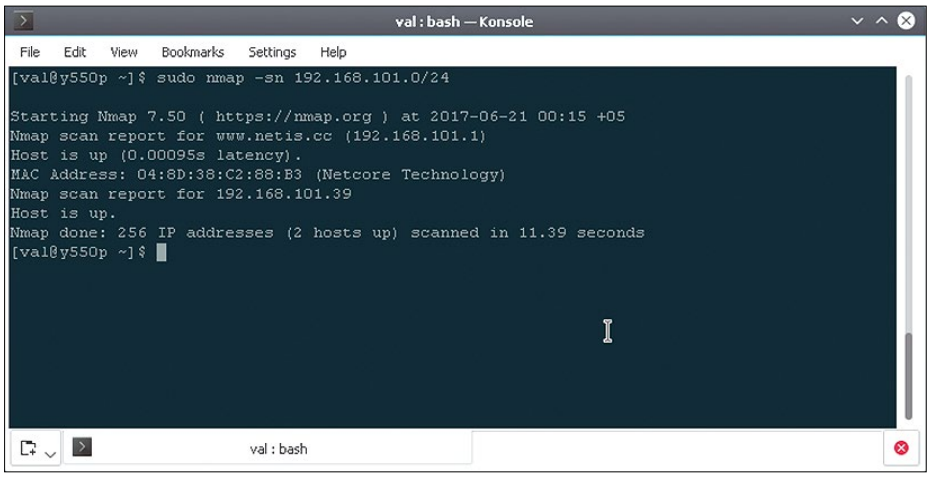


Figure 2: Ping scan results. This kind of output isn't particularly eye-catching, yet it is featured in quite a few movies, including **The Matrix: Reloaded**.

scans everything in the subnet except the first 10 hosts.

So far, so good. What's wrong with ping scan? Nothing, actually, except some administrators may block ICMP on their hosts. Blocking is a bad idea as ICMP has more applications than mere pings (aka ICMP Echo), yet it's rather widespread. If you can't know reliably if the host is off or just blocking pings, your next best guess is to use another technique and combine the results.

If the hosts you are interested in are in the local Ethernet segment (i.e., they see the broadcast traffic you send), ARP ping is a good choice. Instead of pinging each of the hosts, you ask them to resolve the corresponding IP into a MAC address. Nobody suppresses ARP in a sane state of the mind, as this renders the host pretty useless on the IP network. This makes an ARP scan quite effective. It's not 100 percent accurate as well (nothing is), but again, combining the results of two scans reduces the error.

```
$ sudo nmap -PR 192.168.0.0/24
```

ICMP and ARP pings are not the only host discovery options. Other techniques exploit the fact that transport-level protocols, such as TCP or UDP, define a specific feedback if a remote party tries to access an open or closed port. Say, accessing a closed UDP port results in an ICMP Port Unreachable message sent to the originator. As with ARP, these messages are essential for the normal operation and unlikely to be blocked or filtered, which makes them promising candidates for host discovery.

Nevertheless, network providers can monitor for and block suspicious activity, such as a large number of connection attempts made from a single source IP address within a short timespan. Nmap mitigates this with the `-T` switch, which can make scan operation less aggressive, thus less suspicious and bandwidth-consuming. Of course, this also means they would run for much longer. `-T` accepts a single argument, which is either a number in the range of zero to five or a self-explaining keyword: `paranoid`, `sneaky`, `polite`, `normal`, `aggressive`, and `insane`. `normal` is the default. You can find all the details regarding TCP and UDP pings, as well as the `-T` option, at [3].

nping: Not Your Grandpa's ping

Everyone knows `ping`. This ubiquitous tool typically sends ICMP Echo messages and is perhaps the number one way to check if the given host is online. But, as we've learned today, there are many other options.

`nping`, which comes with Nmap, incorporates these options in one tool. It can send TCP and UDP probes, ICMP and ARP pings, and can also reveal intermediate hops working as a `traceroute` substitute. Moreover, it provides many options to tweak just about every bit in a protocol header, from Ethernet to TCP. This makes `nping` not merely a diagnostic tool, but a powerful packet generator you can use for fuzzing, stress testing, and other purposes.

This is how you do a TCP probe for port 80 (HTTP):

```
sudo nping -c 1 --tcp -p 80 scanme.nmap.org
```

`-c` tells `nping` to send one probe only. The tool reports various packet details, such as TCP flags and sequence numbers.

Note that sending TCP probes requires root privileges.

Port Scanning

Now that you know which hosts are online, what should you do next? Perhaps you want to know which services these hosts run – or at least expose to the rest of the world. And this is not a mere curiosity: If you run a service you intend to be internal (such as a database), you must be sure it's not visible from the outside.

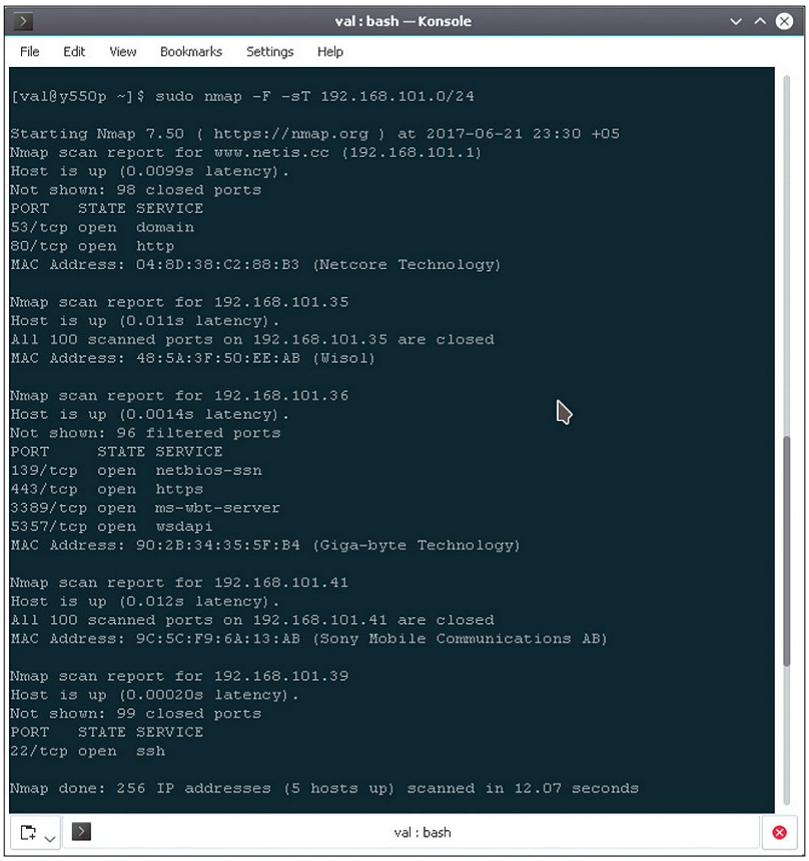


Figure 3: Port scan results. This could be lengthy for a busy network with many hosts running services.

You obtain this information with a technique called port scanning. There are a handful of ways to scan ports, but perhaps the simplest one (and the one requiring no root privileges) is to do a `connect()` to the port in question. This is no different from what an ordinary client application such as a web browser would do. Nmap calls this a TCP connect scan:

```
sudo nmap -F -sT 192.168.0.1
```

The result may look similar to Figure 3, where I scan my home network. `-sT` prescribes a TCP connect scan, and `-F` makes it "fast" by scanning fewer ports than Nmap would scan by default. Some ports are open, as they run services my home router provides to the LAN, such as web management interface (80/TCP) and DNS (53/UDP). You can also spot a supposed Windows machine.

Heuristics other than `-F` are available as well; for example, you can scan the top N most popular ports, according to Nmap's database, with `--top-ports`. For example, the top five ports in my Nmap installation include 21-23 (FTP), 80 (HTTP), and 443 (HTTPS). This works well for common services and quick checks, but for a deeper understanding, you'd want more control over port ranges. Nmap provides it with `-p`: This switch accepts individual ports (`22`) as well as port ranges (`22,222`, `6881-6889`). You can prefix the numbers with `U:` to denote UDP ports or with `T:` for TCP ones.

TCP connect scan is easy yet not particularly fast. To establish a TCP connection, the parties must exchange three messages (SYN, SYN/ACK, ACK) commonly referred as a three-way handshake. This is not required for a port scan, as the very first reply from the remote party indicates whether the port is open or closed. A TCP SYN scan is the faster alternative, which sends only the initial SYN packet. If the remote side responds with SYN/ACK, the port is open. If it sends RST, the port is closed. Anything else, including ICMP error messages, is a clear indication that the port is filtered by a firewall, or the target doesn't run a compliant TCP stack, which is quite rare.

With the following, we can scan the default range of ports:

```
sudo nmap -sS 192.168.0.1
```

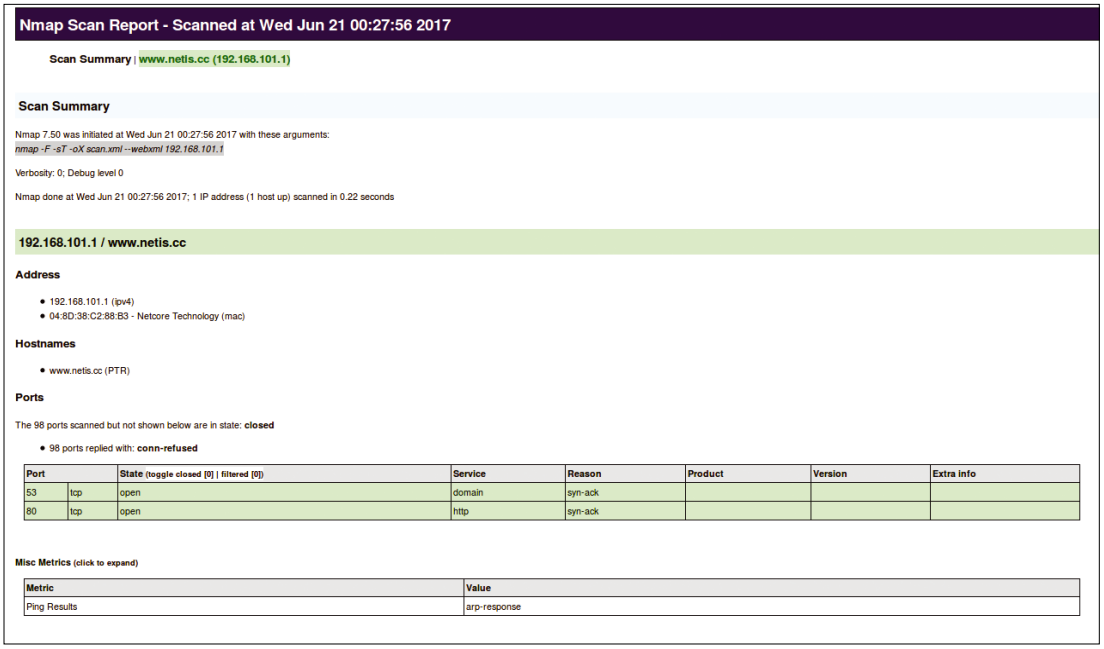


Figure 4: Use XML output for parsing – or producing some good looking reports for a customer demo.

So far, we have seen Nmap output on screenshots in a human-readable format. While this format is most common, it's not the only one available.

