



GIT WORKSHOP
Up close with the classic
Linux version control system

.NET CORE
The benefits of dotnet on Linux

LINUX
MAGAZINE

LINUX

PRO

MAGAZINE

NOVEMBER 2018

.NET CORE

Running Microsoft's flagship
framework on Linux

Haiku
Is this little-known OS
ready to compete with
Linux and BSD?

5 Mind-mapping Tools
Visualize decisions and
organize your best ideas



Counting Pennies
Tabulating audience
feedback with a Rasp Pi

NET

Electron Framework
Cross-platform apps with
JavaScript and HTML

Remote Pair Programming
Cool new coding technique
with the emphasis on teamwork

LINUXVOICE

- Tracking Your Finances with Eqonomize
- maddog – Passing the Baton
- Polo File Manager



FOSS Picks

- Thunderbird 60
- Taskbook
- Git Cola

Tutorials

- Reading sensor data
- Retrieving email attachments

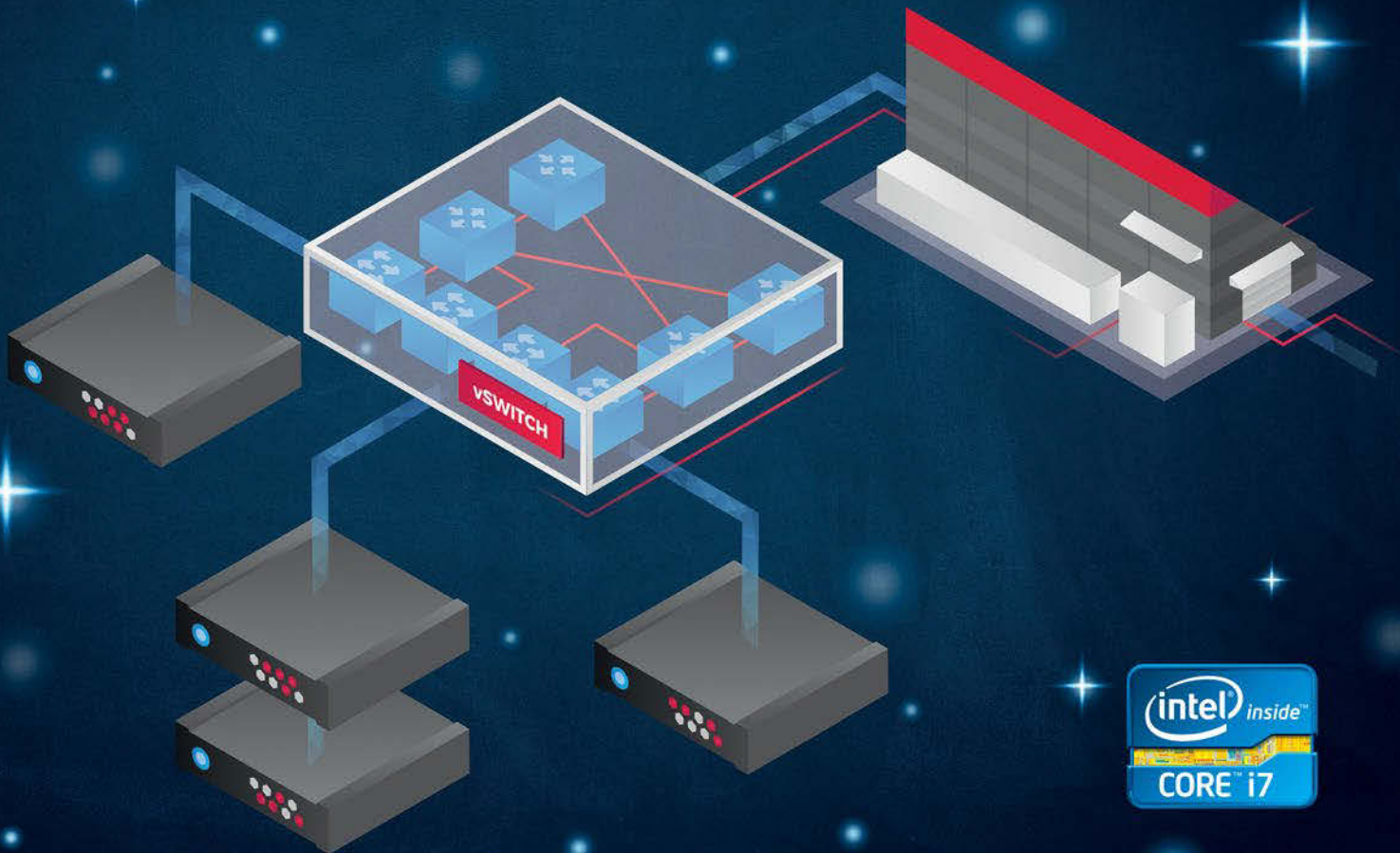
Issue 216
Nov 2018
US\$ 15.99
CAN\$ 17.99

0 74820 58049 3

1 1

vSWITCH IT UP FREE FEATURE

FOR PRIVATE NETWORKS & MORE



e.g. Dedicated Root Server SX61

- ✓ Intel® Core™ i7-3770 Quad-Core
- ✓ 32 GB DDR3 RAM
- ✓ 4 x 6 TB SATA 3 Gb/s 7200 rpm
- ✓ 30 TB traffic inclusive*
- ✓ No minimum contract
- ✓ Setup Fee \$80.50

vSwitch - Build your own virtual networks now

With our new free vSwitch feature, you can connect your Hetzner Online dedicated root servers to each other across our different data center locations via VLAN. So now, for example, you can easily set up a private network via your account on our Robot administration interface.

monthly \$ **80.50**

www.hetzner.com

* 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 \$1.16 per additional TB used.

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. All rights reserved by the respective manufacturers. Intel, Intel Logo, Intel Core and Core Inside are brands of the Intel Corporation in the USA or other countries.

BEING EXCELLENT

Dear Reader,

Try to keep in mind the immortal words of Bill and Ted, "Be excellent to each other." Linux Code of Conflict 2015 [1]

The Linux kernel community is an amazing collection of superhuman programmers who have succeeded beyond anyone's dreams in changing the world. But the kernel list is also known as a pretty rough and tumble place, where maintainers don't mince words and are often known to mince *people with* their words. Past efforts have attempted to dial down the harsh language, but as we learned recently, the issues have lingered.

Few in the Linux community would have missed Linus Torvalds' recent announcement [2] that he was taking some time off to "...get some assistance on how to understand people's emotions and respond appropriately." This dramatic mea culpa from the Linux creator, which appears to have resulted from some form of intervention by top lieutenants, was big news in the Linux blogs and message boards, and it predictably took on an operatic level of dramatic attention, given the grand maintainer's reputation for irascible, and sometimes overly colorful, intensity.

I wish all the best to Linus with his time off, and I commend him for self-awareness, which is currently a scarce commodity in our public life. I feel, though, that the personal story surrounding Linus has tended to overshadow the real story, which is the appearance of a new Code of Conduct [3]. The human story is a fine scoop for the news blogs, but the new Code of Conduct represents a real, lasting change and a positive step to address a persistent problem.

For a bit of context, after a similar intervention three years ago, Linus and the kernel team ratified a rudimentary behavior standard, which was known as the Code of Conflict.

The Code of Conflict was quite brief, and, despite the whimsical quote from Bill and Ted (see above), it had a slightly defensive and legalistic feel to it. The whole first paragraph was not about the rights of contributors but was, instead, a defense of the process, stating "Your code and ideas behind it will be carefully reviewed, often resulting in critique and criticism. The review will almost always require improvements to the code before it can be included in the kernel. Know that this happens because everyone involved wants to see the best possible solution for the overall success of Linux..."

The document did provide a mechanism for contributors to complain if they felt abused or attacked, adding that if "... anyone feels personally abused, threatened, or otherwise uncomfortable due to this process, that is not acceptable. If so, please contact the Linux Foundation's Technical Advisory Board at <tab@lists.linux-foundation.org>, or the individual members, and they will work to resolve the issue to the best of their ability." But the lack of specificity left it largely up to the participants to define what was meant by "be excellent,"

not to mention what was meant by "abused, threatened, or otherwise uncomfortable."

The new document, which is borrowed from an online source known as the Contributor Covenant [4], clears up those questions by enumerating examples of right and wrong behavior:

"Examples of behavior that contributes to creating a positive environment include:

- *Using welcoming and inclusive language*
- *Being respectful of differing viewpoints and experiences*
- *Gracefully accepting constructive criticism*
- *Focusing on what is best for the community*
- *Showing empathy towards other community members*

Examples of unacceptable behavior by participants include:

- *The use of sexualized language or imagery and unwelcome sexual attention or advances*
- *Trolling, insulting/derogatory comments, and personal or political attacks*
- *Public or private harassment*
- *Publishing others' private information, such as a physical or electronic address, without explicit permission*
- *Other conduct which could reasonably be considered inappropriate in a professional setting"*

Perhaps even more significant than these examples is that, rather than starting with a careful "defense" of the process as it was known in the past, the new doc begins with a pledge to the community that is oriented toward creating a better environment for the future:

"In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socioeconomic status, nationality, personal appearance, race, religion, or sexual identity and orientation."

The 2018 Code of Conduct is a significant improvement over the 2015 Code of Conflict and appears to represent a genuine effort to impose some order on the wild world of the kernel developers. Some may miss the fireworks. To be honest, the new Code of Conduct will make working on the kernel more like working for a regular old corporation (most corporations have some similar language in their employee handbooks) and less like living around Mount Olympus.

Mount Olympus is a wonderful place for the people with the lightning bolts, but for everyone else, it often tastes of flash-burned moussaka.



Joe Casad,
Editor in Chief

Info

[1] Code of Conflict: <https://www.kernel.org/doc/html/v4.17/process/code-of-conflict.html>

[2] Linus announcement: <https://lore.kernel.org/lkml/CA+55aFy+Hv9O5citAawS+mVZO+ywCKd9NQ2wxUmGsz9ZJzqgJQ@mail.gmail.com/>

[3] Code of Conduct: <https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=8a104f8b5867c682d994ffa7a74093c54469c11f>

[4] Contributor Covenant: <https://www.contributor-covenant.org/>

LINUX MAGAZINE

WHAT'S INSIDE

Many in the Linux community don't put much trust in Microsoft, but a new guard in Redmond has been changing its ways. This month, we look at .NET Core, a piece of the .NET framework that Microsoft released under an open source license in 2014.

Other highlights this month include:

- **Haiku** – a new beta release of this microkernel system leaves many wondering whether another open source OS has finally arrived (page 22).
- **Remote pair programming** – a pair of free tools could help popularize an agile technique that allows remote coders to work together on the same code (page 44).

At LinuxVoice, we study the Egonomize personal finance app and show you how to create a homegrown program that accesses sensor data on your smart phone (page 70).

SERVICE

- 3 Comment
- 6 DVD
- 96 Featured Events
- 97 Call for Papers
- 98 Preview

NEWS

08 News

- A New Business Model for Open Source Projects
- Linux Mint Debian Edition 3 Released
- Zorin OS 12.4 Released
- Debian Celebrates Its Birthday

10 Kernel News

- Fixing printk() Bit by Bit
- Kernel Internationalization (or Not)
- Kernel Encryption and Secure Boot

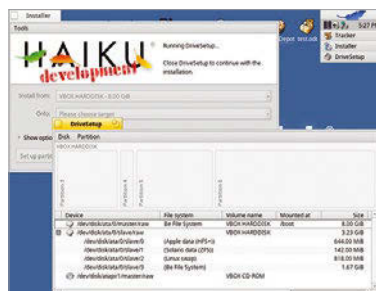
14 HackerOne's Mårten Mickos

CEO of the innovative HackerOne bug-hunting service describes how the company harnesses the power of white-hat hackers around the world.

REVIEWS

22 Haiku

The long-awaited Haiku OS beta release has arrived. This BeOS-inspired operating system may finally be ready for daily use.



COVER STORIES

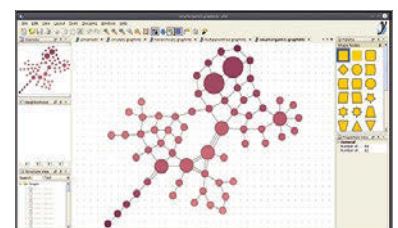
16 .NET Core on Linux

Python enjoys a strong foothold in the Linux space, and Go is quickly becoming a contender. Is it really worth considering an alternative language environment from former Linux-foe Microsoft? On close inspection, the open source .NET Core actually does offer some compelling features that could be worthy of attention.



26 Mind Maps

Mind maps are designed to help display processes and projects clearly in a graphical format. This review explores the design possibilities offered by five mind map programs.



IN-DEPTH

34 Version Control with Git

We'll show you how to get started with the Git version control system, a powerful tool for managing software projects.

40 Command Line – Bulk Renaming

When it comes to renaming multiple files, the command line offers time-saving options in the form of mv, rename, and mmv.

44 Remote Pair Programming

Pair programming saves development costs by putting two coders to work on the same code. Visual Studio Code and tmate bring the promise of pair programming to remote workers.



51 Charly – grepcidr

Often it is the very simple tools that lead to the greatest success. Charly takes an IP address filter and counts the devices in his home, in addition to tripping up spammers.

MAKERSPACE

52 Pi Network Monitoring

Monitor disk space and other parameters with SNMP; view and control Rasp Pi GPIO pins remotely with custom SNMP objects, and create web dashboards with Node-RED.

56 Raspberry Pi Penny Counter

Get feedback for live events with an exit survey that counts pennies.



64 Open Hardware – Challenges

Changes in funding, manufacturing, and technology have helped move open hardware from an idea to reality.



LINUXVOICE

67 Welcome

This month in Linux Voice.

69 Doghouse – Continuity

Developing an exit strategy can ensure continuity for a FOSS project.

70 Economize

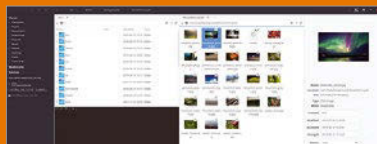
Most accounting programs for Linux are aimed primarily at businesses. Economize focuses on personal use offering a smart solution for getting a handle on your household budget.

74 FOSSPicks

Graham reviews Thunderbird 60, Stress-Terminal UI, Taskbook, SolveSpace, Star Ruler 2, and more!

80 Polo File Manager

If you expect more from a file manager than the ability to move files, Polo might be for you.



85 Tutorials – Cordova Sensor

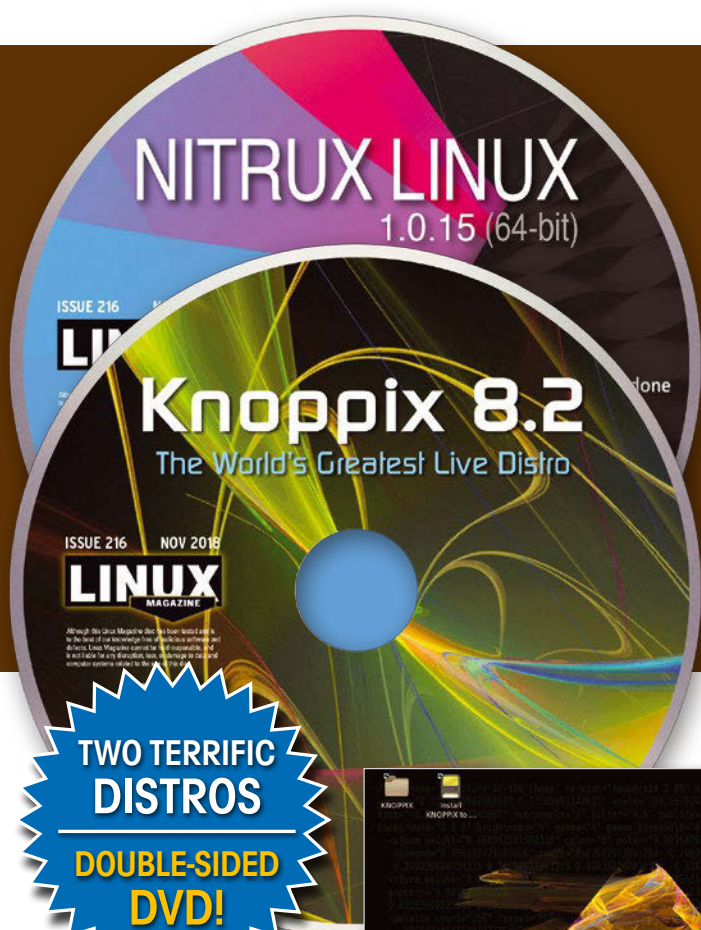
A new universal standard makes it easy to create mobile applications that access your phone's sensor data.



90 Tutorials – Attachment Extraction

Retrieving email attachments manually can be a tedious task. We'll show you a script that fetches attachments automatically and can even save the email as a text file.

On the DVD

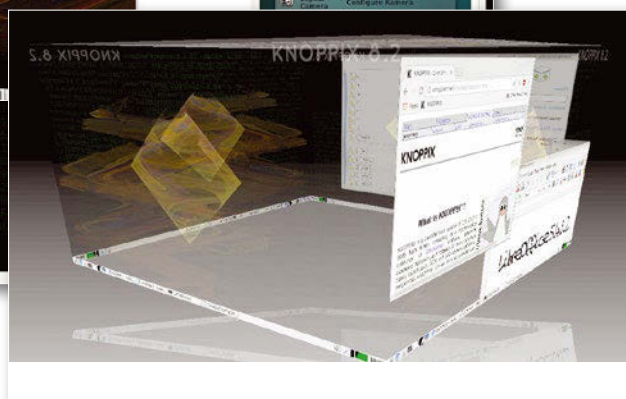
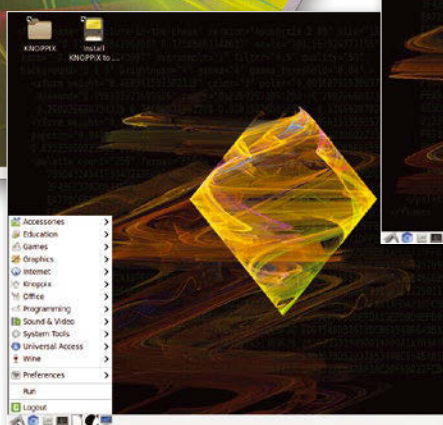


Knoppix 8.2

Knoppix is the ultimate Live distro, with dozens of built-in tools for troubleshooting, monitoring, and resurrecting downed Linux (and Windows) systems. The abundant menus of the Knoppix user interface also include a vast selection of Linux desktop and development tools for a complete admin environment on a single disc.

NitruX 1.0.15

This Ubuntu-based Linux offers a special spin on the KDE Plasma 5 environment. The in-house Nomad desktop, which is based on KDE, provides simplicity and elegance for a unique user experience. The latest version comes with Linux kernel 4.18.5, as well as updates to the graphics stack and support for lots of new hardware.



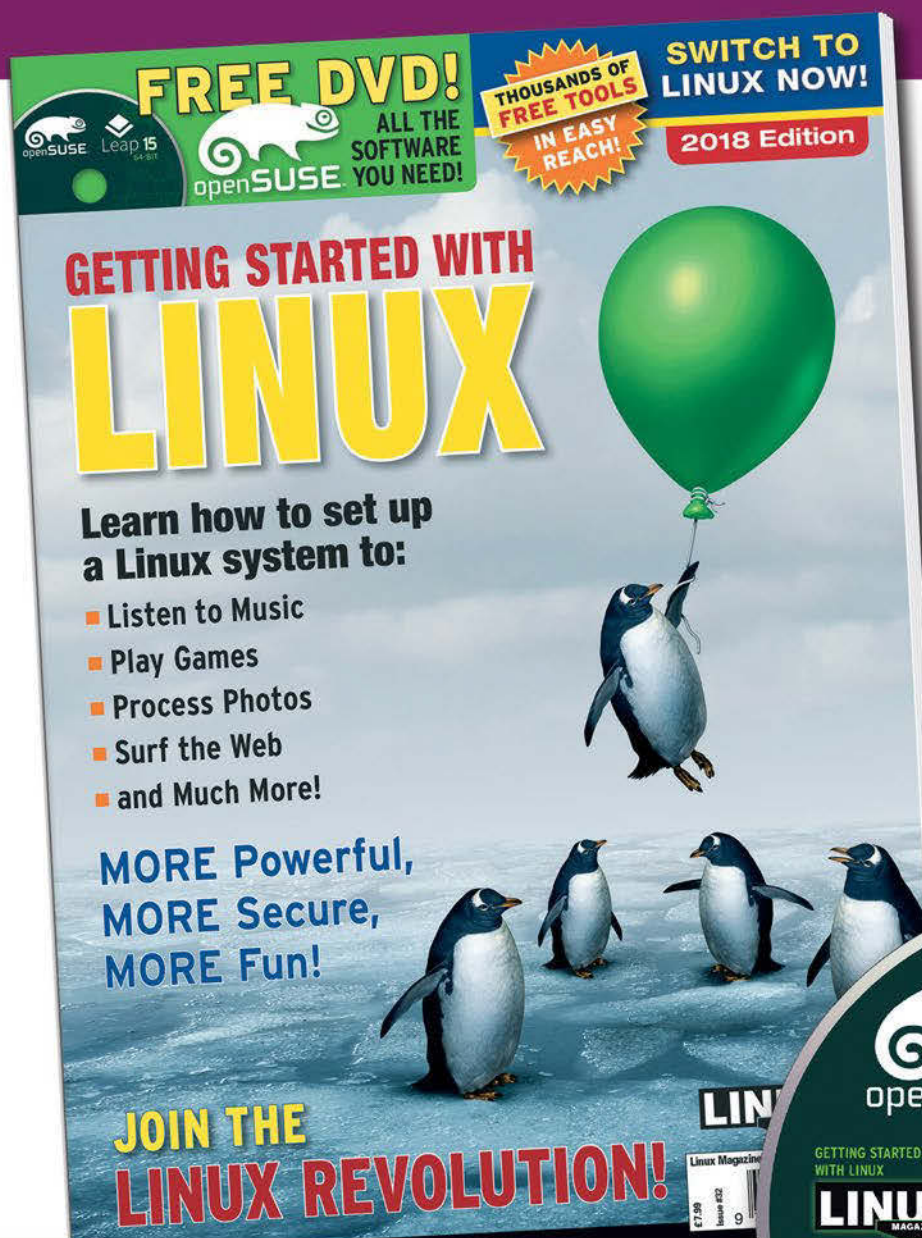
Defective discs will be replaced. Please send an email to subs@linux-magazine.com.

Additional Resources

- [1] Knoppix 8.2: <http://www.knopper.net/knoppix/knoppix820-en.html>
- [2] Contacting Knoppix: <http://www.knopper.net/kontakt/index-en.php>
- [3] NitruX: <https://nxos.org/>
- [4] Nomad Desktop: <https://nxos.org/#nomad-desktop>
- [5] NitruX Help: <https://nxos.org/en/compendium/>

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

This single issue shows beginners how to:



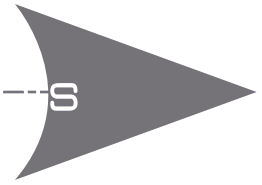
- Install Linux
- Download and install free software for your Linux system
- Create documents and spreadsheets
- Play games
- Surf the web
- Process photos
- Play music and videos
- and much more!

HELP OTHERS JOIN THE LINUX REVOLUTION!

ORDER ONLINE:
shop.linuxnewmedia.com/specials

NEWS

Updates on tech



THIS MONTH'S NEWS

- 08** • A New Business Model for Open Source Projects
- Linux Mint Debian Edition 3 Released
- 09** • Zorin OS 12.4 Released
- Debian Celebrates Its Birthday
- More Online

A New Business Model for Open Source Projects

Storj is a fully open source and decentralized storage solution that brings an Airbnb-like business model to users who have extra storage and bandwidth. At the Open Source Summit North America, Storj announced a new program that extends the revenue generation model to open source projects.

The newly announced Open Source Partner Program enables open source projects to generate revenue every time their users store data in the cloud.

"Our Open Source Partner Program will help open source companies to remain open and free and invest in growth," said Storj CEO Ben Golub.

The program can in fact be a boon for those open source projects that are often constrained by budget.

"It will also enable them to achieve more within their budgets, supporting them in becoming profitable, accelerating roadmaps, or meeting other financial-related goals," Golub added.

Storj tracks usage on the network and returns a significant portion of the revenue earned when data from an open source project is stored on the platform.

Ten new open source players are joining Storj and integrating it with their products. These projects include Confluent, Couchbase, FileZilla, InfluxData, MariaDB, Minio, MongoDB, Nextcloud, Pydio, and Zenko.

On a side note, if you have extra storage and networking bandwidth, you can also join Storj as an individual.

Source: <https://storj.io/#waitlist>

Linux Mint Debian Edition 3 Released

The Linux Mint project has announced the release of Linux Mint Debian Edition (LMDE) 3. What's the need for LMDE, when there is already Ubuntu-based Linux Mint? Isn't it a waste of resources? Not really. LMDE has been created as a backup for Ubuntu-based Linux Mint in the event that Ubuntu ceases to exist.

"Its main goal is for the Linux Mint team to see how viable our distribution would be and how much work would be necessary if Ubuntu was ever to disappear," according to the project.

Since the Linux Mint community controls the UI of Linux Mint (Cinnamon), LMDE creates an experience almost similar to Linux Mint, minus the Ubuntu base. LMDE 3 (Cindy) is based on Debian 9 (Stretch).

In fact, LMDE could be the best distribution to use stock Debian with some fine-tuning. Since Debian is supported for a long time and there are no time-based releases, LMDE follows the same cadence. According to the project, there are no point releases in LMDE.

Those users who are running LMDE 2 can easily upgrade to LMDE 3. LMDE 3 comes in two versions – 32 bit and 64 bit. You can download LMDE 3 from the official page: <https://linuxmint.com/release.php?id=33>



Zorin OS 12.4 Released

The Zorin OS project has announced the release of Zorin OS 12.4. It's based on Ubuntu 16.04 LTS. Zorin OS is known for being one of the most polished Linux-based distributions that's targeted at those who plan to move away from Windows or Mac OS.

According to the project, "Zorin OS 12.4 is the final point release of Zorin OS 12 before the launch of the next major version of Zorin OS. It will be based on Ubuntu 18.04.1 and will be released later this autumn."

In a previous interview, the Zorin OS founder, Artyom Zorin, told me that the goal of the project is to make it extremely easy for a user to use Linux (<https://www.youtube.com/watch?v=7-X1syAGzT8>). Users should not have to choose between convenience and Linux.

Zorin OS has created a very loyal user base. Zorin OS 12 was reportedly downloaded more than 1 million times (<https://zoringroup.com/blog/2017/11/01/one-million-downloads-for-zorin-os-12/>). According to a blog post, "We're also pleased to see that over 60% of these downloads were coming from Windows and Mac OS, reflecting our mission to bring the power of Linux to people who've never had access to it before."

Zorin OS 12.4 comes with the newly-included Linux kernel 4.15, as well as an updated X server graphics stack and compatibility for newer computers and hardware in Zorin OS.

Zorin OS comes in four versions – Core, Lite, Ultimate, and Business. While Core is available for free, other versions are available for a fee and come with extra packages, settings, and official support.

Download Zorin OS: <https://zorinos.com/download/>



Debian Celebrates Its Birthday

The Debian GNU/Linux project celebrated its 25th birthday on August 16, 2018. Debian was created in 1993 by Ian Murdock. The name of the project came from the first three letters of his then girlfriend Debra and his own name – Deb Ian.

In the Debian manifesto, Murdock wrote, "Debian Linux is a brand-new kind of Linux distribution. Rather than being developed by one isolated individual or group, as other distributions of Linux have been developed in the past, Debian is being developed openly in the spirit of Linux and GNU. The primary purpose of the Debian project is to finally create a distribution that lives up to the Linux name. Debian is being carefully and conscientiously put together and will be maintained and supported with similar care."

Debian has evolved to become one of the most popular distributions. Its stable branch dominates the Linux-powered web hosting services. The popularity of Debian also led to an entire generation of Debian-based distributions, including Ubuntu and Knoppix.

Debian has three releases: stable, testing, and unstable. Stable is meant to be used on servers and by users who don't want their systems to change frequently. Stable has packages that are very well tested; as a result, they can be old.

Testing has packages that are not part of stable yet but are in the queue. Most Debian-based distributions, such as Ubuntu, are based on testing. It's also suitable for desktop use on home PCs.

Debian Unstable is the place where all development happens; it's really bleeding edge and is meant only for developers.

Version 9 is the current version of Debian, and its code name is Stretch. Each version of Debian is code-named after a character from the movie *Toy Story*. The unstable branch is code-named Sid, because Sid is the character that breaks everything.

MORE ONLINE

Linux Magazine

www.linux-magazine.com

ADMIN HPC

<http://hpc.admin-magazine.com/>

Shared Storage with NFS and SSHFS

Jeff Layton

HPC systems require shared filesystems to function effectively. Two really good choices for both small and large systems are NFS and SSHFS.

ADMIN Online

<http://www.admin-magazine.com/>

Version 5.2 of the Ruby Framework

Stefan Wintermeyer

Ruby on Rails 5.2 was released during RailsConf, which took place in Pittsburgh in mid-April 2018. Although not much has changed for old Rails applications, you'll find a few notable additions for new ones.

Many Approaches Help Secure a Web Server

Matthias Wübbeling

We submit an Apache web server to the Qualys SSL Server Test and look at how to protect against data theft with a combination of TLS by way of Let's Encrypt, SELinux or AppArmor, a firewall, and restraining your web server's verbosity.

Protecting Documents with Azure Information Protection • Klaus Bierschenk

Azure Information Protection helps businesses control how information in communications between employees is handled.

ADMIN DevOps Focus

<http://www.admin-magazine.com/DevOps>

Security as Code • Chris Binnie

Gauntlt is a sophisticated DevOps tool that can test the security of your continuous integration/continuous delivery pipeline.



Zack's Kernel News



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

By Zack Brown

Fixing printk() Bit by Bit

The `printk()` system call is an important way for the kernel to produce logs and other messages. The kernel doesn't use any standard library functions like `printf()`, so it has to roll its own. But by all accounts, `printk()` is a mess.

Recently, Sergey Senozhatsky tried to spruce it up a little and avoid some potential deadlocks. There was a whole range of deadlocks caused by `printk()` recursing onto itself, and Sergey didn't want to touch any of those. But he said there were plenty of non-recursive deadlock scenarios that needed to be fixed.

Specifically, there were ways to deadlock the system in the output console, and `printk()` would trigger those deadlocks by trying to write to the console. To fix some of these, Sergey wanted to introduce some new helper functions for the TTY (used to implement the console) and UART code (used to communicate asynchronously with the console).

Unfortunately, this would require updating every single serial driver to use the new helper functions. Also, it would only address deadlock issues involving particular types of kernel locks. But Sergey figured a partial fix was at least a beginning.

Alan Cox was not thrilled with Sergey's proposal. He felt that the fixes would add unnecessary code to parts of the kernel that needed to be as fast as possible. Alan felt that "`printk` nowadays is already somewhat unreliable with all the perf related changes," so he favored a simpler approach that would simply defer trying to produce `printk()` output if the lock wasn't available.

Sergey objected that Alan's idea wouldn't address deadlocks that had already been reported by users.

Meanwhile, Peter Zijlstra said that Alan had drastically understated the crappiness of `printk()`. He said, "`printk` is a steaming pile of @#\$#@; unreliable doesn't even begin to cover it."

Petr Mladek, who had also been working with Sergey on this code, explained:

"This patch set adds yet another spin_lock API. It behaves exactly as spin_lock_irqsafe()/spin_unlock_irqrestore() but in addition it sets printk_context.

Where printk_context defines what printk implementation is safe. We basically have four possibilities:

1. Normal (store in logbuf; try to handle consoles)
2. Deferred (store in logbuf; defer consoles)
3. Safe (store in per-CPU buffer, defer everything)
4. safe_nmi (store in another per-CPU buffer; defer everything)

This patchset forces safe context around TTY and UART locks. In fact, the deferred context would be enough to prevent all the mentioned deadlocks."

But Linus Torvalds had strong objections to the whole idea. He said:

"The rule is simple: DO NOT DO THAT THEN.

Don't make recursive locks. Don't make random complexity. Just stop doing the thing that hurts.

There is no valid reason why an UART driver should do a printk() of any sort inside the critical region where the console is locked.

Just remove those printk's; don't add new crazy locking.

If you had a spinlock that deadlocked because it was inside an already spinlocked region, you'd say 'that's buggy'.

This is the exact same issue. We don't work around buggy garbage. We fix the bug – by removing the problematic printk."

In light of this, Steven Rostedt remarked, "Perhaps we should do an audit of the console drivers and remove all `printk`, `pr_*`, `WARN*`, [and] `BUG*` from them."

Sergey objected that this wasn't just a case of removing unwanted `printk()`s from the code. He said, "It's not UART on its own that immediately calls into `printk()`, that would be trivial to fix; it's all those subsystems that [the] serial console driver can call into."

He added, "For instance, kernel/workqueue.c – it may `WARN_ON/printk` in

Author

The Linux kernel mailing list comprises the core of Linux development activities. Traffic volumes are immense, often reaching 10,000 messages in a week, and keeping up to date with the entire scope of development is a virtually impossible task for one person. One of the few brave souls to take on this task is **Zack Brown**.

various cases. And those `WARNs/printks` are OK. Except for one thing: `workqueue` can be called from a serial console driver, which suddenly will turn those `WARNs/printks` into illegal ones, due to possible deadlocks. And serial consoles can call into `WQ`. Not directly, but via `TTY` code.”

He added, “IOW, there is this tricky ‘we were called from a serial driver’ context, which is hard to track, but `printk_safe` can help us in those cases.”

But Linus replied, “We already have the whole `PRINTK_SAFE_CONTEXT_MASK` model that only adds it to a secondary buffer if you get recursion. Why isn’t that triggering? That’s the whole point of it. I absolutely do *not* want to see any crazy changes to `TTY` drivers. No, no, no.”

And Sergey said this was exactly what his code did – it enhanced `printk_safe` to handle this new set of circumstances.

The thread ended inconclusively. But the desire to improve `printk()` is a real one. Absolutely everyone is on board, if they could only figure out the right way to do it.

Kernel Internationalization (or Not)

Sometimes discussions on particular topics seem to come out of nowhere. Recently David Howells posted some sample code for one of the kernel’s new library interfaces. Pavel Machek observed that the code would output English language error messages and remarked, “Not sure that is reasonable, as that is going to cause problems with translations.” David said that simply outputting error codes wouldn’t cut it, because the logs by default would be output via `printk()`, and therefore needed to be human-readable.