If you intend to parse the results with some code, `-oX` dumps the data into XML. It may seem very 1990s, compared to JSON, but you can easily reference an XSL style sheet (try `--webxml`) to make it viewable within any modern

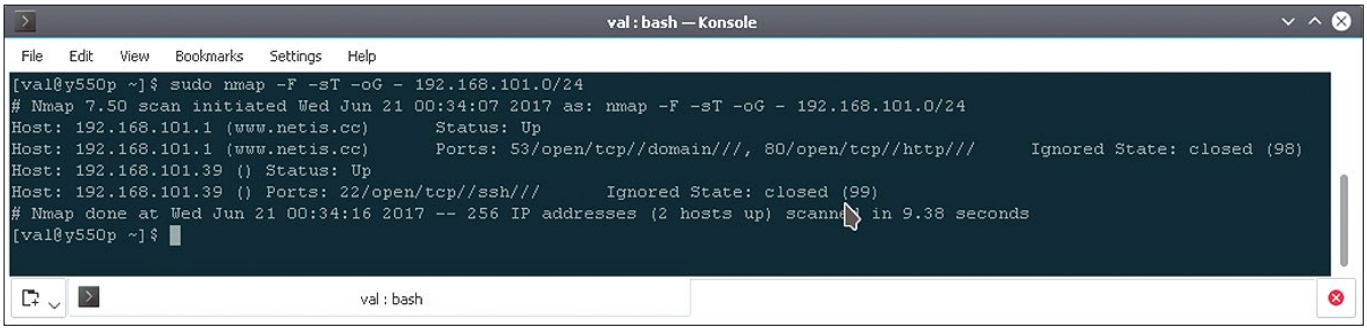


Figure 5: In grepable output mode, each piece of data occupies exactly one (long) line of text.

browser (Figure 4). `-oG` produces "grepable" output (Figure 5), which makes it easier to use Nmap in shell scripts; you can think of it as of something akin to `ip -o`. To pipe Nmap output into `grep`, you can use `-` as a filename.

OS Detection

Knowing that some box on the network runs a web server is only half of the battle. You also want some details, such as whether it is an Apache HTTPD or Nginx, as well as which version. The same stands for the underlying OS.

Why should you be interested in things like this? There are several reasons. Consider a penetration testing scenario. You are trying to break into a system (with the owner's permission, of course) to see how difficult it is and how much time it takes. Knowing which software runs on the server is necessary to look up a vulnerability database for well-known exploits. And, you could be tuning a system to misrepresent itself to an external attacker. In this case, you could run Nmap against it to test if your measures have yielded the desired effect.

There are also two ways to enable these features in Nmap. First, there is one big lever to turn on advanced and aggressive Nmap options (`-A`). The word "aggressive" doesn't refer to the aggressive timing template we've discussed along with `-T` switch. It stands for OS detection, version scanning, script scanning, and `traceroute`. If you don't want all of this, you can turn on individual features with `-O` (OS detection), `-sV` (version scanning), and other similar options [3].

Let us follow the second route. Take one of the hosts running a common service you've discovered in the preceding examples. You can just scan the whole subnet again as well, and even do it aggressively with `-A`, but it would take somewhat longer.

```
sudo nmap -O -sV 192.168.1.35
```

Nmap also tries to guess the host's uptime. The man page [3] explains it can't be done reliably, so this piece of data is only printed in verbose mode. To enable this mode, you supply Nmap the `-v` command-line option.

For Nmap to detect an OS, the target must have at least one open and one closed port; if that is not the case, the tool would complain. Nmap also reports the device type (so you can tell a PC apart from a smartphone) and a freeform OS description. Most importantly, it tells you a Common Platform Enumeration (CPE) for the OS or service detected. CPE serves as a standardized structured naming convention [4]. Have a look at this:

```
cpe:/o:google:android:2.2
```

Here, `/o` says that the item refers to an OS. `google` is the vendor's name, `android` is a product, and `2.2` is the version.

If your target has both open and closed ports, but Nmap can't detect the OS reliably, you'll be given a URL to submit the OS details (Figure 6). This is also an option if the OS is misidentified. Of course, you should only do this if you know which OS really runs on the host in question. This way, you help to make Nmap better for the whole world (Figure 7).

OS fingerprinting is a complex topic, but if you want to know what happens under the hood, *Nmap Network Scanning* provides a complete chapter on OS detection [5]. The book is the official guide to Nmap, and about half of its contents is freely available on Nmap's website. It's a worthwhile read even if you aren't interested in OS detection.

Scripting Nmap

Now, let's briefly cover one of the most powerful Nmap features: Scripting. Notwithstanding the versatility of Nmap, there is always room for extension. You may want to check for a specific vulnerability or for custom software not widely available and thus not in the Nmap database.

The Nmap Scripting Engine (NSE) provides a way to do this. It facilitates automation of Nmap operation with scripts written in Lua. You can write your own scripts, use scripts written by others, or try any of the few hundred scripts that Nmap itself comes with.

All these scripts fall into several categories [6]. There are "safe" scripts, which shouldn't crash remote services or otherwise interfere with remote systems. There are also "intrusive" scripts, which

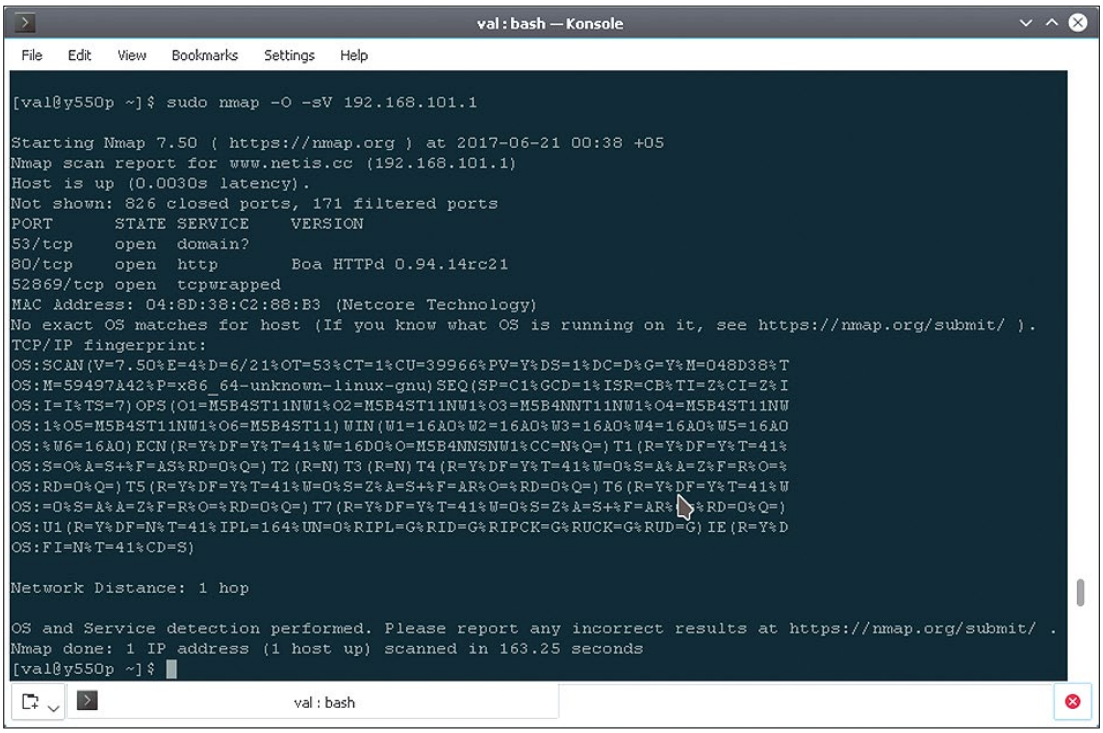


Figure 6: I always thought my home router ran Linux, but Nmap fails to support it. However, note it detected a Boa HTTPD.

you shouldn't run against services you do not want to lose.

Unless a script does version detection only (which makes it fall into the "version" category), it is either safe or intrusive. Scripts looking for a specific vulnerability or exploiting it go into "vuln" or "exploit," respectively.

There is also a special "default" category. It includes scripts that run reasonably fast, are not too intrusive, and provide some value to a general audience. These scripts run when you request Nmap to do a script scan with `-sC`, which is equivalent to `--script=default`, or with `-A` as we saw previously:

```
$ sudo nmap -sC 192.168.0.35
```

Default scripts don't spit out tons of information. The command above should yield a result similar

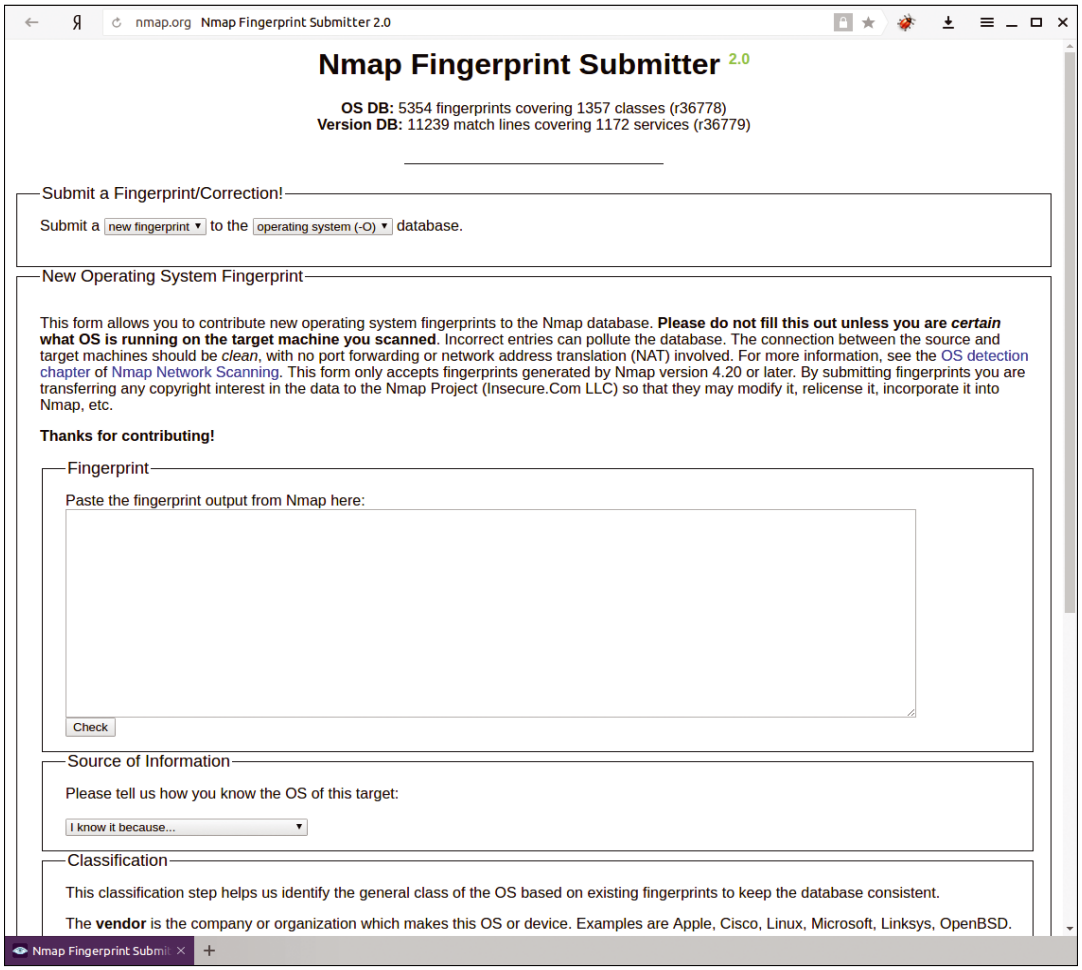


Figure 7: If I knew for sure which Linux runs on my home router box, I'd submit the details here.