Pavel replied, “Errors should have numbers, and catalog explaining what error means what. That way user space can translate, and it is what we do with `errno`. I believe numbers are best. If you hate numbers, you can still use strings, as long as you can enumerate them in docs (but it will be strange design). But anything else is not suitable, I’m afraid.”

David objected, saying the errors consisted of various components that needed to be calculated separately and used as parameters to produce a unified message. But Pavel insisted that regardless of these requirements, it was essential for user code to be able to parse the

errors as well. As a counter-example against plain text messages, he said, “One way is to pass (`'not enough pixie dust (%d too short)` on device `%s in %s', 123, 'foo', 'wardrobe'`). But if you pass it as one string, it becomes hard/impossible to parse. (For example if device is named `'foo in bar'`.)”

At this point in the debate, Linus Torvalds came in with a statement of policy. He seemed to agree partly with David, in that user code did not need to parse the errors, and partly with Pavel, in that errors should still be numerical codes rather than text. Linus said:

“We don’t internationalize kernel strings. We never have. Yes, some people tried to do some database of kernel messages for translation purposes, but I absolutely refused to make that part of the development process. It’s a pain.

For some GUI project, internationalization might be a big deal, and it might be ‘TheRule(tm)’. For the kernel, not so much. We care about the technology, not the language.

So we’ll continue to give error numbers for ‘an error happened’. And if/when people need more information about just what triggered that error, they are as English-language strings. You can quote them and google them without having to understand them. That’s just how things work.

Let’s face it, the mount options themselves are already (shortened) English language words. We talk about `mtime` and create.

*There are places where localization is a good idea. The kernel is *not* one of those places.”*

And Matthew Wilcox pointed out that the `gettext` tool, “uses the English text as a search string and replaces it with the localized string. This is a very common design!”

Pavel affirmed that English language in the kernel was not a problem, but he said that error numbers currently allowed user code to read kernel messages and produce localized language output. He said, “User space does [a] good job translating errors, and it would be good to keep that capability.”

Linus pointed out that even `gettext` was not ideal. And in terms of the readability of error messages, he said, “if you are messing with mount options and things like that, you’d better be able to google the incomprehensible words.

Most of them will be incomprehensible even if you're a native speaker."

He added in a separate email, that kernel errors could generally be highly specialized and auto-generated by the kernel code to match very specific circumstances. At that point he said, "Once the string has been generated, it can now be thousands of different strings, and you can't just look them up from a table any more [...] The string will have various random key names etc. in it."

For message localization, at least for the kernel, Linus declared:

"I really think the best option is 'Ignore the problem'. The system calls will still continue to report the basic error numbers (EINVAL, etc.), and the extended error strings will be just that: extended error strings. Ignore them if you can't understand them.

That said, people have wanted these kinds of extended error descriptors forever, and the reason we haven't added them is that it generally is more pain than it is necessarily worth."

Theodore Ts'o said he also felt this really wasn't a problem worth dealing with. Even super-complex kernel error messages would be best handled, he said, by pushing them off into user space and letting user code sort it out and translate any of it into whatever languages might be needed.

Pavel essentially agreed with all of the above. But since David's original code was something new, rather than a re-vamp of existing code, he said, "we have chance to do it right for a minimum price (because the interface is new, we don't need compatibility)."

But Theodore drove his point home, saying, "I think David's proposal of just returning Error: followed by English text is just fine, and doing more than that is overdesign. The advantage of `dmesg` is that it's well understood by everyone that `dmesg` is English text meant for experts. The problem once we move away from `dmesg`, this tends to cause the I18N brigade to start agitating for something more complicated. And if the only choices were some complex I18N horror through a system call, or just leaving the (English) text messages in `dmesg`, I'd vote for `dmesg` for sure."

The discussion continued for a bit, with various developers arguing for and against various translation techniques,

and explaining the problems with those techniques. Then Linus said, "Really. No translation. No design for translation. It's a nasty, nasty rat-hole, and it's a pain for everybody."

He added, "the fact is, I want simple English interfaces. And people who have issues with that should just not use them. End of story. Use the existing error numbers if you want internationalization, and live with the fact that you only get the very limited error number. It's really that simple."

David replied, "fine by me," and that was the end of the discussion.

The really strange and interesting thing about this discussion is that one of the things corporate Internet companies really focus on these days is accessibility, both for handicaps and many languages. There's political pressure to do it for sensitivity reasons, and economic pressure to do it as a way to open up more markets. An Internet company really can't reach a certain level of success and growth without migrating its interfaces to a fully internationalized infrastructure.

Meanwhile, the Linux kernel has already taken over the entire world, running absolutely everywhere, relied on by absolutely everyone – even people who think they are Windows users – and the issue of internationalization is barely even a question.

Perhaps this is because open source projects have the luxury of shunting off portions of their activity to other projects that either exist now or will exist as soon as someone recognizes they need them. Corporate projects generally can't rely on getting that kind of help from their competitors.

Kernel Encryption and Secure Boot

A persistent security debate in the Linux kernel world centers around media companies trying to prevent the owner of a given system from having full control over that system. Technically, that does constitute a security issue, since it's about access control. And there is a lot of money at stake because media companies want to offer users all sorts of access to their media products, if those products can be protected against copyright infringement. But the kernel developers refuse to implement or accept such features, because they believe the

machine owner has the ultimate right to control their own system.

The debate can become convoluted. Often the media companies don't want to admit that a proposed patch is really intended to take control away from the machine owner, because they know the patch would never be accepted in that case. And just as often, the kernel developers doesn't want to seem like they are arbitrarily rejecting patches for reasons to which the patch submitter has not admitted. What tends to follow is therefore a strange dance of call-and-response, where the kernel developers try to get the patch submitter to admit the true purpose of their patch, while the patch submitter tries to present the patch as having a general-purpose security value beyond any side effect of taking control away from the machine owner.

Recently Chen Yu from Intel wanted to add a kernel feature to encrypt the running kernel image when the user hibernated the system. This would involve installing a kernel module to generate a key from a user's passphrase, encrypting the kernel image, and decrypting when the user resumed the system.

Pavel Machek pointed out that `uswsusp` (userspace software suspend) already provided an encryption feature. He asked Chen to explain the specific security attacks his kernel-based encryption system would guard against that would be better than `uswsusp`'s approach.

Chen referenced the patch log, which read, "Generally the advantage is: Users do not have to encrypt the whole swap partition as other tools. After all, ideally kernel memory should be encrypted by the kernel itself." But Pavel was not satisfied and reiterated that Chen's explanation did not address the specific security attacks and defenses that would be better than `uswsusp`. Pavel also added, "Also note that [Chun-Yi Lee] has patch series which encrypts both in-kernel and `uswsusp` hibernation methods. His motivation is secure boot. How does this compare to his work?"

Chen replied that it was better for the kernel to encrypt the running system than to have user space do it, because it avoided having to transfer the kernel memory in plain-text from kernel space to user space. It would also save time by not having to copy data between kernel and user space. By staying in the

kernel, the user would not have to worry about userspace bugs introducing security holes. And he added that he had been collaborating with Chun-Yi on these patches.

Chun-Yi also spoke up, explaining:

“The pros of my solution is that the signed/encrypted snapshot image can be stored to anywhere. Both in-kernel and user space.”

Yu’s patch is encrypt the page buffer before sending to block io layer for writing to swap. The main logic is applied to swap.c. It’s against the swap solution in-kernel.

The pros of Yu’s solution is that it encrypts the compressed image data. So, for the huge system memory case, it has better performance.

Yu’s plan is using the sysfs to switch different encrypt/sign solutions. And, we will share encrypt/sign helper and key manager in the above two solutions.”

Pavel was unconvinced. He remarked flatly, “Answer to bugs in user space is not to move code from user space to kernel.”

Pavel also asked, “So your goal is to make hibernation compatible with kernel lockdown? Do your patches provide sufficient security that hibernation can be enabled with kernel lockdown?” And Oliver Neukum requested clarification, saying, “if the key comes from user space, will that be enough?” And Pavel replied, “Yes, that seems to be one of problems of Yu Chen’s patchset.”

Pavel explained that he was personally opposed to doing hibernation encryption in the kernel since it could be (and was already) done successfully in user space. But he acknowledged, “We have this weird thing called secure boot [that] some people seem to want. So we may need some crypto in the kernel – but I’d like something that works with `uswsusp`, too. Plus, it is mandatory that patch explains what security guarantees they want to provide against what kinds of attacks.”

In response to Oliver’s question about the key coming from user space, Chen replied, “we once tried to generate key in kernel, but people suggest to generate key in user space and provide it to the kernel, which is what `ecryptfs` do currently, so it seems this should also be safe for encryption in kernel.”

Chen also offered a summary of the difference between his patch and

Chun-Yi’s, saying, “The only difference between Chun-Yi’s hibernation encryption solution and our solution is that his strategy encrypts the snapshot from scratch, and ours encrypts each page before them going to block device. The benefit of his solution is that the snapshot can be encrypt[ed] in kernel first thus the `uswsusp` is allowed to read it to user space even kernel is lock down. And I had a discussion with Chun-Yi that we can use his snapshot solution to make `uswsusp` happy, and we share the crypto help code and he can also use our user provided key for his signature. From this point of view, our code are actually the same, except that we can help clean up the code and also enhance some encryption process for his solution.”

In response to Pavel’s post about Secure Boot, Oliver remarked, “maybe we should state clearly that the goal of these patch set[s] is to make Secure Boot and STD coexist. Anything else is a nice side effect, but not the primary justification, right? And we further agree that the model of Secure Boot requires the encryption to be done in kernel space, don’t we? Furthermore IMHO the key must also be generated in trusted code, hence in kernel space. Yu Chen, I really cannot see how a symmetrical encryption with a known key can be secure.”

And Pavel added, “I don’t think generating key in user space is good enough for providing guarantees for secure-boot.”

Pavel also continued to ask for specific security dangers, and how any of this code might address it, to which Oliver explained:

“Unsigned code must not take over the privilege level of signed code. Hence:

- 1. Unsigned code must not [be] allowed to read sensitive parts of signed code’s memory space*
- 2. Unsigned code must not be able to alter the memory space of signed code – snapshots that are changed must not be able to be resumed”*

But he also asked why key generation in user space would not be secure, to which Pavel replied, “Because then, user space has both key (now) and encrypted image (after reboot), so it can decrypt, modify, re-encrypt...?”

The discussion continued, with new versions of the patches coming out and further feedback. At one point Yu Chen

said, “I’m still a little confused about the ‘resume’ phase. Taking encryption as example (not signature), the purpose of doing hibernation encryption is to prevent other users from stealing RAM content. Say, user A uses a passphrase to generate the key and encrypted the hibernation snapshot and stores it on the disk. Then if user B wants to do a hibernation resume to A’s previous environment, B has to provide the same passphrase. If I understand correctly, the secret key is saved in header and stored on the disk. Which means, any one can read the header from the disk to get the secret key in trampoline thus decrypt the image, which is not safe.”

But Pavel replied, laying his cards on the table:

“No, I don’t think that’s purpose here.

Purpose here is to prevent user from reading/modifying kernel memory content on machine he owns.

Strange as it may sound, that is what ‘secure’ boot requires (and what Disney wants).”

And Yu Chen, laying his cards down too, said, “Ok, I understand this requirement, and I’m also concerning how to distinguish different users from seeing data of each other.”

At one point Oliver said, “While the system is running and the `fs` is mounted, your data is as secure as root access to your machine, right? You encrypt a disk primarily so data cannot be recovered (and altered) while the system is not running. Secure Boot does not trust root fully. There is a cryptographic chain of trust and user space is not part of it.”

Ultimately there was no resolution to the discussion. The debate is so odd! Every time Pavel asked what security weaknesses the patch addressed, he seemed really to be inviting the patch developers to admit that instead of addressing security concerns relevant to the user, they were trying to keep the user locked out of their own system. And for that same reason, the developers seemed to avoid actually listing the security weaknesses Pavel wanted.

At the same time, as Pavel said, Secure Boot is a reality. And as more and more patches arrive to support it, Disney and others do seem to be gradually making inroads towards eventually locking the user out of their own system. ■■■

Using the hacker community to uncover bugs

Hacker-Powered Security

By Swapnil Bhartiya

Mårten Mickos is one of the most respected members of the open source world. The former CEO of MySQL AB during its prime now serves as the CEO of HackerOne, a vulnerability coordination and bug bounty platform. I sat down with Mickos to understand HackerOne's purpose and his perspective on the security of open source software.

HackerOne's Role

In layman's terms, HackerOne brings the hacker community to an organization to hack into their code in search of vulnerabilities. As Mickos said, "Sometimes we joke that if you are going to be hacked anyway, it's better to get hacked by someone you can trust." HackerOne has built a platform for secure intelligence report sharing and payment, along with a reputation system for hackers.

When an organization announces a bug bounty program through HackerOne, the hacker community starts looking at the organization's code and filing



their reports. The platform enables the bug bounty program's organizer to vet these vulnerabilities. The hacker who filed the report gets rewarded.

"HackerOne serves as the portal connecting organizations with the largest community of over 200,000 registered ethical hackers and connecting hackers with more active programs than any other platform," said Mickos.

HackerOne's approach is simple but effective. It acts only as a mediator, without getting involved with the code itself. "HackerOne does not review cus-

tomers' code unless our technical program manager team is instructed to do so in order to help the organization evaluate the severity and advise on a bounty payment," clarified Mickos.

Community-Driven Security

HackerOne has a massive community of more than 200,000 white-hat hackers in its network. "The hacker community is filled with smart, curious, communal, and charitable

human beings. Over 90% of hackers are under the age of 35, 58% are self-taught, and 44% are IT professionals. They come from over 90 countries including the US, India, UK, etc.," said Mickos.

Hackers are rewarded based on the vulnerabilities they find. HackerOne works with each customer to carefully outline a bounty structure based on the bug's severity and its impact on the organization. Hackers are rewarded based on the assessment of each valid bug reported.

“A total of 116 bug reports over \$10,000 were paid out in the past year with the amount paid for critical issues rising to over \$2,000 on average and organizations offering as much as \$250,000,” said Mickos.

Customers determine bounties based on the severity and potential effect on the organization. Most organizations pay bounties through the HackerOne platform. HackerOne requires tax forms from every hacker in order for them to get paid.

To date, HackerOne has paid more than \$31 million in bounties. “Unlike Apple, that takes a 30% cut from developers when they publish their paid app on the App Store, HackerOne doesn’t take any cut from hackers. Hackers will always receive 100% of the bounties they earn,” said Mickos.

But money is not the only motivating factor behind the HackerOne community. “The biggest takeaway of the 2018 Hacker Report was that the ethical hacking community is eager to do good in the world. They are already finding vulnerabilities. Hackers are motivated by opportunities to learn, be challenged, and have fun more than [by] money. While money definitely still attracts hackers to different programs, it’s not the key driver of what they do,” said Mickos.

Hack the USA

HackerOne helps both the public and the private sector. “We work with them [the private sector] to find vulnerabilities in their systems. Every vulnerability we find and fix leaves fewer possibilities for criminals to break in. We are reducing the cyber risk with every step we take,” he said.

In 2016, HackerOne signed a deal with the US Department of Defense (DoD) Defense Digital Service (DDS) team to hack the Pentagon. It became the first bug bounty program in the history of the federal government.

The first vulnerability report was filed within 13 minutes of the launch of the Hack the Pentagon challenge [1]. In just six hours, around 200 reports were filed, and a new report was filed every 30 minutes. During the entire project, more than 1,400 hackers participated in the hack, more than 138 legitimate vulnerabilities were found, and \$75,000 was paid in bug bounty rewards.

The success of Hack the Pentagon led to more projects – Hack the Army [2] and Hack the Air Force. In total, the federal government awarded more than \$300,000 in rewards. Looking at the massive defense budget, this number might look small, but it’s not.

“It’s not a small sum, but if we had gone through the normal process of hiring an outside firm to do a security audit and vulnerability assessment, which is what we usually do, it would have cost us more than \$1 million,” said former Secretary of Defense Ash Carter regarding HackerOne’s Hack the Pentagon.

HackerOne and the DoD just kicked off the sixth bug bounty challenge for the US government. At the kickoff event in Las Vegas, Hack the Marine Corps paid out over \$80,000 to ethical hackers who surfaced 75 unique valid vulnerabilities in public-facing digital assets.

Conclusion

All said and done, HackerOne is not for everyone. It’s intended for organizations whose software can be accessed from the outside. According to Mickos, HackerOne works the best for any organization that is connected to the Internet.

“It should have a process for receiving vulnerability reports from third parties and resolving known vulnerabilities. Some of HackerOne’s most successful programs vary greatly by industry, attack surface, and budget. They are run by teams that prioritize relationships with the hacker community and are dedicated to resolving vulnerabilities as quickly and accurately as possible,” said Mickos.

Some of HackerOne’s major customers include the DoD, General Motors, Google, Twitter, GitHub, Nintendo, Lufthansa, Panasonic Avionics, Qualcomm, Starbucks, Dropbox, Intel, and the CERT Coordination Center, among others.

Are you ready to get hacked? ■■■

Info

- [1] Hack the Pentagon challenge:
<https://www.hackerone.com/resources/hack-the-pentagon>
- [2] Hack the Army project:
<https://www.hackerone.com/blog/Hack-The-Army-Results-Are-In>

The intersection of DEVELOPMENT and OPERATIONS

Check out our new ADMIN DevOps corner!

www.admin-magazine.com/DevOps

SPONSORED BY




**Linux
Professional
Institute**

Exploring Microsoft's .NET Core on Linux

Sure I Can, but Should I?

Python enjoys a strong foothold in the Linux space, and Go is quickly becoming a contender. Is it really worth considering an alternative language environment from former Linux-foe Microsoft? On close inspection, the open source .NET Core actually does offer some compelling features that could be worthy of attention. *By Bradley Campbell*



Microsoft announced that they were releasing .NET Core as an open source project in 2014, and version 1.0 appeared in 2016. Since then, Linux developers and admins have circled .NET warily.

Many wondered whether Microsoft's support for open source was a gimmick, or worse, a trick leading to some kind of unseen lock-in trap. Since that time, however, Redmond has continued to invest time and energy into supporting .NET on Linux, and Microsoft's new affinity for Linux has become evident in other areas as well, from the cloud to the Internet of Things (IoT). Some developers are beginning to take a closer look at .NET Core and what benefits it might offer for the Linux space.

The full version of .NET is a vast framework supporting several different languages and components. See the box entitled "Falling Deeper into the .NET (Framework)." Programmers accustomed to working with statically-typed, object-oriented languages will find C# offers a robust set of classes and APIs, as well as generics [1] and reflection [2]. If, instead, you're looking to move in the direction of functional programming, .NET Core has you covered with its complete support for the F# programming language [3]. Static compilation options remove the need to deploy the run time on target systems, because you can just deploy a compiled executable. (Similar capabilities are provided for the Python world through tools like PyInstaller [4] but are rarely leveraged by Pythonistas.)



Falling Deeper into the .NET (Framework)

The .NET Framework has long been a mainstay of web-based applications development for shops mostly dependent upon Microsoft and Windows-based technology stacks. .NET Framework apps are easily integrated into the Microsoft ecosystem, including platforms leveraging Microsoft Server-based operating systems and IIS-based hosting. Microsoft's development ecosystem and toolchain, including Visual Studio, have made it easy to work with the .NET Framework; features such as push-button deployment [6] of locally developed applications into IIS-based servers have created a strong one-to-one cohesion and mindset across the industry of .NET Framework apps and Microsoft-based application hosting platforms. Historically, and in a significant majority of cases where .NET-related technologies are used, this has been a true statement. The introduction of the Mono framework within recent years, while offering a .NET Framework interop in Linux-based development and hosting environments, has done little to erode this paradigm. Although Mono has created a bit of a Linux-based .NET Framework subculture, the implicit correlation between .NET Framework apps within Microsoft ecosystems largely remains an unchallenged notion in the industry.

The power of the .NET Framework – versatile back-end development with C#, F#, and Visual Basic; flexible web and desktop application frameworks; and powerful abstraction tools such as Language-Integrated Query (LINQ) and the Entity Framework – have given it a very dedicated following. Let's dive into some of these features a bit:

Languages

- C#: An object-oriented language that took many cues from Java. Microsoft claims that “[i]ts roots in the C family of languages makes C# immediately familiar to C, C++, Java, and JavaScript programmers.” [7]

Longtime C# and Linux enthusiasts will correctly point out that Mono has offered a .NET interop to developers on Linux-based platforms for years, but there are some differences between .NET Core and Mono. Mono supports more of the .NET Framework APIs than does .NET Core; however, .NET Core focuses on portability between platforms by removing dependencies on platform-specific technologies such as DirectX. According to Microsoft, the out-of-the-box version of .NET Core “...includes a single application model – console apps – which is useful for tools, local services, and text-based games” [5], though add-ons are available.

The Curious Skeptic

So I've got you thinking there might be something to this whole .NET Core thing and now you want to kick the tires? Microsoft's made it pretty easy to get started. On an Ubuntu 16.04 box, the script in Listing 1 will get you going in no time.

After you run the script in Listing 1, `dotnet` will be symlinked into `/usr/bin`. The `dotnet` command should immediately be available from the command line.

Listing 1: `install_dotnet.sh`

```
01 #!/bin/bash
02
03 echo "Getting license package from Microsoft"
04 wget -q https://packages.microsoft.com/config/ubuntu/16.04/packages-microsoft-prod.deb
05 echo "Installing license package"
06 sudo dpkg -i packages-microsoft-prod.deb
07
08 echo "Installing packages via apt-get"
09 sudo apt-get install apt-transport-https
10 sudo apt-get update
11 sudo apt-get install dotnet-sdk-2.1
12 echo "Done"
```

- F#: A functional programming language (Scala and Haskell being some of the better-known examples of a functional language), which can also be used in object-oriented and imperative paradigms.
- Visual Basic: A Basic dialect that not only finds itself at home in the .NET Framework, but that can also be found embedded as a scripting language in Microsoft Office applications [8].

Desktop and Web Frameworks

.NET supports multiple frameworks to facilitate app development for different purposes and platforms, including:

- ASP.NET MVC: Model-View-Controller (MVC) provides a boilerplate application supporting the MVC paradigm [9].
- WPF and WCF: Windows Presentation Foundation (WPF) and Windows Communication Foundation (WCF) both provide a platform to create desktop applications on Windows (each with different capabilities).

Additional Features

- LINQ: A general-purpose in-code abstraction layer over different types of data storage, from relational database back ends to XML documents [10].
- Entity Framework: The Entity Framework is an object-relational mapping (ORM) baked into the .NET Framework; unsurprisingly, it leverages LINQ for direct access abstractions [11].

Although Microsoft has made great strides with making the .NET Framework more open to other platforms, some aspects of the .NET Framework, including WCF and WPF, are not available on other platforms. .NET Core and the .NET Framework share common functionality, but .NET Core's focus is on interoperability between platforms, hence the absence of WPF and WCF in .NET Core.

So Can I Do Anything Useful?

Creating a new console application is straightforward:

```
dotnet new console -o my_app
```

For the purpose of this article, I'll create an application called `webserver` [12]. From a suitable directory, run this command:

```
dotnet new console -o webserver
```

After some console output, you'll have a new `webserver` directory with the contents in Listing 2.

Listing 2: webserver Directory Contents

```
01 drwxrwx--- 1 root vboxsf 256 Jun 28 19:14 obj/  
02 -rwxrwx--- 1 root vboxsf 2491 Jun 28 18:51 Program.cs*  
03 -rwxrwx--- 1 root vboxsf 178 Jun 28 16:57 webserver.csproj*
```

Open the `Program.cs` file in Listing 2, replace it with the contents of Listing 3, and then save the changes. The compiled executable that results from this code will create a TCP listener on `localhost` running on port 13000.

From Listing 3, you will notice some similarities to more familiar languages like Python or Go. The `using` statements are similar to Go's `import` or Python's `import/from x import y` statements, importing modules to implement only needed functionality.

As you can see based on the nesting in the code example, namespaces serve as containers for classes. As a corollary to Go, namespaces are similar to packages; in relation to Python, namespaces are similar to encapsulating several classes inside of a single file that might be imported as part of a `from ...` statement.

Similarly to Java, classes in `C#` can be public or private (much like Java, inner classes can be private). As a point of difference, Python doesn't provide any mechanisms with which to limit access to classes or class members. While Go has a mechanism to provide privacy, it is scoped at the package level, which is less granular than struct types and methods that the package could encapsulate.

In the case of the `MyTcpListener` class, the class mostly exists as a semantic requirement, as the class itself only implements a `void Main` method, meaning that the function returns no value. It exists solely to implement a listener loop that runs the server.

`C#` comes with a robust set of native functionality, as demonstrated in the program; network stream handlers and TCP clients are first-class citizens, because the program only works with core packages and none of the base constructs are derived from custom implementations. While this is of course also true for almost every other language you may work with on a regular basis, it is also worth pointing out here.

Next up, run the following command:

```
dotnet build -r linux-x64
```

This command will build your project, targeting the `linux-x64` architecture.

The `build` command will create a new `bin` directory. To run the compiled executable, run `bin/Debug/netcoreapp2.1/linux-x64/webserver` from the `webserver` directory. The web server process will start up, listening on port 13000. In another console, you can connect to the server by running Netcat, a simple utility that writes data across a network connection:

```
nc localhost 13000
```

Netcat will connect to the server running on port 13000. In the console running Netcat, type a random string and hit the Enter key – you'll see an acknowledgment of receipt in the server's console and a message indicating the response the server will send to the client. Back on the Netcat side, you'll see the message back from the server. See Figures 1 and 2 for screenshots of consoles of a Netcat-based client sending a message (and getting a response from the server) and the .NET Core app receiving the message from the Netcat client and sending a response back to the client.

The `dotnet build` command builds a debug version of your app. To create a release version, run:

```
dotnet publish --self-contained -f    
netcoreapp2.1 -c Release -r linux-x64
```

After you run the `publish` command, a `Release` directory will exist under `bin` with release artifacts. Congrats – you've created your first app using .NET Core!

Building for Other Operating Systems

In just a short time, I used .NET Core to build executables for a Linux x64 architecture by specifying the `linux-x64` runtime identifier. There are several other run-time identifiers and platforms available, which are described in Microsoft's documentation [13]. Even from a Linux workstation, you can build a binary that will run on an OS X/Mac OS-based machine by specifying the `osx-x64` run-time identifier or for a Windows system by specifying the `win-x64` runtime identifier. Release artifacts are placed into separate directories based on the run-time identifier, so it is quite possible to build releases for multiple platforms from a single workstation or Continuous Integration (CI) server.

Cutting-edge developers might wonder if serverless support exists for .NET Core-based solutions. Amazon Web Services (AWS) recently announced Lambda support for .NET Core 2.1 [14] [15]. .NET Core/Standard is available through version 2 of Microsoft's Azure Functions [16] runtime. Currently, Google

```
bradmatic@bradmatic-VirtualBox: ~/vm_container  
bradmatic@bradmatic-VirtualBox:~/vm_container$ nc localhost 13000  
Foo  
== FOO ==
```

Figure 1: A console shown running the Netcat client.

```
bradmatic@bradmatic-VirtualBox: /media/sf_bradmatic/webserver  
bradmatic@bradmatic-VirtualBox:/media/sf_bradmatic/webserver$ bin/De  
bug/netcoreapp2.1/linux-x64/webserver  
Waiting for a connection... Connected!  
Received: Foo  
Sent: == FOO ==
```

Figure 2: A console shown running the .NET Core-based server.

**Listing 3: Program.cs**

```
01 using System;
02 using System.IO;
03 using System.Net;
04 using System.Net.Sockets;
05 using System.Text;
06
07 namespace webserver
08 {
09     class MyTcpListener
10     {
11         public static void Main()
12         {
13             TcpListener server=null;
14             try
15             {
16                 // Set the TcpListener on port 13000.
17                 Int32 port = 13000;
18                 IPAddress localAddr =
19                     IPAddress.Parse("127.0.0.1");
20
21                 // TcpListener server = new TcpListener(port);
22                 server = new TcpListener(localAddr, port);
23
24                 // Start listening for client requests.
25                 server.Start();
26
27                 // Buffer for reading data
28                 Byte[] bytes = new Byte[256];
29                 String data = null;
30
31                 // Enter the listening loop.
32                 while(true)
33                 {
34                     Console.Write("Waiting for a connection... ");
35
36                     // Perform a blocking call to accept requests.
37                     // You could also use server.AcceptSocket()
38                     // here.
39                     TcpClient client = server.AcceptTcpClient();
40                     Console.WriteLine("Connected!");
41
42                     data = null;
43
44                     // Get a stream object for reading and writing
45                     NetworkStream stream = client.GetStream();
46
47                     int i;
48
49                     // Loop to receive all the data sent by the
50                     while((i = stream.Read(bytes, 0,
51                         bytes.Length))!=0)
52                     {
53                         // Translate data bytes to a ASCII string.
54                         data = System.Text.Encoding.ASCII.GetString(
55                             bytes, 0, i);
56                         Console.WriteLine("Received: {0}", data);
57
58                         // Process the data sent by the client.
59                         data = "==" + data.Substring(0, data.Length
60                             - 1).ToUpper() + " ==\n";
61
62                         byte[] msg =
63                             System.Text.Encoding.ASCII.GetBytes(data);
64
65                         // Send back a response.
66                         stream.Write(msg, 0, msg.Length);
67                         Console.WriteLine("Sent: {0}", data);
68                     }
69
70                     // Shutdown and end connection
71                     client.Close();
72                 }
73             }
74             catch(SocketException e)
75             {
76                 Console.WriteLine("SocketException: {0}", e);
77             }
78             finally
79             {
80                 // Stop listening for new clients.
81                 server.Stop();
82             }
83
84             Console.WriteLine("\nHit enter to continue...");
85             Console.Read();
86         }
87     }
88 }
```

Cloud's Cloud Functions product does not support any .NET language or runtime [17].

Conclusion

This article examined the C# language in comparison to some of the powerhouse languages and runtimes that currently dominate the development and systems administration/DevOps landscape

(particularly in the context of Linux-based environments). I described the ease with which you can bootstrap a system to build apps with .NET Core, how to create a new application, and how to build that application for deployment in testing and production scenarios. Throughout these steps, I walked through some commands you can use with the dotnet CLI tool to achieve the desired outcome. Lastly, I looked at the platforms that currently support

Shop the Shop

shop.linuxnewmedia.com

Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

Want to subscribe?

Searching for that back issue you really wish you'd picked up at the newsstand?

DIGITAL & PRINT SUBSCRIPTIONS



SPECIAL EDITIONS



code written and built on the .NET Core platform, from traditional operating systems to cloud-based serverless function environments. The power, portability, and flexibility of the .NET Core platform, and Microsoft's increasing affinity towards open-source and interoperable solutions, will hopefully leave a marker in your mind as you evaluate potential tools for building your next solution. ■■■

Author

Bradley Campbell is a full-stack and DevOps engineer with experience across multiple technology stacks and industries. Bradley holds a Bachelor of Arts degree in Economics from the University of Virginia. He holds eight of nine AWS certifications, is a Certified Jenkins Engineer, and is CompTIA Security+ CE certified. His programming background is primarily centered around Python, Perl, Go, and JavaScript, although he is always keen on exploring new languages and skills to add to his toolbox. You can find him at <https://bradcod.es> or [@geekmuse](https://twitter.com/geekmuse).



Info

- [1] Generics (C# Programming Guide): <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/index>
- [2] Reflection (C#): <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/reflection>
- [3] Get started with F# with the .NET Core CLI: <https://docs.microsoft.com/en-us/dotnet/fsharp/get-started/get-started-command-line>
- [4] PyInstaller: <https://www.pyinstaller.org/>
- [5] .NET Core Guide: <https://docs.microsoft.com/en-us/dotnet/core/index#workloads>
- [6] Click-once deployment: <https://docs.microsoft.com/en-us/dotnet/framework/deployment/deployment-guide-for-developers#clickonce-deployment>
- [7] .NET languages: <https://www.microsoft.com/net/learn/languages>
- [8] Getting Started with VBA in Office: <https://docs.microsoft.com/en-us/office/vba/library-reference/concepts/getting-started-with-vba-in-office>
- [9] Learn About ASP.NET MVC: <https://www.asp.net/mvc>
- [10] .NET LINQ: https://msdn.microsoft.com/en-us/library/bb308959.aspx#linqoverview_topic1
- [11] Entity Framework 6: <https://docs.microsoft.com/en-us/ef/ef6/>
- [12] This application is based heavily upon an example published by Microsoft: <https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.tcplistener?redirectedfrom=MSDN&view=netframework-4.7.2>
- [13] .NET Core RID Catalog /Using RIDs: <https://docs.microsoft.com/en-us/dotnet/core/rid-catalog#using-rids>
- [14] AWS Lambda: <https://aws.amazon.com/lambda/>
- [15] AWS Lambda Supports .NET Core 2.1: <https://aws.amazon.com/about-aws/whats-new/2018/06/lambda-supports-dotnetcore-twopointone/>
- [16] Supported languages in Azure functions: <https://docs.microsoft.com/en-us/azure/azure-functions/supported-languages>
- [17] Writing Cloud Functions: <https://cloud.google.com/functions/docs/writing/>

LIBRECON

powered by CEBIT®

THE REFERENCE EVENT FOR
OPEN TECHNOLOGIES IN THE
SOUTH OF EUROPE

21+22 Nov · 2018
Bilbao (Spain)

STATE OF THE ART OF TECHNOLOGIES
LIKE THESE



AND MUCH MORE COMING SOON!

Stay tuned to our website and suscribe to our newsletter

LATEST TRENDS IN OPEN
SOURCE APPLIED TO:

Industry

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

Public authorities

The most transparent administrations are based on open technologies.

Finance

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

ORGANIZED BY:

[esle]



POWERED BY:

CEBIT®



Deutsche Messe

FURTHER INFORMATION:

www.librecon.io

info@librecon.io

twitter.com/librecon

facebook.com/librecon

es.linkedin.com/in/librecon

A BeOS-inspired desktop operating system

Haiku Close Up

The long-awaited Haiku OS beta release has arrived. This BeOS-inspired operating system may finally be ready for daily use. *By Alexander Tolstoy*



When it comes to open source operating systems (OSs), GNU/Linux is the undisputed champion, but it does have contenders worth noting. Haiku OS, a microkernel OS inspired by BeOS (an OS for multimedia desktop use developed in the 1990s) is one such contender. Haiku brings the BeOS foundation to the modern age by adding up-to-date drivers and populating its software store, HaikuDepot, with the latest open source titles commonly found on Linux.