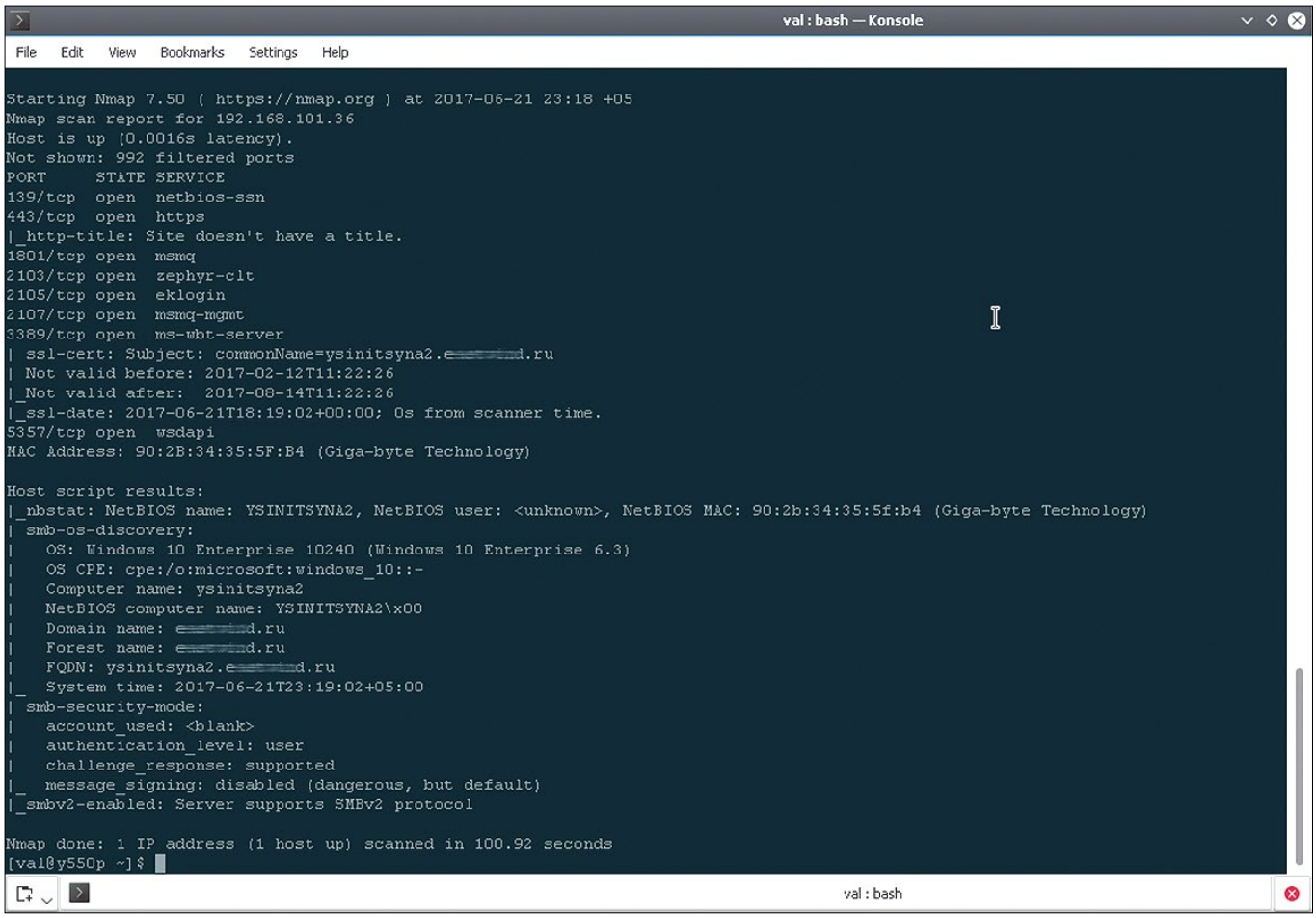


Figure 8: Nmap running a whole arsenal of default scripts against a reasonably secured Windows 10 machine.

to the one shown in Figure 8. Of course, Nmap doesn't run every default script against every host. Typically, scripts are host- or port-bound. For instance, an `http-title` script that displays the main web page title runs only if port 80/TCP is open. Figure 8 also shows some SMB-related scripts (including one doing custom OS detection) that run only against SMB-enabled hosts.

The `--script` switch is actually rather flexible. It accepts a script's name (with or without the `.nse` extension), a directory from which to run `.nse` scripts, a category (such as `default` above), or an expression. Say, `--script "http-*"` runs all scripts whose names begin with `"http-"`. `--script "http-* and not intrusive"` does the same but omits intrusive ones. It is also possible to send scripts their own arguments with `--script-args`. You can find more details at [3]. As a teaser, that's how you can brute force a MySQL server:

```
$ sudo nmap --script=mysql-brute -v $server_ip
```

If the above command yields something under "Valid credentials," you have some bad news for the `$server_ip` administrator.

Nmap is a network security tool, and security is a vast and complex topic. A false sense of security is sometimes worse than no security at all, and the worst thing you can do is to assume you are a security expert when you are actually not. This short introduction to Nmap isn't meant to be a substitute for a proper network security class, nor was it designed to help you break into your neighbors' computer systems. This being said, it is always good to know what's happening in your network, and Nmap is another worthwhile piece of the puzzle here. ■■■

Info

[1] Nmap homepage: *https://nmap.org*

[2] Zenmap homepage: *https://nmap.org/zenmap/*

[3] Nmap man page: *https://nmap.org/book/man.html*

[4] Official CPE Dictionary: *https://nvd.nist.gov/products/cpe*

[5] Lyon, Gordon "Fyodor." *Nmap Network Scanning*, Nmap Project, 2009. Chapter 8, Remote OS Detection: *https://nmap.org/book/osdetect.html*

[6] Lyon, Gordon "Fyodor." *Nmap Network Scanning*, Nmap Project, 2009. Chapter 9, Nmap Scripting Engine: *https://nmap.org/book/nse-usage.html#nse-categories*

FOSSPicks

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

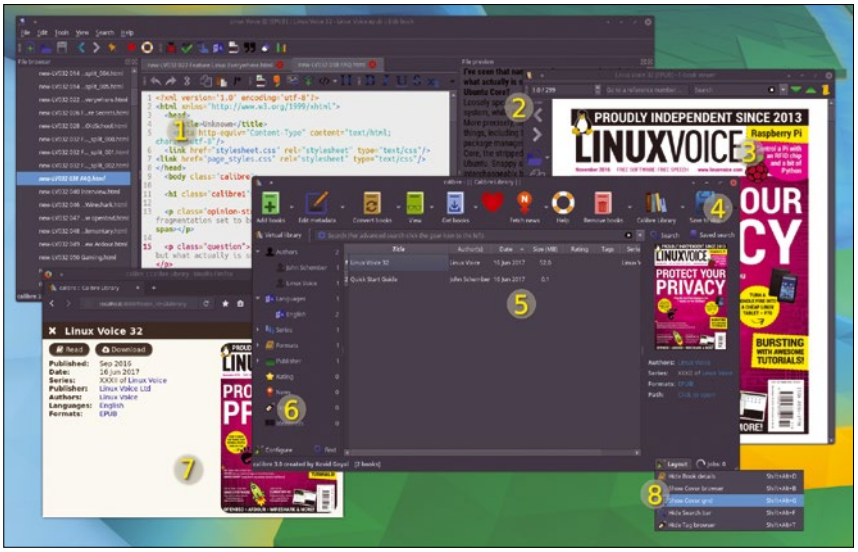
Graham tears himself away from updating Arch Linux to search for the best new free software. BY GRAHAM MORRISON

eBook manager

Calibre 3.0

Linux Voice produced an ePub version of each of its 32 editions, and the software we used to generate them was Calibre. It's complex and a little convoluted, but it's also powerful and crammed full of features. Not only does it enable you to write and publish your own digital publications, it's also brilliant at managing your entire collection and sending items to your devices. Thanks to its plugin system, those devices even include proprietary e-readers, such as Amazon's Kindle. For years, those digital editions of *Linux Voice* were created with Calibre 2.x, which is why the release of 3.0 is such a milestone.

The best thing about the new version is that it includes a new content server. Selecting one of your books and clicking *Connect share* from its context menu starts a new local server, and anyone with a browser on your network can now browse to your server and access your library. These users can browse by collection, see each of your titles, download the files, and even read their content within the browser window. The download option will help when people take their devices offline, as the file stays in local storage. This is brilliant for families, but it could be even more brilliant in an office environment where you want to share your own documentation set or knowledge base with a team, such as reference material for engineers or journals for researchers.



1. Edit books: Edit the content and style sheets to generate ePubs from basic HTML. 2. Preview books: While editing, the preview of the book updates in real time, which is ideal for detecting errors. 3. Read books: The Calibre ebook reader is one of the best for any desktop. 4. Configuration: There are many options, from library sorting and icon themes, to a hugely capable plugin system. 5. Book manager: Create multiple libraries and customize how those libraries appear. 6. Download news: You can also synchronize news sources and push these to your devices. 7. Content server: Calibre can run a local web server for users to access and read content. 8. Layout: Change how your collections appear and are sorted.

The second best thing about this release is its appearance. The icons look dramatically better and more professional, the user interface (UI) has been reorganized, and the UI now works with high DPI displays. Many people in publishing use high DPI screens, so this last feature is going to help with adoption – especially as it was a difficult application to hack into working with high resolutions, unlike many standard Gnome or KDE applications. And, if you need it, Calibre will also output to Microsoft's .docx format.

The installation is a little odd. The download page on the official website provides a long command that essentially downloads and executes a Python installer as root. It works, but I have serious concerns about asking users to run `sudo` on scripts like these that are downloaded within the same command. Of course, you always have a choice, and it's great that the developers provide an alternative to waiting for your distribution to provide packages, but we would recommend waiting for a signed package for your distribution, if you're at all worried. Waiting will definitely be worth the effort. It's taken more than 10 years for Calibre to get to this fantastic state, and although digital publishing has followed an arc of popularity over that time, an application like this is essential for anyone who loves reading.

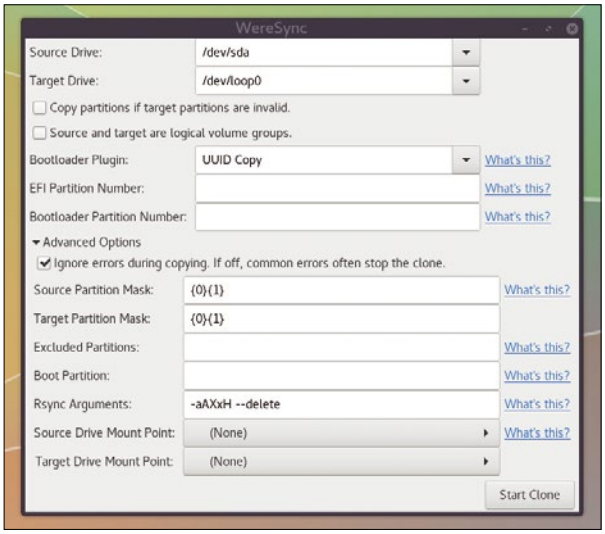
Project Website
https://calibre-ebook.com/

Drive cloning

WereSync 1.0b

Backup is a pain for so many reasons, but the main one is that you have to think about it at all. This is a problem that Apple has almost solved with its own Time Machine/Time Capsule software and hardware combination. This autonomously backs up files as a background operation, seldom bothering the user unless disaster strikes or they need an older version of something restored. Linux can do this, too, of course, especially with the latest generation of versioning filesystems, but it always requires thought and preparation – even if it's `rsync` running on a cron job. WereSync is a new backup solution that attempts to do something similar, although rather more ambitiously; it's designed to clone an entire drive, while you're using it, onto another drive. The second drive then becomes an identical copy that can be booted just like the first, if needed.

The unique selling point for WereSync is that you don't need to be an expert to use it, unlike `rsync` or the cloning Russian roulette of `dd`. This is thanks to its GUI, which at its minimum will take just a couple of arguments and happily go off and duplicate your data. The new drive will need to be big enough to store your data, but new UUIDs will be generated regardless, which means you can have both drives on the same system. It's basically a subset of options provided by `rsync`, with `rsync` running in the background. This is a good choice because `rsync` is probably the most popular backup tool we have, happily copying data and change deltas across drives and networks. It's well-tested, especially at scale, and works. Building atop of this backup stalwart is a good strategy.



If you've ever wanted to use rsync but didn't know which options to choose, WereSync is your friend.