Haiku Specs

Haiku is a single-user desktop OS based on the re-implemented NewOS kernel

from BeOS – a hybrid kernel designed from the bottom up to be “pervasively multithreaded” (long before multicore CPUs emerged). Haiku is not a Unix-based system, but it has a POSIX compatibility layer added on top to provide a standardized shell and GNU userland utilities, such as *coreutils*. The OS is written in C++ and sports modular design with most components, or “kits,” referring to specific functions (kernel, input, media, etc.). The default filesystem, OpenBFS, is a modern 64-bit journaling filesystem with support for case-sensitive names. The original compiler is the historic GCC 2.95, which makes Haiku binary-compatible with the legacy BeOS applications. However, Haiku also

supports and provides modern compilers, including Rust and the GCC 7 compiler (v. 8 is in the works).

Haiku has relatively modest hardware requirements. It needs an x86-compatible CPU (Pentium II or above), 128MB of RAM (1GB is recommended), and at least 700MB of hard drive space. Our aging Sandy Bridge-based machine felt like a super-speed jet with Haiku (Figure 1)!

A Blast from the Past

Haiku has come a long way from a geeky project to an OS that can be confidently recommended for daily use. Haiku’s predecessor, BeOS, was developed by Be Inc. and founded by former Apple CEO Jean-Louis Gassée in the early 1990s.

Lead image by Annie Spratt on Unsplash

BeOS took advantage of then-modern computer hardware features (e.g., symmetric multiprocessing) by utilizing modular I/O bandwidth, pervasive multithreading, preemptive multitasking, and a 64-bit journaling filesystem known as the Be File System (BFS). The BeOS GUI displayed the principles of clarity and a clean, uncluttered design. In 2001, BeOS copyrights were sold to Palm Inc. Initially, the OpenBeOS project attempted to update the discontinued BeOS 5.0.x. To avoid infringement of the BeOS trademark, BeOS enthusiasts renamed the project to continue updating the OS. As a result, Haiku OS was born in 2004. Four years later, the project registered its own website [1].

Early on, very few people were involved in Haiku development. Moreover, Michael Phipps, the founder of the Haiku project, quit in 2007 leaving the project's future uncertain. This partially explains the relatively modest amount of work completed by the time Haiku R1/Alpha 4.1 was released in late 2012.

Today, Haiku has more than 100 contributors, mainly from the US and the EU, and their hard work really shows. For years, Haiku 64-bit nightly builds were unable to run older 32-bit apps; this problem was recently fixed, so now Haiku includes a set of 32-bit compatibility libraries. It also finally supports UEFI boot, boasts Ethernet and wireless driv-

ers compatible with FreeBSD, ships with a working software store out of the box, and offers lots of great open source software titles, including LibreOffice and Krita. Consequently, using Haiku's nightly images makes much more sense than sticking with the aging Alpha version because of all the latest software, driver stacks, and bug fixes.

Hardware Support

Haiku hardware support is reminiscent of Linux from the early 2000s. You may encounter systems that will not boot Haiku Live media because of unsupported graphics (e.g., Nvidia 8x series and newer). However, Radeon and Intel chips are supported, although on certain configurations, Haiku will default to the VESA driver, which does not provide hardware acceleration. Regardless, the OS is still so fast and fluent that you hardly notice any lack of responsiveness even with VESA; the system will still perform fine. Nevertheless, beginning with Google Summer of Code 2017, work on Intel's hardware acceleration in Haiku has been a focus, which currently has born fruit. The i915 Direct Rendering Manager (DRM) driver in Haiku now supports acceleration for all of Intel's HD/Iris chips, which means that getting the best graphics support is more likely on such hardware.

Haiku supports some of the main libraries used by games, like Simple DirectMedia Layer (SDL) versions 1 and 2, PhysicsFS, OpenAL, GLEW, FreeType, and others. If hardware acceleration is not available, Haiku switches to working with OpenGL through Mesa, but only with a software renderer.

Network card support is different. An official Haiku hardware compatibility list contains lots of gray (i.e., unsupported) lines. Network connectivity can be a real showstopper. Even if Haiku boots fine but remains offline, there is little you can do to set up the system, unless you manually download *.hpkg packages on another machine and transfer the files to Haiku, perhaps via a USB drive. However, most Atheros and Broadcom and some of the Intel Pro wireless chips work correctly under Haiku, to a large extent thanks to the merging of the FreeBSD 11 driver to the Haiku code tree in recent months. As for the other types of hardware and peripherals, the overall impression is quite good. For example, Haiku includes the Gutenprint package in its official repository, guaranteeing support for a huge variety of printers, comparable to Linux.

Installation

The Haiku development team uses Buildbot – an automatic tool for building nightly OS images. Downloading and installing a nightly image is preferable for new users, because this offers the latest hardware support and the most recent software additions. For example, all recent nightly builds already have HaikuDepot – a software store and a graphical front end to Haiku's package manager. The aging Alpha R4.1 version doesn't have HaikuDepot, which is why it might not be the best choice for new users.

A Haiku nightly image is about 600MB. The download page offers two flavors: the anyboot image and the raw hard disk image. If you want to burn an image to a CD/DVD or write it to a USB thumb drive with dd or various ISO image writers, choose the anyboot image.

Booting from Haiku installation media is usually very fast. The installer starts with a language selection screen that has two buttons: one for booting to a Live desktop session and another

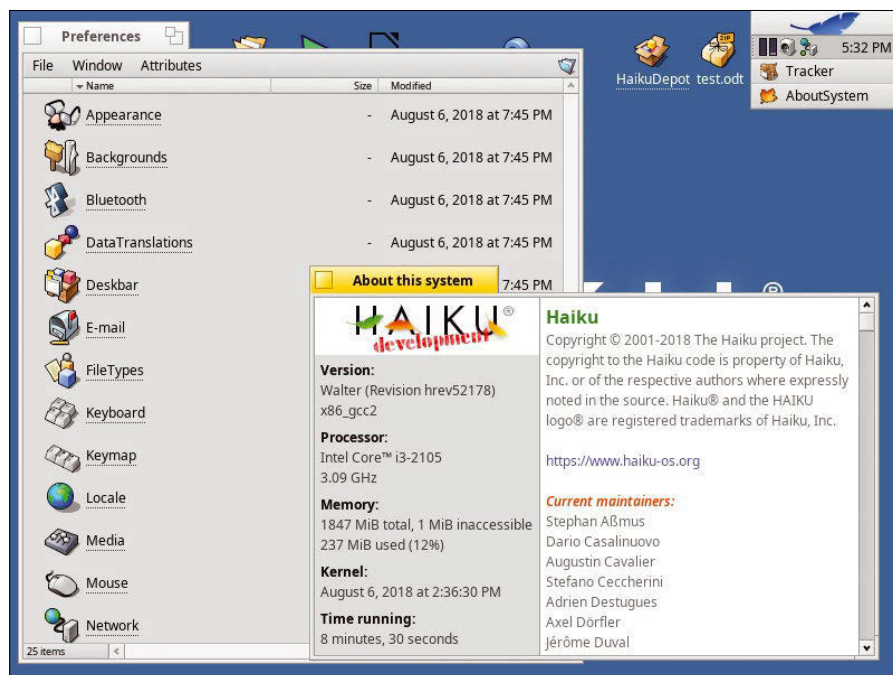


Figure 1: Despite its size, the AboutSystem dialog contains frequently used information about the current Haiku system.

for proceeding with the installation. The live session is a perfect way to see if Haiku has detected all of the machine's hardware, such as a WiFi or an Ethernet card.

If you decide to install Haiku on your hard drive, you only need to complete a few steps with the installation wizard. Choosing a target partition is similar to the macOS installer: You start with no

available targets, invoke the DriveSetup utility (Figure 2), prepare a partition, close the utility, and then select the newly formatted partition as your target. The installer then transfers the core package set to your hard drive, installs a bootloader, and finishes – all steps require just a few minutes. After rebooting the system, you will immediately see the Haiku splash screen with a

progress bar with BeOS-inspired icons. Interestingly, Haiku has been making steady progress in UEFI support since 2016, which means that you'll most likely not have trouble booting a Haiku nightly image in the UEFI mode as well.

The Haiku desktop boots in just a fraction of time compared with an average systemd-based Linux system (which is not slow at all, by the way). The desktop is a faithful copy of BeOS; therefore, it may look a bit archaic at first. Although it is unlike anything you've seen on Linux, the Haiku desktop is tailored for productivity and ease of use. Instead of panels along screen edges, a Deskbar resides at the top of the screen. The Deskbar acts as an all-in-one system tray, applications menu, and task switcher. Other notable features are the desktop with icons and windows with tabs instead of header bars. Most desktop elements cannot be changed, reflecting a design approach similar to macOS.

Getting Acquainted

It takes a little time to get used to Haiku. You can populate your desktop with application launchers by creating links. To do so, open Haiku's file manager, Tracker, navigate to the applications folder, right-click on an app you want to add to the desktop, and select *Create link | Desktop*. By default, Haiku ships with a small set of apps, most of them remakes of the original BeOS titles. For instance, WebPositive, the default basic WebKit-based browser, mimics BeOS' NetPositive. HaikuDepot also offers other more capable WebKit-based browsers, such as Otter Browser (Figure 3) and Qupzilla.

In fact, HaikuDepot is a treasury of apps that brings this OS nearly on par with Linux, or at least very close. HaikuDepot is a pleasure to use. On one hand, it is a robust package manager, similar to Synaptic (Figure 4). On the other hand, its *Show only featured packages* checkbox filters Haiku repositories to a curated list of productivity and multimedia apps – similar to elementaryOS. Additionally, HaikuDepot lets you rate apps with stars. If you don't have a Haiku community account, you can create one right from HaikuDepot. Becoming a Haiku app reviewer could not be easier!

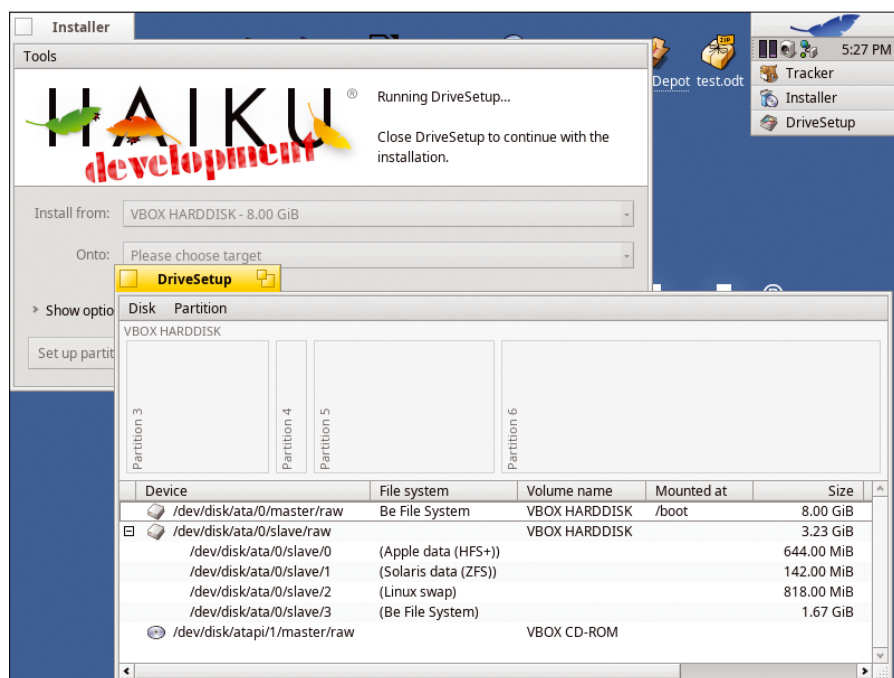


Figure 2: Haiku coexists with many filesystems, including ext3/4, HFS+, ZFS, and some niche systems. The default DriveSetup utility is friendly and functional.

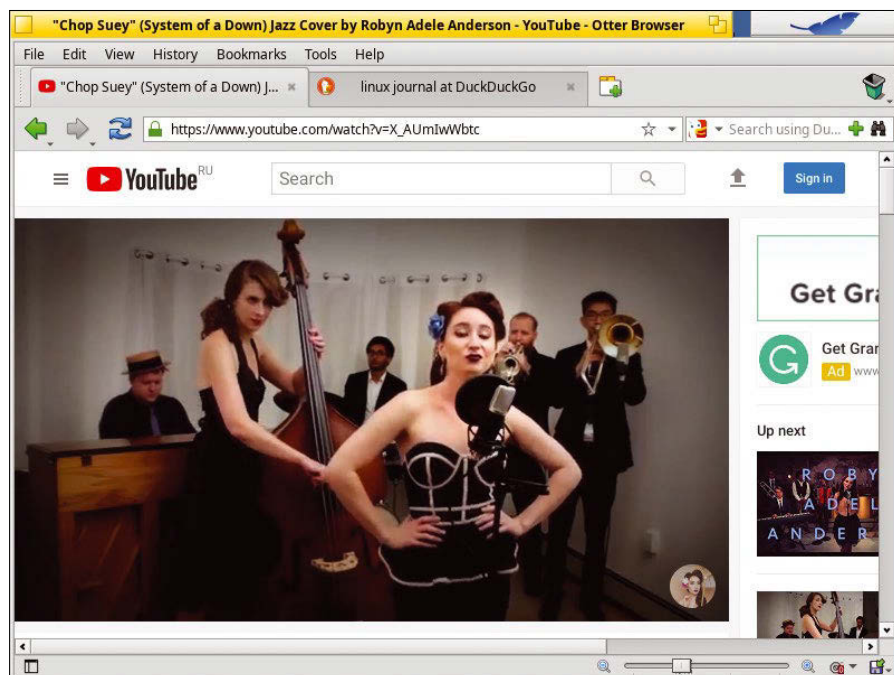


Figure 3: Otter Browser, a modern WebKit-based browser inspired by the Opera 12 UI, runs on Haiku.

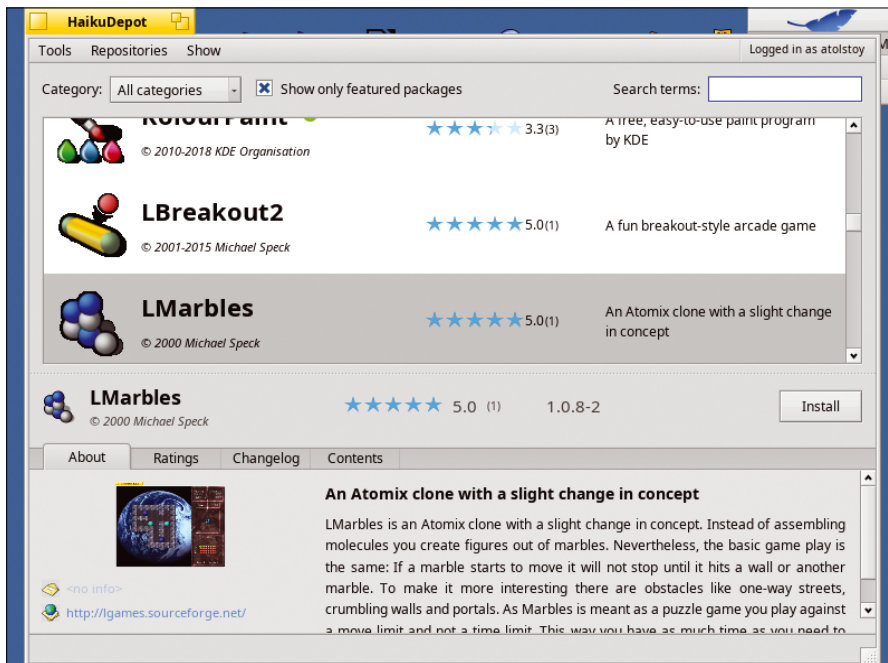


Figure 4: The Haiku Depot app is the easiest way to add open source software your system. Register to become an app reviewer right in the app within just a minute!

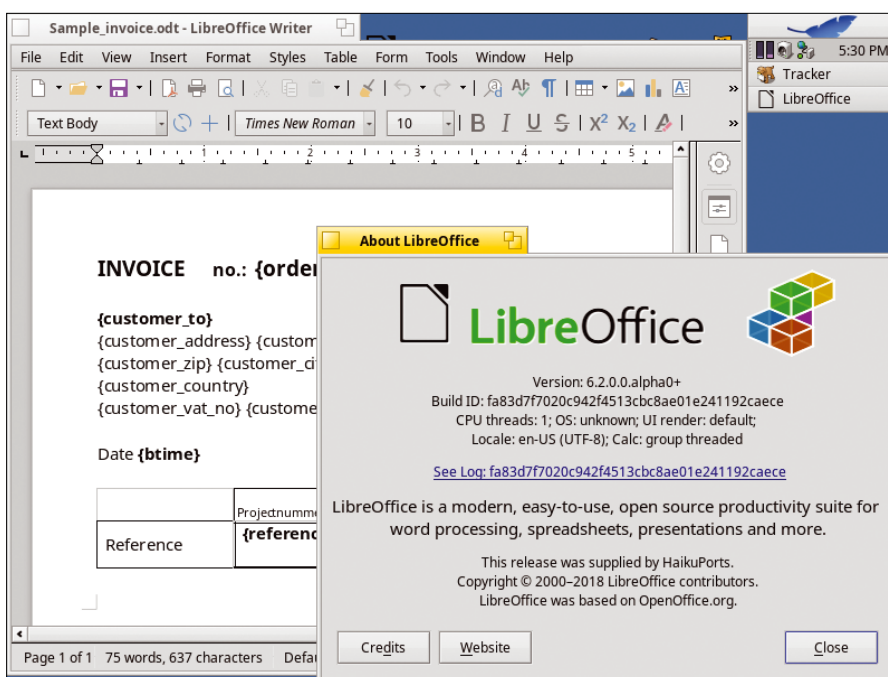


Figure 5: The LibreOffice addition to Haiku makes the OS even more appealing.

2018 has been the year of dramatic improvements to Haiku’s repositories. Now you can install apps, such as LibreOffice 6.x, Calligra Suite 3.1, Krita 4.x, Scribus 1.5, and more. Apart from LibreOffice, you may notice that the rest of this list is Qt-based software. Gtk is not ported yet for Haiku, meaning neither are Gimp, Inkscape, Firefox, Thunderbird, and most other notable Gtk-based

apps. Truth be told, lots of open source apps have yet to be ported to Haiku. Despite this, compared to where Haiku was just a few years ago, the present day Haiku is a vast improvement. In the end, those mentioned Qt5-based apps are often underestimated by end users. A good example is KolourPaint, which has almost everything for image manipulation that an average user needs.

A curious user might wonder how the Haiku team made LibreOffice look like it was using a Gtk VCL plugin in the absence of Gtk itself. The answer is that Haiku, at the time of writing, has successfully built a bleeding-edge KF5 back end of LibreOffice 6.2, so that each app from this office suite uses Qt5-style widgets and looks native in Haiku (Figure 5).

Of course, having the latest versions of open source apps can be a double-edged sword. Although you can easily install these new versions with a few mouse clicks in Haiku Depot, they often have not been tested properly. Even rolling-release Linux distributions, such as Manjaro and openSUSE Tumbleweed, undergo more serious continuous package testing to ensure stability. Haiku is missing that right now. Additionally, the core Haiku system is only approaching beta quality. For a regular user, this can result in sporadic system freezes and lockups. You can also find yourself in Kernel Debugging Land (KDL), Haiku’s built-in kernel debugger, where you can at least gather some useful debug information and try to google it.

Conclusions

Despite the aforementioned caveats, Haiku is still worth a try. Although Haiku is a relatively small project, it stands on the shoulders of BeOS, a perfectly designed, well-tested, and polished operating system, which is why Haiku feels like a finished product. Haiku is perhaps the only non-mainstream OS that offers unexperienced users such a great variety of available software. Furthermore, it is not Unix-based and attracts geeks, tech enthusiasts, and anyone who feels adventurous. ■■■

Info

[1] Haiku website:
<https://www.haiku-os.org>

Author

Alexander Tolstoy is a long-time Linux enthusiast and tech journalist. He never stops exploring hot new open source picks and loves writing reviews, tutorials, and various tips and tricks. He was a lucky Linux fan until his fortune | cousay told him that his own qualities will prevent his advancement in this world. What bad randomness!



Mind-mapping tools

TREE TIME

Mind maps are designed to help display processes and projects clearly in a graphical format. This review explores the design possibilities offered by five mind map programs. *By Erik Bärwaldt*

Structuring complex processes is part of the daily grind in many organizations. To help people understand their decision and thought processes, mind-mapping programs depict process steps graphically. Users can view an onscreen tree view to visualize dependencies.

However, visual clarity depends on the design possibilities offered by the mind-mapping program: Illustrations break up abstract content, and different font attributes emphasize the relevance of indi-

vidual project steps. I tested five mind map programs to see how their design features compare.

CmapTools

CmapTools [6], developed at the Florida Institute for Human and Machine Cognition, is available for several platforms. Its license is proprietary, but the software is free of charge. The website also features a server variant for an on-premises server and an iPad version. You can also use CmapTools in the cloud, which does not require a dedicated application on the client, just a web browser.

CmapTools is a Java application and therefore needs a Java Runtime Environment. The 180MB package is a bit unconventional: You can call the binary package directly after setting the execution rights (`chmod u+x`), and an installation wizard launches. The wizard creates a user-defined directory and unpacks the program files.

Binaries do not automatically end up in the usual `/usr/bin` or `/opt` directories. Instead, you have to put in some manual

work – even if you just want to launch the program from the menu.

The CmapTools graphical user interface (GUI) shuns common conventions: It opens with two independent empty windows. One window contains the work project; the other is used for controlling and configuring the software (Figure 1). The second window also displays the individual projects in a tree view later.

You define your concept map in the right-hand window by first generating a new node with a double-click in the work area and then populating it with text by double-clicking on the four question marks within the node.

You can connect two nodes using the arrow symbols above the source node and dragging the resulting line to the target node. To label the line, click on the question marks that are automatically inserted in the middle of the line.

CmapTools offers a dialog in the *Format | Styles* menu that lets you adapt the graphical appearance. CmapTools accesses the fonts built into your operat-

Not On Board

Mind-mapping programs that have been without a maintainer for some time were not included in this test: the well-known Freemind [1], which last published a new beta release two years ago, and Visual Understanding Environment (VUE) [2], which was updated about three years ago.

Other solutions, such as Lucidchart [3], MindMeister [4], and Mindomo [5], are only available online and are not intended for on-premises use.

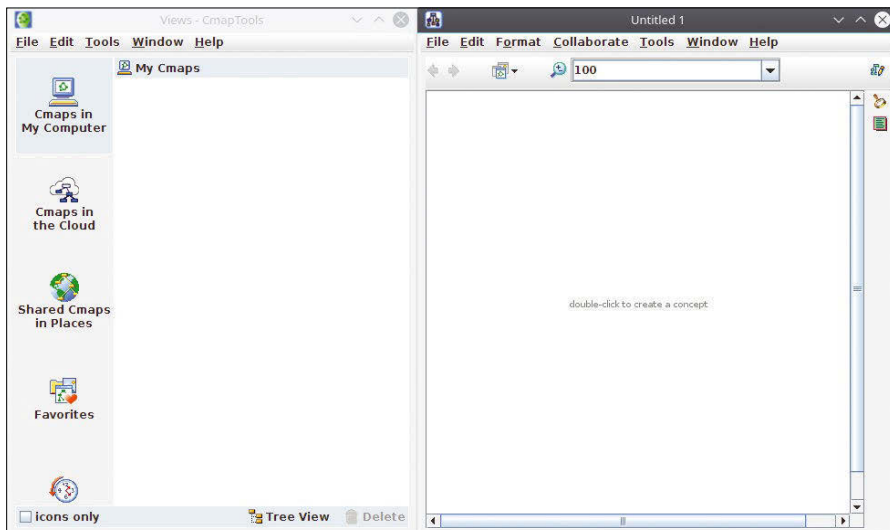


Figure 1: CmapTools stands out because of its unusual appearance.

ing system but does not support all available font sizes. Additionally, the program offers various backgrounds from templates, which you can access via a symbol chart.

The *Cmap* tab lets you colorize the entire view's background and save user-selected sections separately. If you want to combine related groups in a box to visu-

ally highlight them, you can use the nested node function. Select *Tools | Nested Node | Create*. After selecting the area to represent the container, you can edit the appearance of the group of objects combined in the nested node.

For example, you can highlight the nested node with its own background color and adjust the color of the items

inside. You can also add shadows or special fonts, attributes, and font sizes. The nested node acts like a dedicated node: Arrows pointing to it look like normal nodes (Figure 2).

A special feature of CmapTools is the ability to attach multimedia content to nodes. These extensions then appear as small symbols at the node's edge. Clicking the symbol plays the linked media (external applications handle this task). You can add matching links to the content nodes using *Edit | Add & Edit Links to Resources* and *File | Add Web Address*.

In addition to various cloud functions and options designed for collaborative work, CmapTools also supports exporting to various formats: You can export a worksheet to PDF, PS, SVG, XTM/XCM, CXL, and IVML. You can also save a mind map in JPEG format or as a web page.

A printed version of the mind map did not exhibit any quality defects in our lab. Fonts appeared clear and without raster effects, and the program reproduced attributes and frames as shown on the screen.

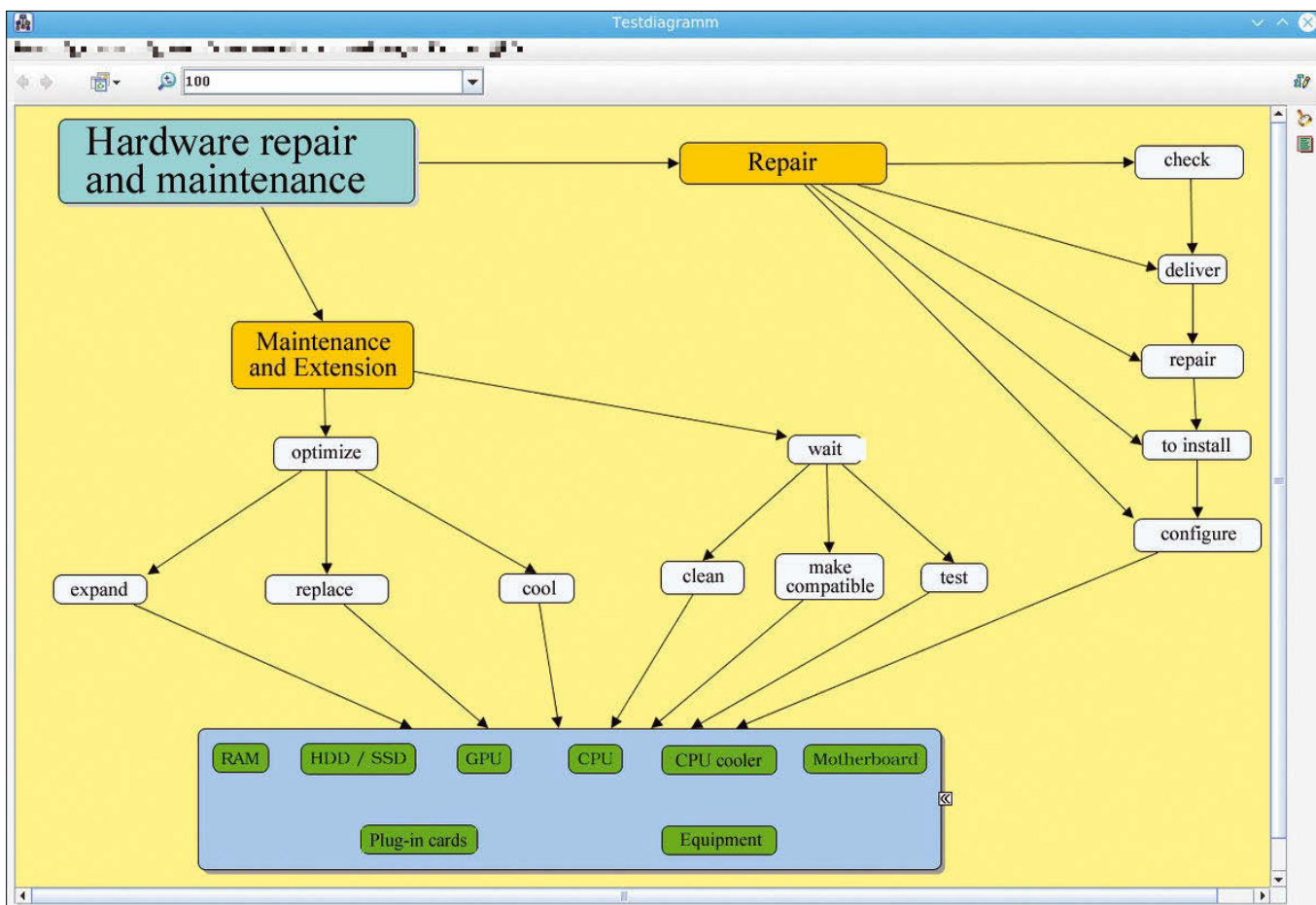


Figure 2: Nested nodes in CmapTools allow a wide variety of design options.

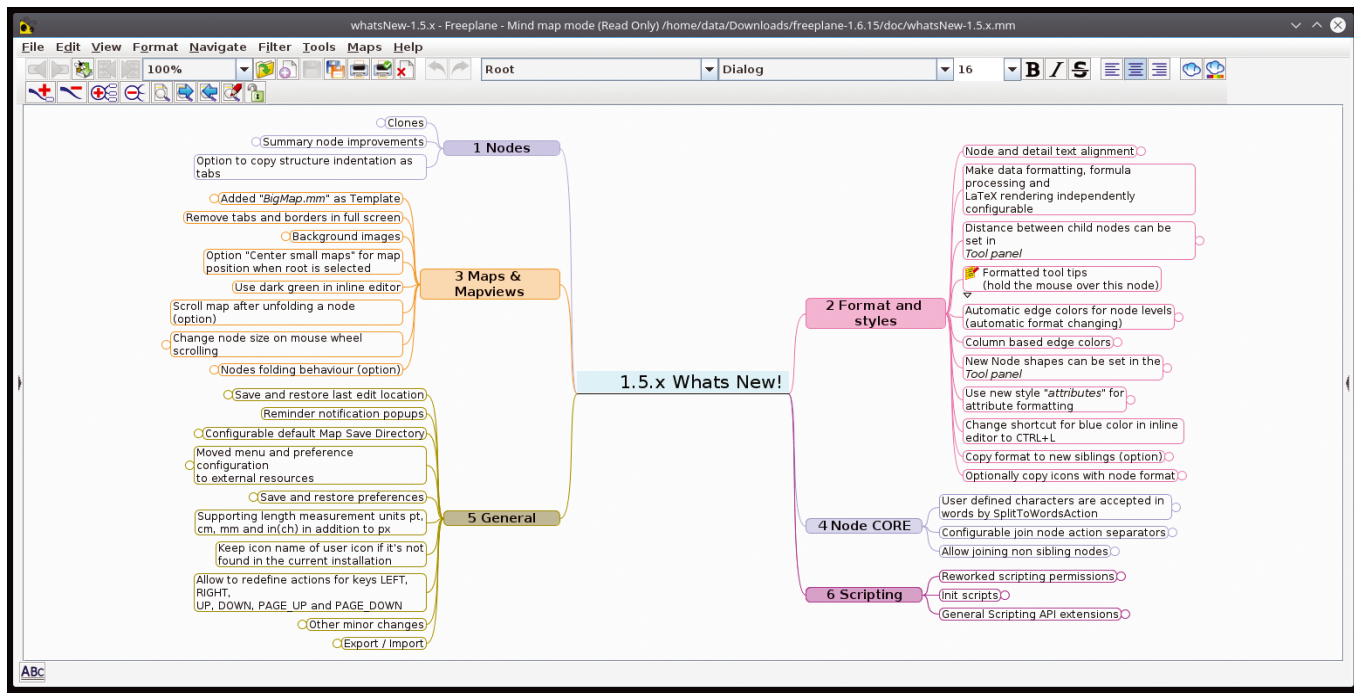


Figure 3: Freeplane shows an example mind map at startup.

Freeplane

Freeplane [7], a fork from the FreeMind project, is another cross-platform, Java-based application available for Linux under the GPLv2 license. Freeplane is desktop-only software; you will search in vain for a server or cloud variant.

You will need to install Freeplane manually in your menu structure. The program comes with a conventional desktop featuring a large empty space for designing and displaying mind maps (Figure 3,) with a menu and buttonbar above for access to important functions.

Use *File | New map* to create a root node. Further nodes are created by pressing the Insert key. Freeplane automatically adds a connecting line to the root node. If you select one of the newly created nodes with a simple left-click and then press Insert, you will create another hierarchy level with a new node starting from the

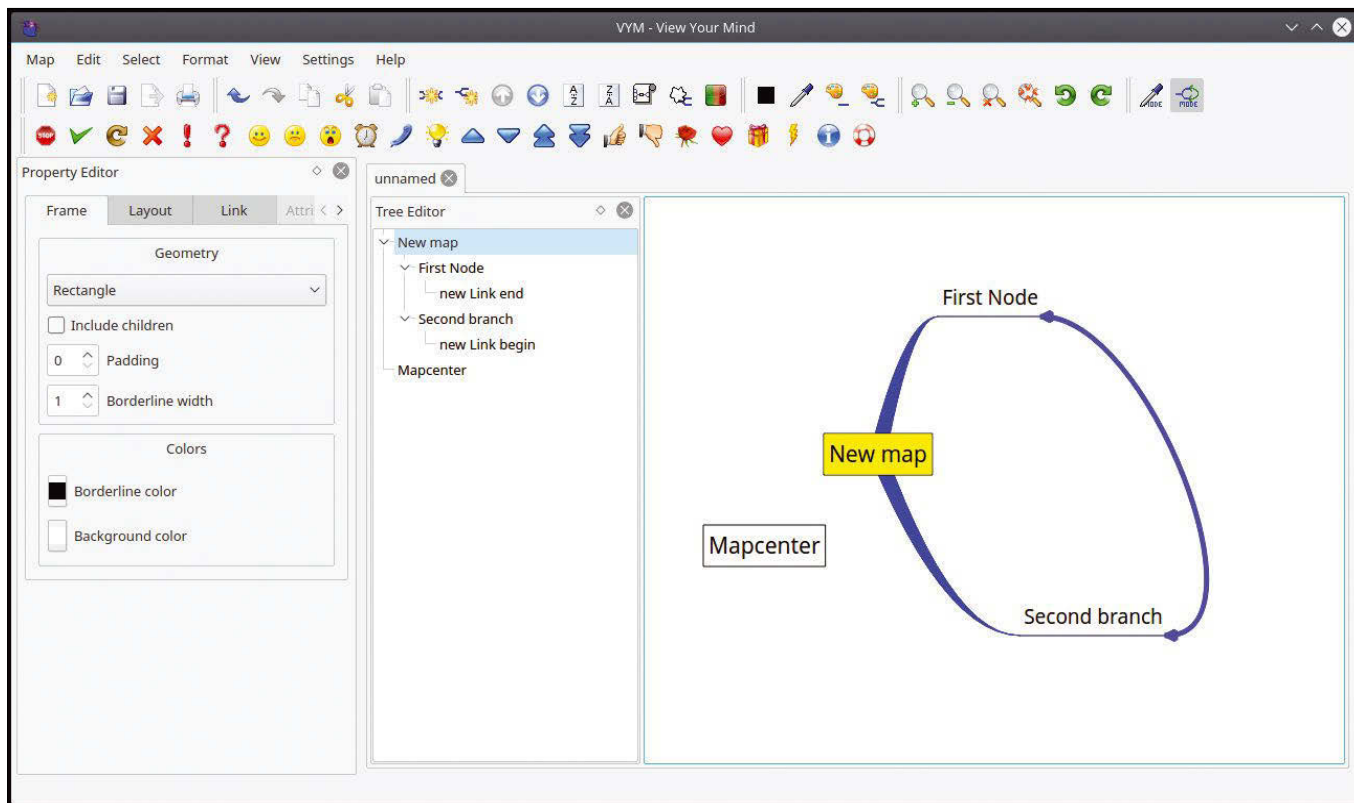


Figure 4: VYM is a solid all-rounder for conventional mind maps.

selected node. In this way, you can add nested hierarchies to your mind map.