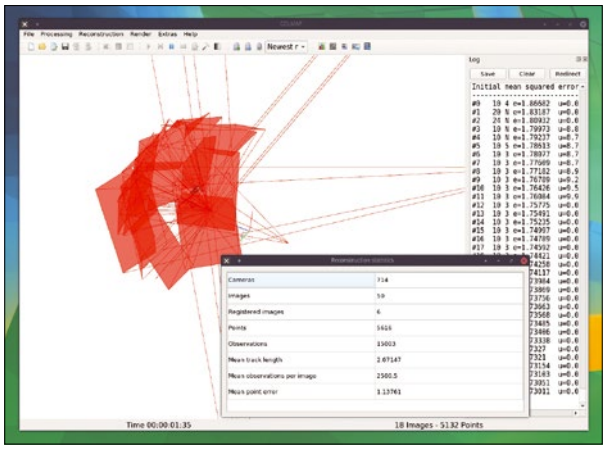
Project Website
https://github.com/DonyorM/weresync

Structure-from-motion generator

COLMAP 3.1

Despite a widespread consumer launch, virtual reality (VR) is still in the developmental phase. This is mainly because of the cost and intrusiveness of this first generation of hardware. There's little doubt, however, that at some point in the not too distant future, VR headsets that track movement and map that movement into a 3D interactive world will transform the way we view the world.

One element of this future will be the recreation of real locations. VR will let us virtually visit and walk around famous landmarks, your childhood home, and historical places. And, the technology that drives these recreations is called photogrammetry. One of the best photogrammetry solutions for Linux is the combination of VisualSFM and MeshLab. But VisualSFM isn't quite open source and is often difficult to install. Now there's another viable option, COLMAP, which performs many of VisualSFM's functions and is potentially capable of better results. COLMAP is both a command-line and GUI-driven application that takes images as input and generates 3D structures as output. Unfortunately, because this is the cutting edge, the process isn't that simple. It starts with taking the images themselves, importing them into COLMAP, and calculating the camera and lens variables before mapping the various photographs into 3D space. Only then can a sparse set of points be generated from the multiple viewpoints of the same structure, which in turn leads to a dense cloud of points, which leads to polygons and finally, a 3D model with optionally generated textures. COLMAP does most of this through a series of tables and requesters, with a 3D viewport to help make sure everything is working as it should.



In many ways, photogrammetry often feels like an ongoing research paper, but its 3D techniques are getting easier to use.

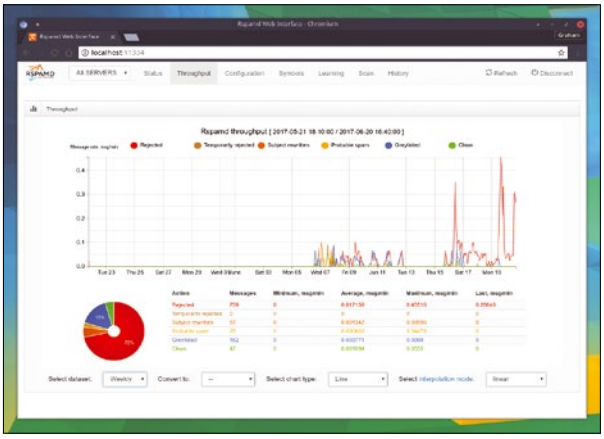
Project Website
https://github.com/colmap/colmap

Spam filtering

Rspamd 1.6.0

I covered an earlier version of this brilliant competitor to SpamAssassin before, but it's worth revisiting for two reasons. The first is that the pace of its development has been phenomenal, with the release of 1.6.0 being the pinnacle of 12 months of continued development. The second is that more people need to hear about Rspamd because, although SpamAssassin is brilliant and doing a wonderful job for thousands of people, it's always good to have a viable alternative. And, Rspamd is just that. It's a spam zapper that cuts out the cruft without taxing your CPU.

Updates over the past year have made huge improvements to its spam identification. For me, it correctly bounces around 95 percent of all the spam I receive, and that's probably 10 percent better than when I was running SpamAssassin. Rspamd can even use your SA filters, and it has the same Bayesian learning capabilities, which means spam you mark as junk is quickly assimilated by the recognition algorithms. It's also a lot less resource intensive than SpamAssassin and lives quite happily on a cheap low-end box hosted somewhere in the back of beyond. Version 1.6.0 made some big changes in the back end, mostly by removing the custom delivery agent, but the upgrade wasn't difficult. If you're already running a standard Postfix mail server, then installing Rspamd 1.6.0 is going to be easier than installing earlier versions, as you can miss out on a couple of important steps, but good configuration still takes time and patience – just like with SpamAssassin. Mostly, configuration files are migrated without your having to touch anything, and you get a quick and powerful spam filter for your own email. Combine this with something like Sieve on IMAP and your spam filtering can be almost as automatic and as good as Google's, only without the huge infringement on your privacy.



The web interface offers a fascinating insight into what spam is hitting your system and how it's dealt with.

Project Website

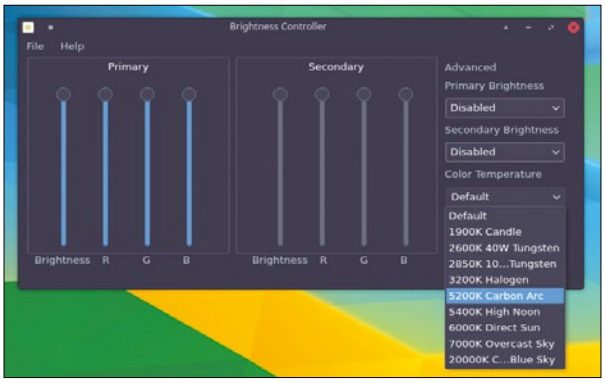
https://rspamd.com/

Screen control

Brightness Controller

Like many people who use a screen for long periods of time, I've found using the wonderful Redshift makes a marked difference in both reducing eye strain and with the mental adjustments necessary when day turns into night. In case you've not seen it, Redshift changes the color temperature of your screen to match your surroundings, and in particular, reduces the amount of blue/white light projected from your screen as dusk begins. It's the kind of feature you can now find elsewhere, from your iPhone and Android devices to Mac OS, but Redshift brought the features to open source users long before it became cool.

Brightness Controller gives you more fine-grained control over the color balance of your screen, without linking those changes to your environment. Thanks to X11's `xrandr`, you can quickly change the color temperature of the output from a comprehensive list of presets, including candlelight and 20000K Clear Blue Sky. Sliders also let you control the amount of red, green, and blue manually, alongside a brightness slider to adjust the overall brightness. What's even more impressive is that if you have more than one monitor, you can adjust their brightness levels separately, with up to four displays being supported in the latest beta version. You can even save and load these values as a settings file. The quality of the brightness adjustment matches that of a monitor's direct controls and yet gives much finer granularity to the overall levels. Changes of 10 or even 5 percent are never adequate when you're working in the dark, and a utility like this is a great alternative to playing with both Redshift and your monitor's menu interface.



Forget your monitor's terrible user interface and use Brightness Controller instead.

Project Website

https://github.com/lordamit/Brightness

Secure web browser

Tor Browser 7.0

Firefox is a wonderful web browser, but it's currently going through something of a transition. This is because it's partly on its way to implementing sandboxed tabs for each site you visit. This is a badly needed upgrade because many of us experience memory and performance issues with Firefox, and we all need it to remain a strong browser to keep the Internet as open as possible. But, it also means some things are currently broken. So this is a good time to look at alternatives, and you could do much worse than using the Tor browser, at least more often.

The reason why Firefox is important is because it's used as the browser foundation for vital projects like the Tor Browser, which has just hit version 7.0. The Tor Browser wraps the stable, long-term support version of Firefox (currently version 52) with the seriously capable anonymity of Tor. This means anyone can get potentially private Internet access without knowing or configuring any of the wider Tor options, which can get complex quickly. Simply download an executable of the browser, verify you have the same binary provided by Tor, and run in-place, or even better, off a live-booted USB stick. You get all the stable features of Firefox with a hopefully untrackable Internet browsing history. And what's even better about this release is that it's already using Firefox's imminent multiprocessing and sandboxing, making it faster and more secure. Tor disables

A simple slider lets you balance security versus convenience for complete control over your privacy.

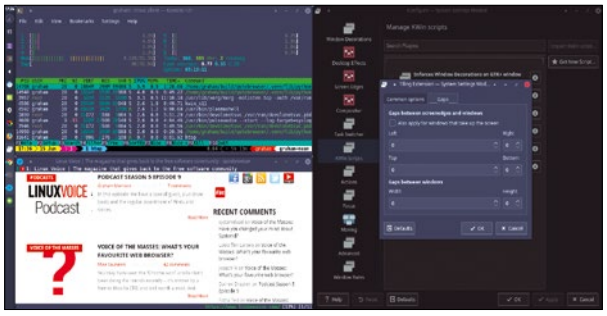
WebGL2, Web Audio, Social, SpeechSynthesis, and Touch APIs so that it's much harder for sites to generate a fingerprint for you and use this to track you across pages and domains. It hardly feels like you're making a compromise, yet your Internet access is much harder to track than with Firefox alone.

Project Website
https://www.torproject.org/

Tiling window management

KDE Tiling Extension

This is a slightly different type of product for this section, as it's not a standalone application or utility. Instead, it's a script for KDE users that makes KDE operate as if it was running a tiling window manager. A tiling window manager, such as xmonad, i3, or bspwm, takes full control of where windows are placed on the screen. Starting with a single application taking over the entire screen, new windows will split the full screen display according to which layout has been selected. It may be purely columns, for example, with each new application subdividing the screen into vertical slices for each window. But layouts can also be more complex. The user can drag the borders between the window panes and define new layouts too, perhaps splitting both vertically and horizontally, depending on your own needs. Tiling window managers are very useful and gaining in popularity, but using them often means you also lose access to many of the convenience features that make Gnome and KDE popular, for example. KDE's Tiling Extension is one possible solution. It's a set of third-party scripts that can either be downloaded by clicking *Get New Scripts* in the KWin Scripts configuration panel, or directly from the project's GitHub page. When activated, the script takes over control of window placement and size, making KDE feel very much like it's running a tiling window manager. Hotkeys allow you to change the layout, and the



Tiling can be disabled or enabled per desktop, application, and window type, for the best of all worlds.

mouse can still be used to resize and move windows from one slot to another (try holding the Alt key and click drag). Turn off tiling for specific types of windows, such as dialogs, or applications, such as Yakuake, and disable tiling completely for separate virtual desktops.

Project Website
https://github.com/faho/kwin-tiling

Modular music composer

BEAST 0.11

Even though this is a shiny new release and the first for over 12 months, running BEAST feels like stepping back in time. This is partly thanks to its ancient Gtk+ user interface, which features large icons, dithered shading, and primary colors. But it's also because this is a music composition tool that uses an old-fashioned analogy to help you create music. The analogy is the idea that simple modules can be connected together with virtual wires to create complicated structures. The process was pioneered, through necessity, by the pre-Moog synthesizers of the '50s and early '60s, where engineers (e.g., Delia Derbyshire) would squeeze every ounce of potential from hardware by rewiring individual modules for specific requirements. The Alsa Modular Synth takes a similar approach in software, although its modules are more traditional and aligned with those of classic synthesis. BEAST, on the other hand, combines many of these modules with a myriad of other views, so that you can make music using a variety of different methods and sound sources in the same application.