If you select a higher level node and press Enter, a node is created at the same hierarchy level. Freeplane automatically draws a connecting line from the higher level in the node hierarchy when you create a new node, which makes it easy to identify individual hierarchies at a glance. If you want to delete nodes, just press the Delete key.

Freeplane positions the individual nodes freely, often resulting in compromised clarity, especially with mind maps containing several hierarchy levels. If you mouse over the node line where it connects to the node, an ellipse appears that allows you to move and realign the node freely on the desktop.

If so desired, you can move a selected node up or down in a hierarchy, or you can drag it directly to another node in the target hierarchy. The target node then shows one of the four sides with gray shading. Just drop the node you want to move to place it.

You can draw additional lines to connect the individual nodes. First,

Ctrl + click to select the two nodes and press Ctrl + L. The program draws a line between the two. If you then right-click anywhere on this line, you can adjust its appearance in the Connector dialog.

By default, Freeplane displays the existing nodes without frames. Only the root node and free-floating nodes have a border. To tell the software to draw borders around nodes within the hierarchies, you need to select the *Format | Node core | Bubble* option.

To make the mind map's graphical representation more sophisticated, the program has an integrated toolbox. You can access this toolbox by right-clicking on the user interface and then selecting the *Tool panel* option in the context menu.

To customize the lines running between the individual nodes, a separate dialog group is available in the *Format | Edge properties* menu. You can change individual lines between selected nodes or in complete hierarchies. In addition to thickness and shape, you can change line color.

Using the *Format | Map background* menu option, you can insert a background image that extends over the entire work area or, alternatively, over the visible area. Freeplane supports images in BMP and SVG formats.

Unfortunately, Freeplane is unable to draw a frame around multiple nodes and then process them as a single node. Thanks to the node cloud around a hierarchy of unconnected nodes, a visually highlighted node group can be created. But you cannot drag connecting lines onto the cloud. It is only possible to establish individual connections between nodes inside and outside the cloud using the selected nodes' context menus.

Freeplane is therefore not totally suitable for visualizing combined, unconnected node groups within a larger mind map.

Freeplane offers numerous export filters for processing mind maps. The filters include the HTML, XML, XHTML, and MediaWiki formats. Freeplane also supports various Microsoft formats, as well as ODT. When exporting to graphic formats, the software supports PDF, JPG,

3,400 PAGES OF



THE COMPLETE LINUXVOICE

ARCHIVE DVD: ALL 32 ISSUES!



Since April 2014, Linux Voice has showcased the very best that Free Software has to offer. Now you can get it all on one searchable DVD.

Includes all 32 issues in EPUB, PDF, and HTML format!

Order now!
shop.linuxnewmedia.com



SVG, and PNG and saves the complete mind map without display errors. The results can therefore easily be post-processed in other applications.

Freeplane does not exhibit any weaknesses when printing. However, it is recommended that you first define the printable areas in the *File | Page setup* dialog. Freeplane typically creates the mind map in landscape format, and you will want to print it out in the same manner.

VYM

View Your Mind (VYM) [8] is a Qt-based program that is perfectly integrated into the KDE desktop environment. The software, which is licensed under the GPLv2, is available from the repositories of almost all recent Linux distributions. VYM, which is being developed at a rapid pace, offers a conventional user interface, but the tree view displayed on the program window's left is somewhat unusual. With the help of the tree view,

you can quickly home in on individual segments in extensive mind maps.

Like Freeplane, VYM offers a menu-bar and a generously dimensioned buttonbar, but they do not offer the same functionality, such as options for modifying the font type and size and arranging the text in the nodes. If necessary, you can customize the buttonbar functionality in the *View | Toolbars* menu.

You can immediately edit the default root node. Right-clicking and selecting the option *Add | Add branch* creates a new hierarchy level with a node. VYM then selects the original node again so that further peer nodes can be created. To create a peer node, just press the *A* key. VYM automatically links newly generated nodes to the parent node with connecting lines, designing them to be as symmetrical as possible to help the user keep track of nodes, even in the case of extensive hierarchy levels.

You can label the nodes by entering text in the individual nodes, which VYM inserts without line breaks. This can lead to confusing layers, especially if you have long node labels. To format the text, press *E*, which opens an editor in the lower part of the window. You can use the editor to enter line spacing to display individual nodes in multiple lines.

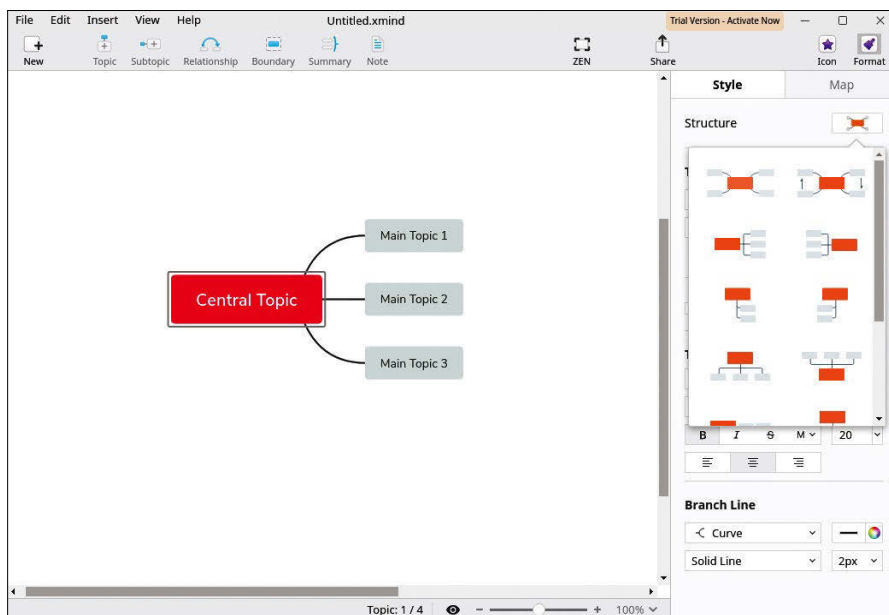


Figure 5: XMind groups hierarchies automatically at first.

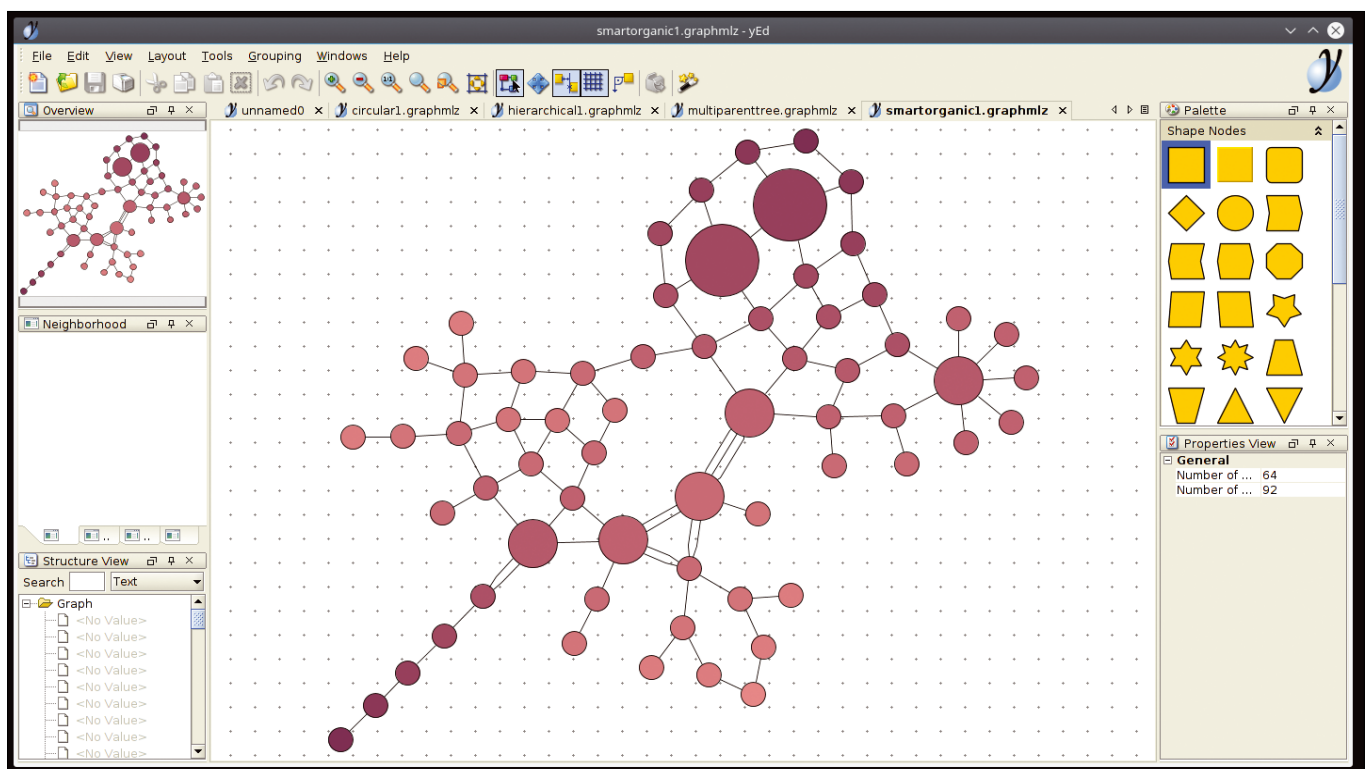


Figure 6: Dot grids and guides in yEd makes it easy to arrange new nodes.

Besides plain text, icons from the toolbar can be used to draw attention to a node. When you click on an icon in the toolbar, VYM then adds the it to the active node.

To change fonts and font sizes, click on the *RichText* button in the editor window. However, the VYM text editor shows some weaknesses. For example, moving text in the editor window may not affect how the node displays. Also, text centering does not work as expected if you then change the font.

VYM offers only rudimentary functions for retroactively drawing lines. You can draw a line – referred to as an xLink by the program – between two nodes that are not directly connected to each other. To do this, select the starting node, press *L*, and draw the line from the start node to the end node while holding down the left mouse button. The dialog defines whether the line is dashed or continuous and at which ends arrowheads appear.

However, because the lines terminate at the same endpoint of a node, this makes the overall presentation very confusing as soon as several lines from the same hierarchy lead to a single node. The recommendation is to draw xLinks from superordinate hierarchies to the target point to reduce the number of lines.

You cannot drag external frames around multiple floating nodes. To create a free-floating node with several entries in VYM, you need to right-click in a free

area of the desktop and choose the *Add Mapcenter* option. In the editor window, enter and format the text, which visually simulates several nodes in a frame. You can position the results freely in the mind map.

To draw frames around individual connected tree hierarchies, select the first hierarchy and then the *Property window* option from the context menu, which opens the Property Editor dialog. There, you can draw various borders using the *Frame* tab and *Include children*, if desired (Figure 4).

VYM, like all popular mind mappers, uses its own format, but exports mind maps to other formats without any problems. The software supports PDF, PNG, and SVG. Additionally, you can export to several text formats, but the formatting is then lost. Many export formats are still considered experimental and should not be used in a production environment.

Mind maps printed without any problems. VYM has a simple print dialog that does not offer a page preview.

XMind

The Java-based XMind [9] is one of the most popular mind-mapping programs, although the Linux version is well hidden on the manufacturer's website [10]. OpenJDK works well with XMind, but you'll need a Linux distribution with the APT package manager to run the installation script. Under distributions with the RPM package manager, the script terminates the installation. You

then need to change to the program directory in the terminal and call the software with `./XMind`.

XMind displays a somewhat unconventional workspace. The mandatory root node appears in the middle after you start a new session.

XMind creates a tree structure with branching sub-nodes starting at the root node. You can create a new node at the same level by pressing *Enter*; pressing *Tab* creates a new hierarchy.

XMind adds connecting lines to the individual nodes and generates them using preset format styles. When creating the mind map, XMind displays a format area with several options. The *Format* icon at top right pulls out groups on the right that contain *Style* and *Map* options. For example, the *Structure* selection area determines the general type of mind map and thus its appearance (Figure 5).

To change node groups, you do not have to select them individually (by *Ctrl* + clicking on the nodes), but you can select nodes in a hierarchy level or several levels by dragging a frame around them.

You can generate unconnected nodes by double-clicking on the free area of the desktop and then moving them freely in the workspace.

For additional information concerning a group of nodes, XMind lets you add labeled brackets to the sides of nodes. After selecting the nodes, press the *Summary* button in the bar at the top and

Table 1: Overview of Design and Print Functions

	CmapTools	Freeplane	VYM	XMind	yEd
License	Proprietary	GNU GPLv2	GNU GPLv2	LGPLv3/EPLv1	Proprietary
Platform	Java	Java	Qt5	Java	Java
Node hierarchies	Yes	Yes	Yes	Yes	Yes
Free nodes	Yes	Yes	Yes	Yes	Yes
Formatting Options					
Variable font sizes	Yes	Yes	Yes	Yes	Yes
Font attributes	Yes	Yes	Yes	Yes	Yes
Changeable fonts	Yes	Yes	Yes	Yes	Yes
Node frame	Yes	Yes	Yes	Yes	Yes
Node backgrounds	Yes	Yes	Yes	Yes	Yes
Attachments (notes, images)	Yes	Restricted	Yes	Yes	Yes
Line formatting	Yes	Yes	Yes	Yes	Yes
Output					
Fold-away segments	Yes	Yes	Yes	Yes	Yes
Print dialog with CUPS	Yes	Yes	Yes	Yes	Yes
File export	Yes	Yes	Yes	Yes	Yes
Own file format	Yes	Yes	Yes	Yes	Yes

enter an explanatory text in the text box next to the parentheses.

XMind also offers a way to add images, notes, and links to nodes. The *Hyperlink* context menu option lets you specify a URL in a separate dialog that places a small link icon in the node. A click on this globe icon opens the linked website in the browser.

The *Notes* option lets you enter a message in an overlay window. After completion, a small clickable notepad icon appears in the node.

You can easily create additional connecting lines in XMind by selecting the starting node and then clicking on *Relationship* in the buttonbar. Now draw a connecting line from the starting node to the target node while holding down the left mouse button. Right-clicking on the line and selecting the *Properties* option in the pop-up context menu displays the Relationship dialog on the right side of the program window, which you can use to change the line's appearance.

Like most mind-mapping tools, XMind can add background colors to nodes or worksheets. The easiest way to do this is to select the required nodes first and then the *Properties* option from the context menu. In the Format pane, the *Style* tab offers several styles with background colors and borders. Clicking on one of them enables it for all the selected nodes.

XMind exports worksheets to various external text and image formats under *File | Export To*. Options marked *[Pro]* are only available in the professional version (e.g., the PDF format).

The mind map, both on screen and output from the printer is accurate and clear. Third-party applications can easily open exported files. The software relies on the CUPS system for printing. A dialog lets you insert headers and footers into the mind map before printing.

yEd

yEd [11], a mind map program developed by yWorks, also uses Java and is available across platforms. A Linux installation script is available for download from the developer's website. Although the 64-bit version comes with a current Java 10 Runtime Environment, the 32-bit version requires a Java 8 Runtime Environment on the installation system [12].

If you select an empty document on the welcome screen, yEd shows you a rather unconventional program window. On the right and left, control elements frame a work surface. The left window segments are used for overview and navigation, and the right side gives you immediate access to graphical elements and properties, such as different node types, line types, and node content settings.

yEd lets you to develop a mind map by dragging individual elements from the *Palette* section without having to arrange the nodes in any way. Various settings for the visual appearance of the labels and nodes are available under *Properties* in the upper-right corner.