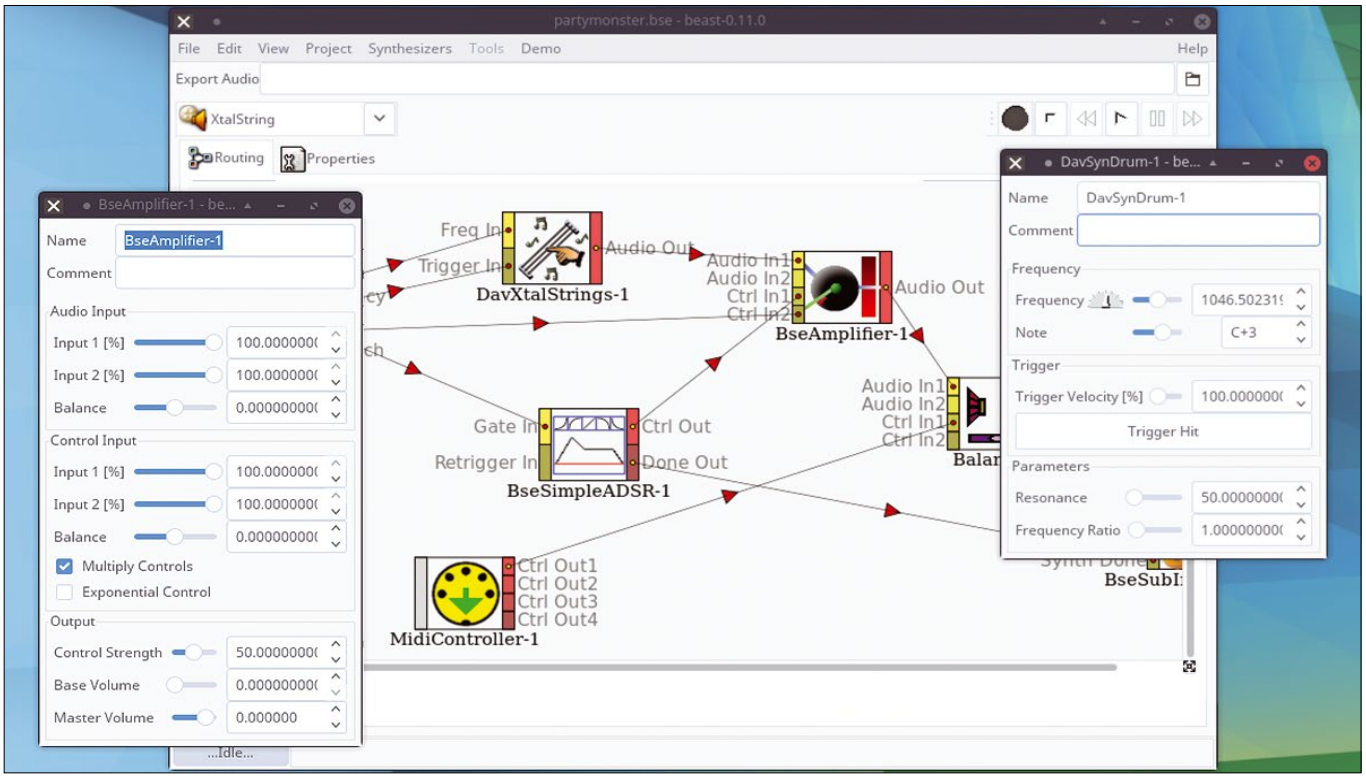
The modular component is most visible in the routing view, where you can edit how sounds are put together and controlled. There are many different modules to try, each with their own collection of inputs and outputs. Input can be taken from audio and MIDI sources, and LADSPA effects can be used just like internal modules. There's a variety of oscillators, modulation sources, envelopes, and effects for creating both normal sounds and crazy outlandish effects. You can then control this sound using the note editor, which works just like Matrix in a more standard sequencer. Click on a point to add a note. These are then painted into the track view so you can change their playback order and the overall construction of your music. A mixer view lets you control the levels and add further effects, and everything can be exported as a single audio file.

BEAST is an excellent piece of software and perfect for older hardware or low-resolution screens, as so much can be done without launching anything else. It's also very inventive, as you can change so much about how sounds are generated and put together. You can use the routing view to create notes as well as sounds, for example, which is excellent if you're trying to recreate the kind of algorithmic music of Steve Reich, or just experimenting with different kinds of input hardware. There's even an built-in audio editor, so you don't need to launch Audacity if you don't want it.

This major update adds SoundFont support, which is probably the most common sample bank format for computers. The DSP engine is also multithreaded, which is ideal if you have a penchant for costly reverberation, and the GUI is being rebuilt using HTML and JavaScript, which should help BEAST herald in a new era. There are now many different kinds of music and audio software for Linux, all with their own strengths and weaknesses, but if there was an award for the music software with the broadest feature set, BEAST would win it.

Project Website

https://testbit.eu/wiki/Beast_Home



Modular synthesizers are ridiculously expensive. Fortunately, open source modular synths like BEAST make them accessible to everyone.

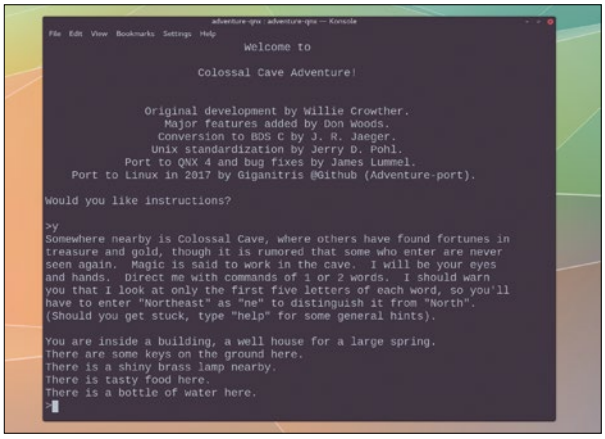
Old school gaming

Open Adventure

Almost 40 years after their invention, text adventure games, or interactive fiction, are still alive and flourishing. There are popular competitions each year to find the best works, and employees from the companies like Magnetic Scrolls are bringing their old games to back to life.

Who would have thought you'd be able to buy a Linux version of The Pawn so many years after its release? Some of this will be nostalgia, of course. Many of us learned to type while trying to work out whether rubbing the lamp did anything, although very few of us have learned to navigate thanks to the maze of twisting little passages. But, because those early machines lacked any real graphical prowess, the text descriptions in interactive fiction seemed infinitely superior in our imaginations. That's perhaps why there is more to interactive fiction than pure nostalgia, and perhaps why the scene is still thriving.

One of the very first of these games was Colossal Cave Adventure, written by William Crowther and Don Woods. Rather surprisingly, the game has never properly been available under an open source license. This has more to do with many people treating it as public domain than for any other reason. But, considering its significance, it seems right that the game gets a proper release. And, that's just what's going to happen, thanks to no other than Eric S Raymond. Raymond has placed a port of the last version (v. 2.5 from 1995) ported for modern environments onto his GitHub page, with the approval of both of the original authors. Raymond has called the original "an important historical artifact," and it's certainly a milestone in computer evolution. If you've never played it, there's never been a better time.



"No text adventure that followed it would be innovative to quite the same degree."-- ESR

Project Website
https://gitlab.com/esr/open-adventure.git

Roguelike

Dungeon Crawl Stone Soup 0.20

Our gaming section seems to be dripping in nostalgia this month, with text adventures and the release of a major update to this popular Rogue-like adventure. Rogue-style games are another genre that started when the only graphics you could squeeze out of your hardware was made up of the ASCII text characters built into your terminal. And because storage was usually limited, too, the only way to create a huge gaming environment was to develop a way of generating levels and maps randomly. This is what Rogue and its progeny do so well, and why the game is still playable today. After decades of playability tuning, the random layout of each level as you traverse from one dungeon down to the next, combined with the random encounters and the way you power up your character, makes for some compulsive gaming.

Dungeon Crawl Stone Soup keeps tightly to this old concept, including the lack of any save unless you quit the game and a million different ways to die. It has huge number of characters and attributes to choose from, and a set of quick challenges if you don't feel up to spending infinity descending a set of dungeons. The graphics fit the game perfectly, being a step up from text but remaining cool enough to be seen in an episode of *Mr. Robot*. This new update adds a Barachim species, upgrades for ogres, two new spells, and new items, such as scarves! And, there are new dungeons to explore, made slightly easier by the noise meter showing your relative noisiness as you creep around the corridors. You don't even need to install the game if you don't want to because multiple public servers let you play the game online, where you can even watch other players and match your woeful score against those who have been playing for decades.

One of the best features of Crawl is that you can watch other players exploring live through a browser – it's like Twitch for the 1990s!

Project Website
https://crawl.develz.org/

MADDOG'S DOGHOUSE

How you approach a problem goes a long way toward success in code development. BY JON "MADDOG" HALL

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®.

Agile Minds

I am writing this article from Japan, where I am spending the weekend with a former student who I taught 30 years ago. This student was a "late bloomer," as he is only 10 years younger than I am, and we have kept in touch over these many years.

My friend does not do much coding any more, and compared with some people, he has not written that much code in his life. For that matter, I never wrote as much code as some people in the Free Software arena. I tell people that my "C" looks a lot like Fortran, but what I write usually works, and works well.

Both of us have moved on to the job of empowering other people to get their work done on time and "under budget." The difference is that when I say "use Free Software and collaboration to help," most of the people I work with do not have to be convinced; in fact, they would look at me strangely if I said anything else. My friend is not so lucky.

So the first night that I was in his house, we broke out several beers, and the time was spent commiserating about how his company kept insisting on doing things "the same way," which typically meant buying technologies from other tech firms and then paying royalties on that software forever, even when the supplier was not adding any value to the solution.

"I do not mind paying for quick bug fixes to problems we have," my friend said, "but we are a relatively small customer to our supplier, and they react to all of their larger customers first, and we get the support last. In addition, there are some products, which we purchase, that we actually have created solutions in-house, which we could substitute and stop paying useless royalties, but our management does not let us do the work."

My friend accepted the agile method of developing code many years ago. In fact, before people became "agile," most Unix people had already developed the habit of creating a quick, throwaway prototype – with the intention of rewriting the code completely before creating the production copy – as something that the end user could work with and see if the basic premise of the code was being met.

What my friend's company was doing, however, was "waterfall" programming, with relatively short periods of requirements gathering, design, coding, and testing, with little or no time for feedback at any level. His fellow workers pointed out that according to the schedule, they would get the job done on time, but my friend pointed out that if they were missing key functionality, they would not find out until the project was due, and then it would be too late.

Many years ago it was very difficult to install the Berkeley Software Distribution (BSD) on a system like a VAX 11/780. You had to boot the media, configure the kernel using a text editor, and type in names of hardware controllers, such as the KMC-11, as well as input the controller's interrupt vector and address in the memory bus. After you saved the configuration file, you rebuilt the kernel, rebooted the system, and proceeded with the rest of the installation.

I noticed that the VMS system could configure itself, even on the same hardware, and I asked why Unix could not do this. "Cannot be done" was the reply I was given.

I went back to my cubicle and created a small prototype using a diagnostic program, some shell tools, etc., and I had a working copy in less than half a day. The engineer who said that it "cannot be done" then took my prototype and created production code in another day.

Another issue facing my friend was the caliber of employees that were sent to him. My friend takes the attitude (as do I) that good engineers can learn a new computer language or IDE in a very short time, but a great engineer will keep learning and improving over their lifetime. The person who says "I do not know that language, but I can learn it quickly" or "I have not heard of that technology, but I can find out about it" are the types of people that my friend and I want to hire.

My friend has a question he asks of potential employees: "What do you say to me if I tell you that we need something done quickly?" His favorite answer was the programmer who said "The first thing I will tell you is that I have to call my wife and tell her I will be home late tonight." That candidate was hired.

Sometimes the answer is not the code, but the way you approach the problem. ■■■

Highly Parallel Programming with Apache Spark

BY BEN EVERARD

Churn through lots of data with cluster computing on Apache's Spark platform.

As a society, we're creating more data than ever before. We're monitoring everything from the planet's weather to the performance of our computers, and we're storing all this information. But how do you process all this data? On a single machine, you can get a few terabytes of disk space and a few hundred gigabytes of memory (at least, you can if your pockets are deep enough), but how do you churn through a petabyte of raw ones and zeros? Basically, you're going to need more than one computer, and you're going to look for a method of running your programs on many machines at the same time: Apache Spark [1].

Before you run off and buy a rack of servers, slow down! We're going to start by introducing Spark on a single machine. Once you've mastered the basics, you can scale up.

Spark is a data processing engine that is often used with Hadoop for managing large amounts of data in a highly distributed manner. If you move forward with Spark, you're probably going to end up with a complete Hadoop setup; however, that's also getting ahead of ourselves. We can start Spark as a standalone service on a single computer.

The first thing you need is Java v1.8. On Ubuntu and derivatives, enter:

```
apt install openjdk-8-jre
```

On other systems, take a look at your package manager for details.

The second thing you need is Spark itself. You can download Spark from the Apache website [2]. I used the latest version at the time of writing (2.1.1 Pre-build for Hadoop 2.7). Download this version and extract the downloaded file.

Open a terminal and navigate to the folder that you've just extracted. From there, you should be able to run the following command:

```
bin/pyspark
```

This command will start Spark and drop you into a PySpark shell. Spark isn't just a programming environment, it's a way of scheduling jobs across a cluster of machines. This is true even in this case when we're just running on a cluster of one machine. There's a web interface to this server on port 4040, so point your web browser to *localhost:4040* to see a list of what's running (Figure 1). At the moment, it won't show much, but this interface is useful for keeping an eye on your Spark server as you have more machines and jobs.

Spark isn't specific to Python. In fact, Scala is the default language, but there are tools for many languages, and Python is my language of preference. PySpark is a tool for submitting Python jobs to the Spark cluster.

Take a look at a simple PySpark program:

```
import re
words = sc.textFile("/usr/share/dict/words")
def check_regex(word):
      return re.match('^[bean]?
                        [bean][bean][bean]$',word)

out = words.filter(check_regex)
out.take(5)
```

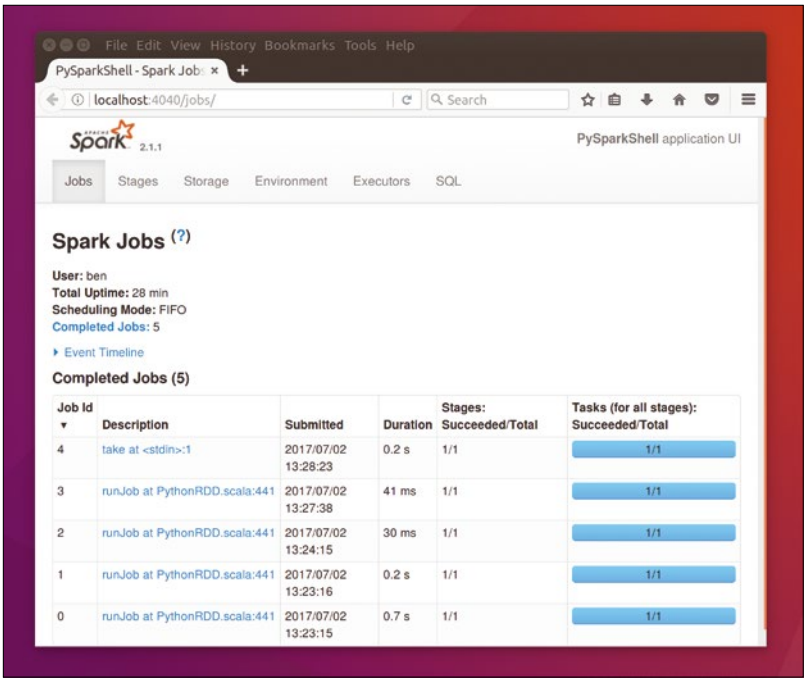


Figure 1: The Spark web interface shows you what's running on your cluster.

```
Terminal File Edit View Search Terminal Help
17/07/02 13:57:14 WARN ObjectStore: Failed to get database global_temp, returnin
g NoSuchObjectException
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.1.1
      /_/

Using Python version 2.7.12 (default, Nov 19 2016 06:48:10)
SparkSession available as 'spark'.
>>> import re
>>> words = sc.textFile("/usr/share/dict/words")
>>> def check_regex(word):
...     return re.match('^[bean][bean][bean][bean]$',word)
...
>>> out = words.filter(check_regex)
>>> out.take(5)
[u'babe', u'bane', u'bean', u'been']
>>>
>>>
>>> words_df = words.map(lambda x: (x, )).toDF(['word'])
>>> out = words_df.filter("word like '%ing'")
>>> out.take(5)
```

Figure 2: A simple Spark program looks for four-letter words that match the pattern.

Resilient Distributed Datasets (RDDs) are the core concept of Spark. They're basically data structures that are stored across all the machines in the cluster so that any operation can be easily parallelized across all the machines. Each RDD is resilient because it can't change. Any operation on an RDD creates a new one while leaving the old one intact.

Our really simple code here takes the `words` file from your machine (if it's not at this location, you can download a `words` file from the Linux Voice site [3]), points your program to the downloaded file), and builds an RDD, with each item in the RDD being created from a line in the file. Essentially, what we have now is an array containing one entry for each word in the English language.

RDDs aren't regular arrays, though. They have methods that allow you to work on them. In this example, I will use the `filter` method, which takes a function as its argument, and this function is run on every item in the RDD to create a new RDD. I will also use the `take` method, which gets us a sample from the RDD; in this case, the argument `5` means I want five items from the RDD.

The `sc` referenced in the program is `SparkContext`, which serves as an entry point for Spark functionality. If you're trying to use Spark from outside the PySpark shell, you'll need to import `SparkContext` and set it up. Take a look at the documentation for details of how to set up `SparkContext`, as it's a bit different depending on how you're running Spark.

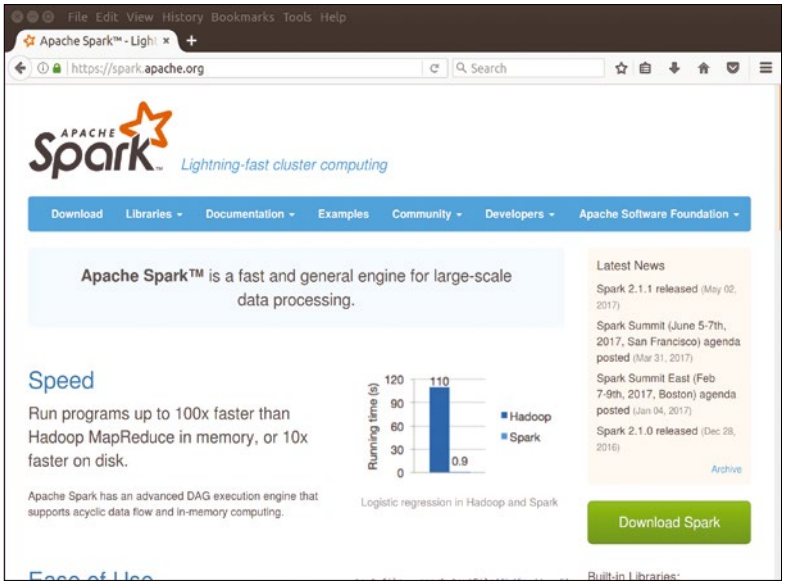


Figure 3: The Apache Spark website has lots of useful information to help you get up and running quickly.

This simple program returns up to five words containing just the letters b, e, a, and n (Figure 2). It is, admittedly, not a particularly impressive program, but it gets us started with Spark.

Another core Spark concept is DataFrames. DataFrames are very similar to RDDs except for the fact that they have a schema. In the previous example, each

entry in the RDD has a single element, a word, but this doesn't have to be the case. RDDs and DataFrames often contain complex sets of data, and setting them against a schema allows you to make more structured queries.

A *schema* is basically a table that we want the data to fit into. As you'll see soon, we have to define a name for the columns in the table as a Python list (in this case, the list `['word']`).

Take a look at the following:

```
words_df = words.map(lambda
 x: (x, )).toDF(['word'])
out = words_df.filter("word like '%ing'")
out.take(5)
```

This code follows on from the previous code block and needs to run in the same PySpark session. The first line does two things; first, it uses the `map` method of the RDD to wrap the words up as tuples. This is important because DataFrames need schemas, and the schema can't be just a single value; it has to be a list, tuple, or row. The second part of the first line builds a new DataFrame from this map, and the DataFrame has a schema with a single column called `word`.

Again we use the `filter` method to pull out some data we're interested in. This time, however, we're not passing a function but a string. Database users amongst you will recognize the syntax as the same as the `where` clause in an SQL statement. We're asking for every row where the column word is like `'%ing'`, and the percent sign matches any text in SQL, so this means every row where the word ends in `ing`.

Now you've run a few tasks in Spark, you can go back to your browser at *localhost:4040* and you should see several completed tasks that have run on your Spark cluster.

This has been a very whirlwind tour of the basics of Apache Spark. Hopefully, you now understand what Spark is and how you can program in it. See the Apache Spark website for examples, documentation, and other information on using Spark (Figure 3). The real advantage of Spark is when you're dealing with massive datasets. You can create DataFrames on the fly and query them efficiently across massive clusters of computers. It's not unusual to have Spark clusters with well over a terabyte of RAM, where huge datasets can sit and be processed without ever hitting the disk, leading to really powerful analyses taking place incredibly quickly. ■■■

Info

[1] Apache Spark: *http://spark.apache.org*

[2] Download Spark: *http://spark.apache.org/downloads.html*

[3] Download words file: *http://www.linuxvoice.com/words*

Organize Your Life with CherryTree

Work smarter and faster using this hierarchical note-taking app that's packed with power user features.

BY MIKE SAUNDERS

Think of all the bits of information we work with on a daily basis: notes for work, notes for your personal life, shopping lists, phone numbers, web bookmarks, passwords, code snippets, photos, and more. Yet, we all have different ways of juggling this info. You may have a single text file called `NOTES.txt` on your desktop with everything crammed inside (in which case, you might at least want to add some structure to it using Markdown, as described in the previous issue).

Or, perhaps you're using a web-based tool to arrange your notes, or you've gone hard-core Emacs Org-mode. Although many applications can be coerced into functioning as note and information managers, wouldn't a tool that's completely dedicated to the purpose be better? One that's built from the ground up to structure your notes, runs on Linux – oh, and is free and open source?

That's exactly what CherryTree is: a "hierarchical note taking application." CherryTree has been around for quite a few years now, yet it is only at version 0.38.1. Don't let that low version number mislead you, however; I've used the software on and off over the years and have always found it to be robust and reliable. (Indeed, as is the case with Inkscape, which is still at version 0.92 despite being used professionally for years, I think a bump to 1.0 would be good marketing. But, developers have their own reasons for waiting.)

Anyway, CherryTree is loaded with features, but some aren't so easy to find or work with. In this tutorial, then, I'll guide you through the process of setting it up, using the basics, organizing your notes and exploring its power user features.

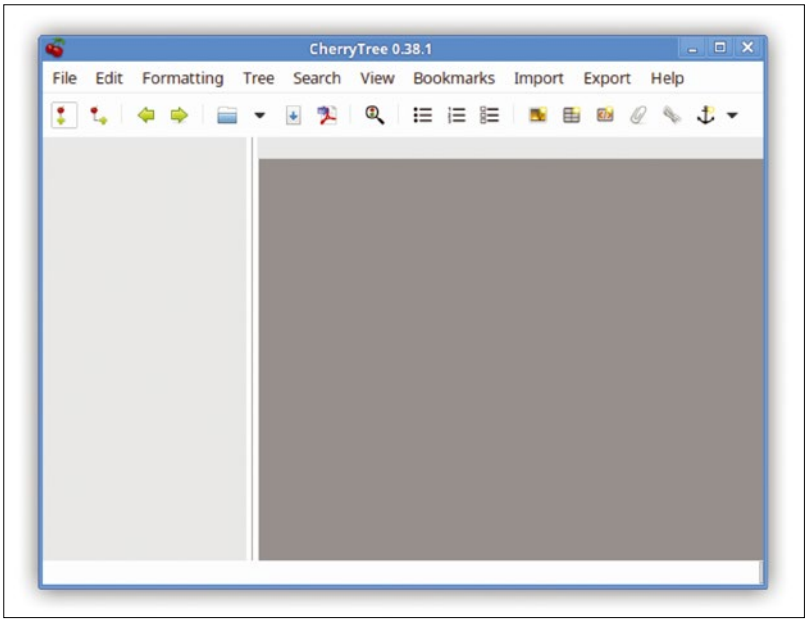


Figure 1: CherryTree isn't very welcoming at first, but this tutorial helps you figure it out.

First Steps

CherryTree is available in most major distros, so just have a peek in your package manager. If you can't find it, you can grab the latest version (with a `.tar.xz` extension) from CherryTree's website [1]. To run it, the dependencies you need to install beforehand are as follows:

- python2
- python-gtk2
- python-gtksourceview2
- p7zip-full
- python-dbus
- python-enchant
- python-chardet

With these installed, extract the `.tar.xz` file you downloaded and run CherryTree with:

```
tar xfv cherrytree-0.38.1.tar.xz
cd cherrytree-0.38.1
./cherrytree
```

Alternatively, if you installed it via your distro's package manager, you can start it from the menu. Either way, you'll see a rather unwelcoming initial screen, as in Figure 1. CherryTree is a rather advanced tool designed to manage heaps of information and shouldn't be oversimplified – but, still, a welcome message and some pointers for the various bits of the interface would be good.

What you will see is the two-pane layout of the app. The left-hand side is used to contain items in your CherryTree file – also known as "nodes" – and the right-hand pane is used for displaying and editing content inside specific nodes. So, let's get

started: Click the leftmost button on the toolbar (if you mouse over it, you'll see that the tooltip says *Add a Node having the same Parent of the Selected Node*).

This will pop up a dialog box, asking you to give the node a name and customize some settings. Call this node *TODO*, make sure that *Rich Text* is selected in the Node Type section, and then click *OK* at the bottom. Now you'll see the interface come to life: The TODO node is available in the list along the left, and on the right you can start typing to add content. Note the status bar along the bottom of the window, which tells you about the type of node you're editing, when it was initially created, and when it was last modified.

The editor pane feels very much like a simple plain text editor, and you'll notice that few buttons in the toolbar are dedicated to modifying the formatting. You'll find bulleted and numbered lists, but that's about it – other formatting buttons are only displayed if you widen the window substantially. Still, a Formatting menu is always displayed, so click in there and have a look around; you'll see the usual bold, italics, and underline options, along with text justification settings and a few headline styles. One particular menu item to note is *Set/Unset To-Do List*, which lets you create lists with checkboxes next to them. You can then click on the checkboxes to tick them – ideal for a TODO list.

You'll also notice that most of the formatting options have keyboard shortcuts listed alongside. CherryTree isn't a newbie-friendly word processor – it's a power tool, designed to be used quickly and efficiently. As such, it's worth memorizing the most important shortcuts so that you can apply formatting without removing your hands from the keyboard.

At this stage, you may not be a big fan of the default color scheme for rich text nodes (white on dark blue). To change this, click *Edit | Preferences* in the menu, then go to the Rich Text section on the left. Under Theme, choose *Light Background, Dark Text* and click *Close*. Your setup will look like the one in Figure 2.

Pretty as a Picture

As well as text with various formatting options, you can add other things to a node's content. Look on the toolbar, for instance, and you'll see that you can insert images. When you do so, you're prompted to rotate the image to your liking and set its size before it's added to the node. Again, this may seem rather fiddly to work with, but remember that CherryTree is a tool for managing structured data rather than WYSIWYG documents like Scribus.

You can also insert tables into a node, specifying their dimensions and optionally populating them with content from a CSV file. To edit the names of columns, move them around, and delete them, just click on the title. If you want to add source code snippets to a node, click the *Insert a CodeBox* button in the toolbar. You can then set up the box's size, and apply syntax highlighting for many different languages. This makes CherryTree especially useful for coding (or sys admin) tasks, where you want to mix up notes and chunks of code.

A few other toolbar buttons are worth knowing about. *Insert File* doesn't actually put the contents of a file inside the node but, instead, provides a link to the file; double-click it, and it'll open using whatever program is assigned for that file type in your desktop environment. Similarly, you can insert links to websites, files, folders, and other nodes in your CherryTree setup, giving the links names, just as you would do in an HTML document. To turn a link into normal plain text, right-click on it and choose *Dismiss* from the menu.

Try experimenting with combinations of images, tables, code boxes, and links to see how they can be combined. To delete images, tables, and code boxes, just click to the right of them, which will display a large cursor – then hit the backspace key.

As your nodes become longer and more detailed, you may want to add some navigation options for them. A handy way to do this is to create a table of contents – or more precisely, get CherryTree to do all the hard work for you. Try adding a few headings to your node, via the formatting menu (choose the *h1*, *h2*, or *h3* options). Scatter them around in your node and then go to the top. *Click Edit | Insert TOC*, and you'll see that a bunch of links are inserted into the node. These links contain the text you used for the headings; click the links to jump to the corresponding location in the node.

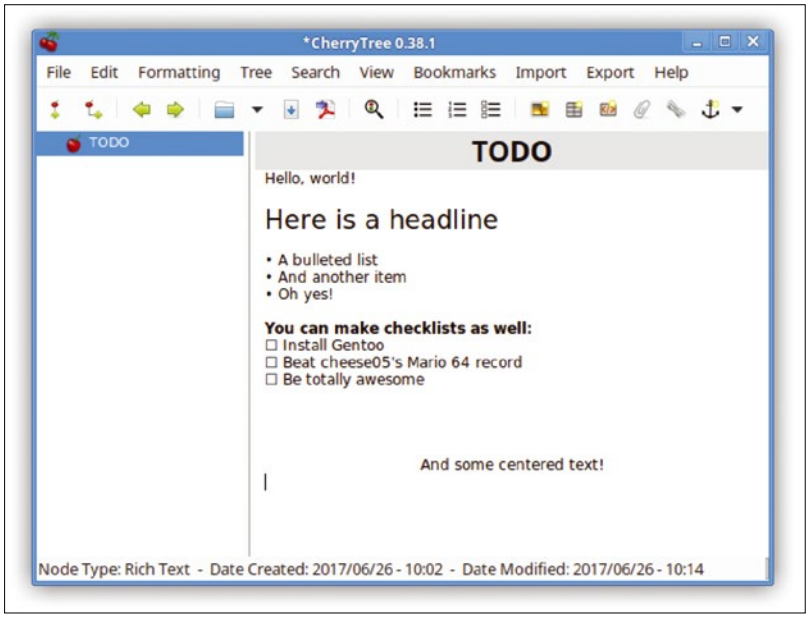


Figure 2: Rich text editing is available in nodes for adding formatting, bulleted lists, and checkboxes (ideal for to-do lists).

The Edit menu contains a few other items that you may find useful, such as the ability to insert a timestamp (i.e., a string containing the current date and time), special characters, and horizontal rules to break up content. If you've pasted in some code from a website or terminal and it has lots of annoying trailing spaces (i.e., spaces lingering on the ends of lines), then the *Strip Trailing Spaces* option becomes a huge time saver. You can even enable a spelling checker via the Edit menu, if you need it.

Juggling Nodes

So far, I've focused on editing and managing content inside a single node. You've see that CherryTree is already a pretty good app for this job, but its real strengths can be found in its node system. At the moment, you have just one node, so click *Tree | Add Node* from the menu, and you'll see that a second node is available to select in the panel on the left. This is an empty node, so click it and you can begin editing content on the right. You can now freely switch between nodes – and if you add even more, you can use the forward and backward buttons on the toolbar to move between them, much as in a web browser.

With this in mind, you can create separate nodes for different types of information: personal to-do list, work items, interesting source code snippets, and so forth. But, as mentioned previously, CherryTree is a hierarchical note-taking application. How does that come into play here? Try selecting one of your nodes in the left-hand panel, then go to *Tree | Add SubNode* in the menu. Give the node a name and click *OK*.

You'll see that the new node has now been created as a child of the one you selected – in other words, it's part of a higher-level node. Using subnodes, you can start to structure your information much more effectively, adding multiple levels of subnodes where it makes sense. Look at Figure 3, for instance: There, I've created a general Notes node, where I can put scraps of information I need for daily life.

Underneath that, I've created a Coding subnode – so this is also for notes, but on programming-related topic. There, I can add general tricks and tips that are not about any specific programming language. Underneath that, I've added yet another subnode, this time for Python snippets that may come in useful in my programming jobs.

You can see how useful this approach is when you have lots of information to manage. CherryTree has some helpful extras for managing nodes, such as *Tree | Insert Today's Node* from the menu. This sets up a new node for the current year, then a subnode for the current month and a subnode of that for the current day. You can use this to keep a diary, for example, or just make notes that are relevant to a specific day.

Using drag and drop, you can move nodes and subnodes around inside the left-hand panel;

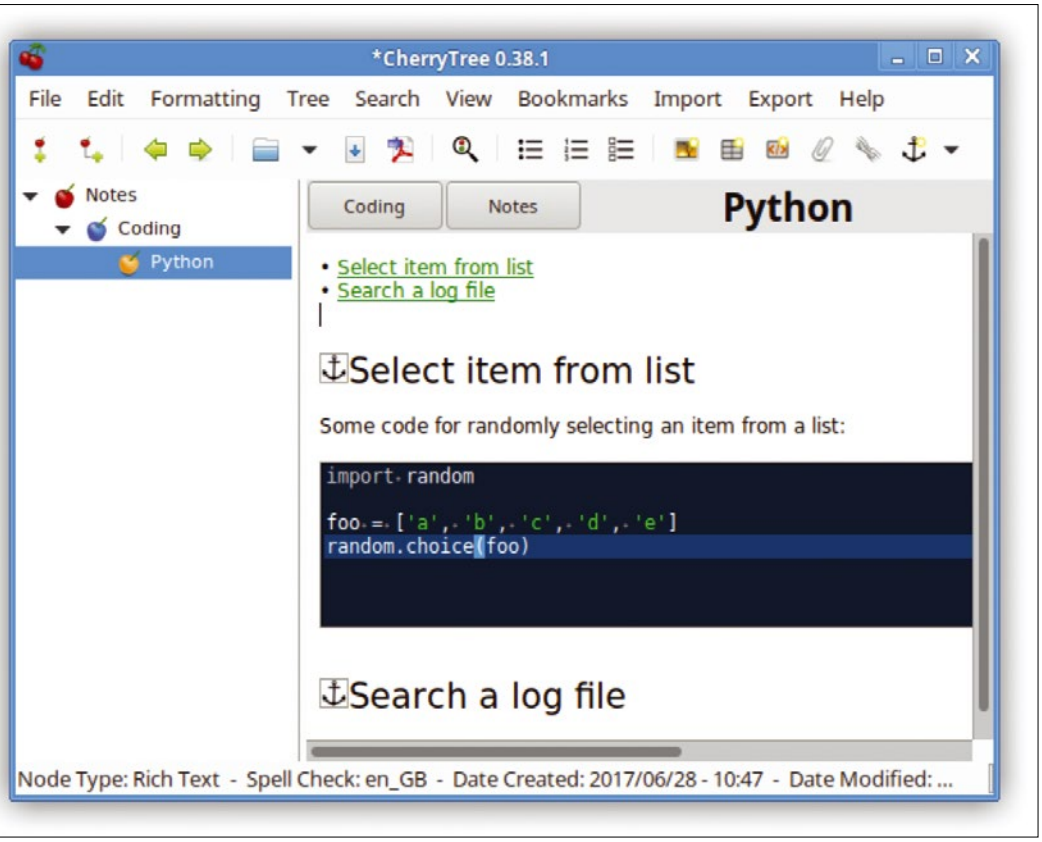


Figure 3: Here, I've created a couple of sub-nodes, that I'm using to keep track of my coding projects.

you can take one child node and make it a child of another node, for example. When doing this, be careful, because the undo/redo option doesn't work in this case. A context menu also pops up when you right-click on a node; to duplicate it, change its properties or move it around in the panel.

Searching and Linking

Now, how can you create a link from one node to another? In the Edit menu, click *Insert/Edit Link*. You'll be prompted to provide a name, and then in the following dialog box, click *To Node* and then the relevant node. It's also possible to link to specific parts of a node, known as anchors. To try this, click *Edit | Insert Anchor* in the menu and then enter a name. This places an anchor symbol inside the node, where the cursor currently is, but it doesn't add any text. When you now go to *Edit | Insert/Edit Link* and chose *To Node*, you'll see that you can optionally choose an anchor inside the node you select (if one exists).

After you've been using CherryTree for a while and have built up lots of information, you may find it hard to pinpoint specific bits of info (although a well thought-out structure helps here). Under the Search menu you'll find two especially useful menu items: *Find in Node Content* and *Find in All Nodes Content*. As the names suggest, the first performs a search only on the currently selected node, while the latter searches through all nodes. You can perform especially complicated searches using regular expressions, and search forward and backward. Note that you can perform find and replace operations via the Search menu also – again, either in a single node or across all nodes.

So that's CherryTree. I've explored all of its main features, and you can see just how useful it is compared with plain text files or web-based note-taking apps. You can find out more in the "Saving, Importing, and Exporting" box and in the app's manual (Figure 5). If you end up using CherryTree to organize your whole life, drop us a line and let us know! ■■■

Saving, Importing, and Exporting

When saving your nodes via *File | Save*, you're given a few choices: SQLite or XML format, with optional password protection. Note that CherryTree extracts password-protected documents into a temporary directory when you're editing them, so if the app crashes, it may leave that temporary directory open for viewing. Ultimately, this means CherryTree is fine for storing non-supercritical passwords (e.g., web forum logins) on single-user machines, but for really serious passwords (like online banking) a dedicated password manager is arguably a better choice.

If you take a look at the Import menu (Figure 4), you'll see that CherryTree can suck in data from an impressive range of sources, including many other note-taking apps and services, such as NoteCase, Gnote, and Zim. The variety of export formats is smaller, but you can save your work as PDF or HTML. It's also possible to dump all nodes into separate plain text files or output everything to a single file. You can even select a bunch of nodes and create a separate CherryTree document from them via *Export | Export* to CherryTree Document.

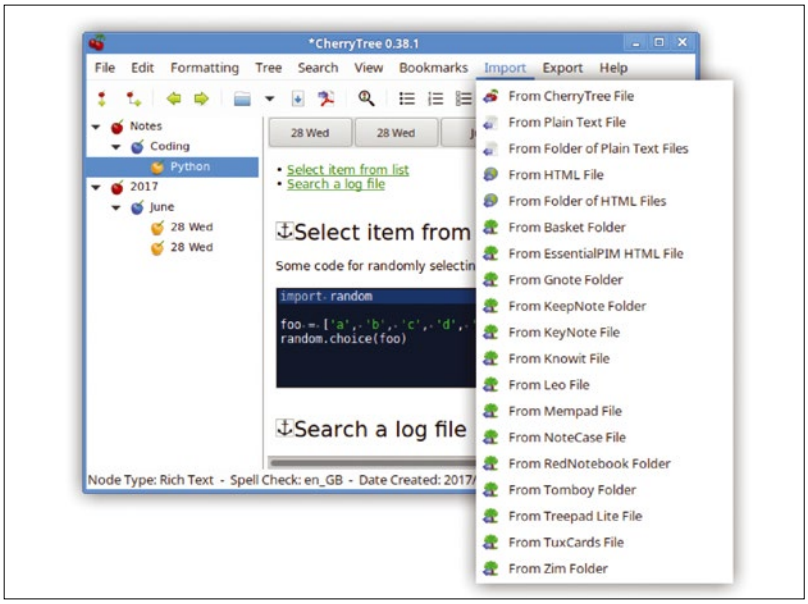


Figure 4: CherryTree can import data from a vast range of sources, including popular note-taking apps.

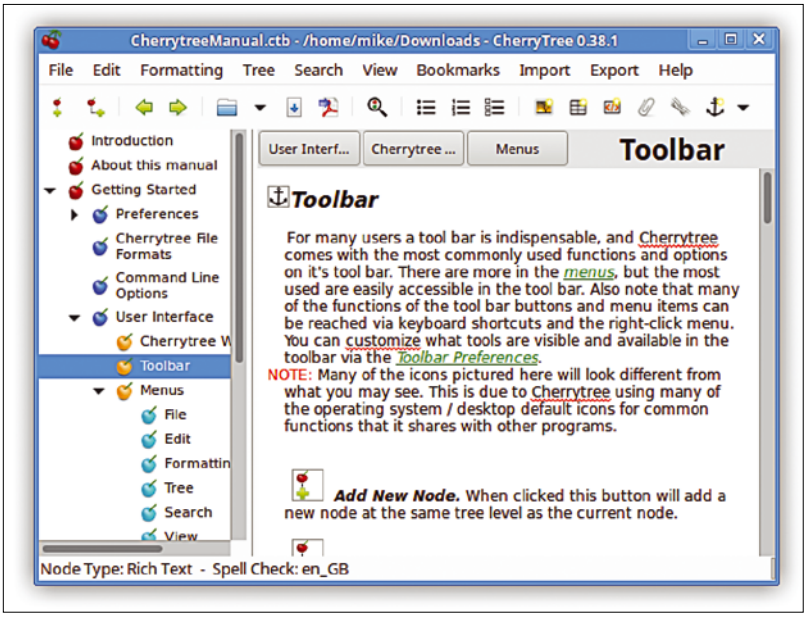


Figure 5: The app's own manual (`CherrytreeManual.ctb`, from the website) is a CherryTree document that demonstrates the power of the program.

Info

[1] CherryTree: *http://www.giuspen.com/cherrytree/*

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*.

SUSECON 2017

Date: September 25–29, 2017

Location: Prague, Czech Republic

Website: *http://www.susecon.com*

SUSECON 2017 features exceptional technical content presented by SUSE employees, customers, partners, and community enthusiasts. SUSECON focuses on helping you find ways to build and define your future to provide a flexible and efficient infrastructure. The dynamic and entertaining keynote speeches inspire, inform, and invigorate.

JAX London

Date: October 9–12, 2017

Location: London, England

Website: *https://jaxlondon.com/*

JAX London is a four-day conference for cutting-edge software engineers and enterprise-level professionals. JAX brings together leading innovators in the fields of Java, microservices, continuous delivery, and DevOps. Tracks include Emerging Technologies, Java Core & Languages, Agile & Communication, Big Data & Machine Learning, and more!

Linux Kernel Summit

Date: October 24–27, 2017

Location: Prague, Czech Republic

Website: *http://events.linuxfoundation.org/events/linux-kernel-summit*

The annual Linux Kernel Summit brings together core kernel developers to discuss the state of the existing kernel and plan the next development cycle. New in 2017 are four days of sessions and workshops opened to a larger group of developers, along with the half-day, invitation-only Maintainer Summit.

EVENTS

InterDrone	September 6–8	Las Vegas, Nevada	http://www.interdrone.com/
Bremer IT-Sicherheitstag	September 11-12	Bremen, Germany	https://www.heise-events.de/bremeritst7
WiSTEM 2017	September 11–12	San Francisco, California	http://www.womeninstemconference.com/
Storage Developer Conference (SDC)	September 11–14	Santa Clara, California	http://www.snia.org/events/storage-developer
Open Source Summit North America	September 11-14	Los Angeles, California	http://events.linuxfoundation.org
Kieler Open Source and Linux Conf.	September 15–16	Kiel, Germany	http://kilux.de/
14th Annual 2017 HPC for Wall Street, Cloud, AI & Data Centers	September 18	New York, New York	http://www.flaggmgmt.com/hpc/
OSBConf	September 25-26	Cologne, Germany	http://osbconf.org/
SUSECON 2017	September 25–29	Prague, Czech Republic	http://www.susecon.com/
data2day	September 26–28	Heidelberg, Germany	https://www.data2day.de/
CopyCamp 2017	September 27–28	Warschau, Poland	https://copycamp.pl/en/
Internet Security Days	September 28–29	Brühl, Germany	https://isd.eco.de/
JAX London	October 9–12	London, England	https://jaxlondon.com/
it-sa	October 10–12	Nürnberg, German	https://www.it-sa.de/
Heise Cloud Conference	October 17	Cologne, Germany	https://www.heise-events.de/cloudkonf
All Systems Go!	October 21–22	Berlin, Germany	https://all-systems-go.io/
All Things Open	October 23–24	Raleigh, North Carolina	https://allthingsopen.org/
Open Source Summit Europe	October 23–25	Prague, Czech Republic	http://events.linuxfoundation.org/events/open-source-summit-europe
WebTech Conference	October 23–27	Munich, Germany	https://webtechcon.de/
heise devSec	October 24–26	Heidelberg, Germany	https://www.heise-devsec.de/
EclipseCon Europe	October 24–26	Ludwigsburg, Germany	https://www.eclipsecon.org/europe2017/

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

Managing Editor
Rita L Sooby, rsooby@linux-magazine.com

Localization & Translation
Ian Travis

News Editor
Swapnil Bhartiya

Copy Editors
Amber Ankerholz, Amy Pettle

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen, Lori White

Cover Image
jossdiim and Sergey Kandakov © , 123RF.com

Advertising – North America
Ann Jesse, ajesse@linuxnewmedia.com
phone +1 785 841 8834

Advertising – Europe
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 99 34 11 48

Publisher
Brian Osborn, bosborn@linuxnewmedia.com

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
616 Kentucky St.
Lawrence, KS 66044 USA

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)
Fax: 1-785-856-3084

For all other countries:
Email: subs@linux-magazine.com
Phone: +49 89 99 34 11 67

www.linuxpromagazine.com – North America

www.linux-magazine.com – Worldwide

Printed in Germany on recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Distributed by COMAG Specialist, Tavistock Road, West Drayton, Middlesex, UB7 7QE, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 616 Kentucky St., Lawrence, KS, 66044, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 616 Kentucky St., Lawrence, KS 66044, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany.

Approximate
UK / Europe Sep 02
USA / Canada Sep 29
Australia Oct 30
On Sale Date

Issue 203 / October 2017

Home Automation

The home of the future has arrived, and Linux users love rigging up ingenious solutions to small problems around the house. Next month we look at some important tools and technologies that are driving the quest for open source home automation.

Preview Newsletter

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: ***www.linux-magazine.com/newsletter***


THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE
OCTOBER 23 - 25, 2017 | PRAGUE, CZECH REPUBLIC
LinuxCon, ContainerCon, CloudOpen and the new Open Community Conference combine under one umbrella name in 2017 — the Open Source Summit. At Open Source Summit, you will collaborate, share information and learn across a wide variety of topics, with 2,000 technologists and community members.
LINUX PRO READERS SAVE 15% ON ATTENDEE REGISTRATION WITH CODE LNM15.
go.linuxfoundation.org/osseulnm

SUPERMICRO®

Up to 14 Million IOPS

The Fastest NVMe Servers

with the New Intel® Xeon® Scalable Processors

The Leader in Server Technology

- NVMe and 205W CPU Support
- Free Air Cooling supported by most system configurations
- 3 UPI, 2666/2933MHz 24 DIMM support and beyond
- Supermicro Rack Scale Design (SRSD) ready,
- Battery Backup (BBP®),
- 100G/OPA/25G connectivity
- QAT, JBOF Supported

intel XEON PLATINUM inside

Intel Inside®. Powerful Productivity Outside.

SUPERMICRO X11 SERVER PLATFORMS

SUPERMICRO®

Learn more at supermicro.com/X11