For a defined layout, you can select one of the numerous templates from the *Layout* pop-out at upper left. Selecting the *Grid* button in the buttonbar, lets you add a dot grid to the input area, which makes it easier to arrange nodes symmetrically (Figure 6).

yEd also enables a line grid for freehand mind maps. When you create a node, yEd shows you lines to neighboring nodes if there are horizontal or vertical matches. These temporary line grids help to position new nodes accurately below or above existing nodes – even for horizontally arranged nodes.

yEd offers different setting options for the individual nodes, as well as for line shapes and thicknesses, that help you group the nodes visually. You can also label connecting lines and supplement them with arrowheads to indicate the direction. To adjust the node size, draw a thin frame around active nodes with handles at the corners.

yEd's one-click function, which is still tagged as experimental, automatically arranges mind maps or mind map segments. A special algorithm tries to sort the nodes in the different hierarchies as meaningfully as possible. This feature worked surprisingly well in the test: I created a hierarchical representation without creating the partly overlapping connecting lines that often make a mind map unclear.

Like the other mind-mapping programs, yEd has an export function. The freeware program offers options for various graphic formats, including BMP, JPG, GIF, PNG, and SVG in the *File | Export* menu. yEd also exports high-quality EPS and HTML, so it is easy to integrate

mind maps with other applications. Office formats of any kind are missing.

When it comes to printing, the yEd print dialog lets you add a header and a footer. You can adjust the size of the mind map to match the print area and paper size.

Conclusions


The mind-mapping programs examined in this article all help to visualize projects and processes clearly (see Table 1). The differences are in the details: CmapTools and yEd are by far the most suitable for freehand mind maps that do not require a – typically hierarchical – layout form. The other test candidates have problems drawing frames or freely arranging connecting lines.

VYM is well suited for simpler mind maps; XMind and CmapTools successfully integrate notes and win hands down when it comes to integrating multimedia content. Freeplane, on the other hand, is a solid all-rounder for daily use.

Because all the candidates have export functions, nothing stands in the way of integrating the mind maps into other applications. It is a little annoying that each program has its own storage format, and the formats are not mutually compatible. The candidates don't allow importing from other mind map programs, so if you create a mind map in one program, you will need the same program to edit it. ■■■

Info

- [1] Freemind: http://freemind.sourceforge.net/wiki/index.php/Main_Page
- [2] VUE: <http://vue.tufts.edu>
- [3] Lucidchart: <https://www.lucidchart.com>
- [4] MindMeister: <https://www.mindmeister.com/de> [in German]
- [5] Mindomo: <https://www.mindomo.com>
- [6] CmapTools: <https://cmap.ihmc.us/cmaptools/>
- [7] Freeplane: <https://www.freeplane.org>
- [8] VYM: <https://sourceforge.net/projects/vym/>
- [9] XMind: <https://www.xmind.net>
- [10] XMind for Linux: <https://www.xmind.net/download/linux/>
- [11] yEd: <https://www.yworks.com/products/yed>
- [12] yEd 32-bit version: <https://www.yworks.com/downloads#yEd>



OSMC

MEET THE
MONITORING
EXPERTS

PROGRAM ONLINE

Register now

osmc.de



Managing software development projects with Git

GIT IT RIGHT

The Git version control system is a powerful tool for managing large and small software development projects. We'll show you how to get started. *By Roman Jordan*

With its egalitarian spirit and tradition of strong community involvement, open source development doesn't scale very well without some form of version control.

Over the past several years, Git [1] has settled in as the most viable and visible

version control tool for the Linux space. Git was created by Linus Torvalds himself, and it got its start as the version control system for the Linux kernel development community (see the box entitled "The Birth of Git"). Since then, Git has been adopted by hundreds of open source projects and is the featured tool

on several large code-hosting sites, such as GitHub.

Even if you aren't a professional developer, if you work in the Linux space, you occasionally need to download and compile source code, and, more often than not, that means interacting with Git.

Many Linux users pick up occasional Git

The Birth of Git

The Git version control system was born in an unexpected whirlwind of development back in the Spring and Summer of 2005. At the time, the Linux kernel developers were using the BitKeeper version control system. Unlike many version control tools of the time, BitKeeper had a distributed architecture, which worked well for the remote development process practiced within the dispersed kernel community.

The ever-practical Linus adopted BitKeeper because he saw it as the best available solution for the Linux team, however, BitKeeper came with some complications. The biggest issue was that, although BitKeeper was provided free of charge to the kernel team in what was called a "community" edition, it didn't have a free (as in freedom) open source license. Several restrictions were placed on using the software, including a prohibition against reverse engineering and creating unauthorized extensions. Several leading free software developers, such as Richard Stallman and Alan Cox, objected to a high-profile open source product like Linux using a non-free code management system, but the kernel community continued to use BitKeeper in an uneasy truce.

Then in 2005, word got out that Andrew Tridgell, creator of Samba, who was, at the time, an employee of the Open Source Development Labs (OSDL), which oversees Linux kernel development, had reverse-engineered the proprietary BitKeeper protocols

and developed a free client, thus violating the BitKeeper license. BitMover, the company that owned and maintained BitKeeper, acted swiftly to cancel the community license, thus suddenly ending the kernel team's access to BitKeeper version control.

Linus already knew he didn't like any of the other code management systems available at the time, so when he lost access to BitKeeper, he started to build his own. The Git project was officially announced on April 6, 2005, and by June 16, it was already operational enough to manage the Linux kernel 2.6.12 release [2]. Linus passed the Git maintainership to Junio Hamano in July 2005, and Hamano has continued to develop and improve Git ever since. In an interview with TechCrunch in 2012, Linus said that one of his biggest successes was "...recognizing how good a developer Junio Hamano was on Git, and trusting him enough to just ask if he would be willing to maintain the project." [3]

Git has seen many improvements since 2005, thanks to Hamano and others within the Git development community, but the rapid rise of Git from the ashes of the BitKeeper debacle, and its gradual emergence as the world's most popular version control framework, have added another chapter to the legend of Linus Torvalds as a genius software developer. Linus isn't just the creator of Linux; he is creator of both Linux and Git – two incredibly successful tools that are in use all over the world.

Lead image © Stephen Rees, 123rf.com

commands on the fly without ever getting a formal introduction to what Git is and how it works. This article is the first in a two-part series aimed at building a better understanding of Git for everyday Linux users. This first article shows how to install Git, create a Git project, commit changes, and clone the repository to a remote location. Next month, you'll learn some advanced techniques for managing code in Git.

Git makes it easy to manage different versions of files side by side. The software is decentralized, and a server connection is only required for synchronization. Daily work is handled locally, which leads to much better performance. And after more than ten years of active development, Git is surprisingly easy to use, even for beginners.

Before You Start

Git is included with almost all distributions. On Red Hat systems, you can

A Question of Settings

Git saves the settings as a function of the option specified in the `config` command. You have three options: `--system`, `--global`, and `--local`. The setting determines the storage location for the configuration. The configuration files are evaluated in the opposite direction, starting with the local, project-specific data. The documentation shows the specification of global, i.e., user-specific configuration data. The software stores the information tagged as `--global` in the `.gitconfig` file in the user's home directory. See the man page `man git-config` or the help with `git help config` for more information on configuration options.

complete the install with `sudo yum install git`, and on Ubuntu, enter

```
sudo apt install git
```

You first need to configure a name and an email address. Without this information, Git either issues a warning or generates a dummy name and address. For the user John Doe with the email address `john.doe@example.org`, the process is shown in Listing 1.

The `git config --list` command displays the settings. You can change these settings at any time. Git uses a multi-level structure for these settings (see the box entitled "A Question of Settings").

Let's Go

The project in this example is located in the `~/mproject` directory and consists of the text files `readme.txt` and `project.txt`. The commands from Listing 2 create the project, but they do not yet create a repository.

Creating the actual repository requires three steps (Listing 3): First, initialize

the project in the main directory, in this case `~/mproject`, then register the files it contains, and transfer them to the Git database, the repository.

The `git init` command creates an empty repository in the `.git` subdirectory (a hidden directory). It doesn't matter whether or not the files are in the actual directory, Git does not take project-specific data into account.

Use `git add` to add a file to the index. The current version of the file is now in the staging area. It contains versions that are flagged for the next commit. Or in Git speak: The file is staged.

The `git commit` command adds the marked files to the repository. This process is known as a commit or check-in. Each commit has its own text. Git displays the first line of the text in various outputs; it should therefore be short and as precise as possible. You can then describe the commit in detail, separated by an empty line.

You pass this text to the software with the `-m` option. Without this option, Git

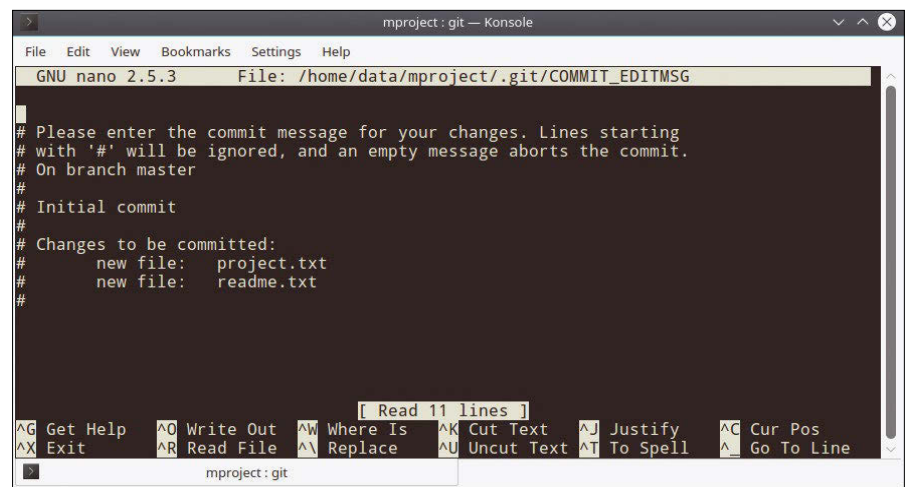


Figure 1: A commit without specifying text brings up the default editor.

Listing 1: Name and Address

```
$ git config --global user.name "John Doe"
$ git config --global user.email john.doe@example.org
$ git config --list
user.name=John Doe
user.email=john.doe@example.org
```

Listing 2: Creating a Project

```
$ cd
$ mkdir mproject
$ cd project
$ echo "readme.txt file" > readme.txt
$ echo "file project.txt" > project.txt
```

Listing 3: Creating a Repository

```
$ cd ~/project
$ git init
Empty git repository initialized in /home/john/mproject/.git/

$ git add readme.txt
$ git add project.txt

$ git commit -m "First Commit"
[master (Basic-Commit) 77558e4] First Commit
2 files changed, 2 insertions(+)
create mode 100644 readme.txt
create mode 100644 project.txt
```

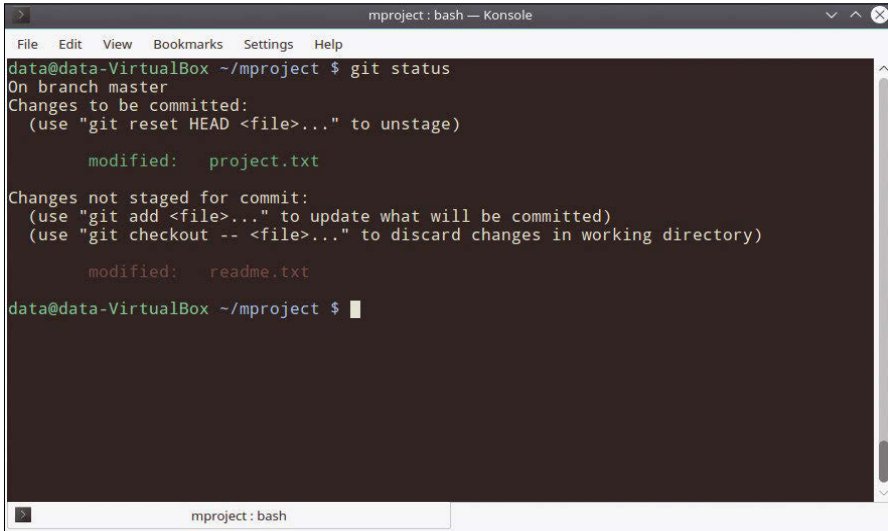


Figure 2: Two versions of a file and possibly instructions for working with them.

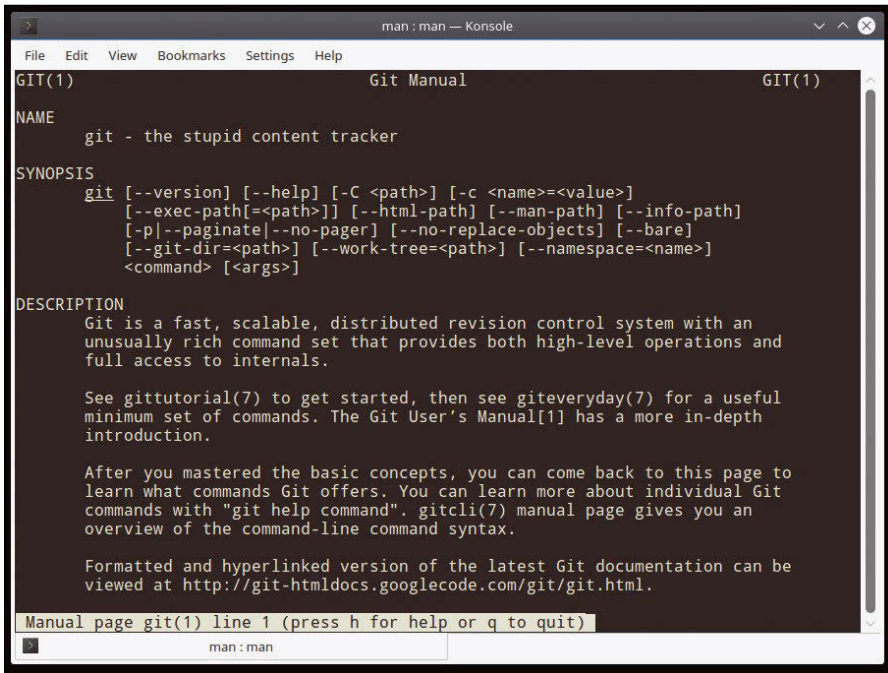


Figure 3: The `man git` man page gives you a quick overview of git commands.

starts the default editor, unless configured otherwise (Figure 1). If no entry exists, Git aborts the commit.

Properly Managed

A version control system (VCS) manages versions, or, more precisely, versions of

files. A version is created when you add a file to a project or edit a file already contained in the project. Using this system, you define the version of a project, such as a program or a web page.

A VCS logs who made what changes, when, and why. The log

Table 1: All Cases Covered

Call	Content
<code>man gittutorial</code>	Git-based project flow
<code>man giteveryday</code>	Frequently used commands, including examples (Fedora)
<code>man gitcore-tutorial</code>	Procedure in detail; partly using outdated commands
<code>man git</code>	General manual

makes it possible to trace the changes, compare different versions, and restore previous versions. It also displays the changes that project members have made in parallel.

The software manages the files in a repository, or repo for short, which is basically a directory [4]. Since the work is usually done on a copy, and the original is typically located in another directory (ideally on a different physical medium); a kind of backup is automatically created.

If several people work on a project, the use of a VCS is actually obligatory. Synchronization quickly becomes a problem without it.

Versions

A Git-versioned project keeps three versions of a file. The version in the working directory is the one you work on. Once the file has reached a state that you want to keep, transfer it to the staging area using `git add file` and continue working on the version in the working directory.

You can repeat this process as often as you like. However, you always overwrite the previous version in the staging area. There is exactly one version for each file in the staging area. Any following commits adopt this latest version. The version in the working directory is irrelevant.

Figure 2 shows two different versions of the file `project.txt`, one in the staging area and a second in the working directory. The repository contains the third version.

Git sometimes gives hints when executing some commands. The hints often refer to how you undo a particular action.

Help

In addition to the general manual (`man git`), the installation comes with several specific manuals (see Table 1). If you call `man git`, you will find an overview of the subcommands, including a short

Table 2: Getting Started

Command	Function
<code>init</code>	Create or initialize empty repository
<code>add</code>	Add files to the staging area (basis for a commit)
<code>commit</code>	Transfer staging area versions to repository
<code>status</code>	Request status of files in working directory

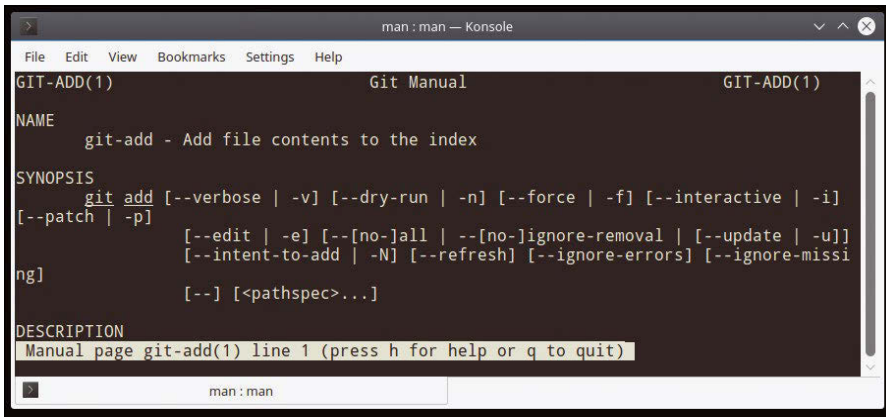


Figure 4: In addition to a page with general information, many distributions also have pages explaining the various subcommands.

description in the GIT COMMANDS section (Figure 3).

Further documentation is located in `/usr/share/doc/git`. The scope of the documentation depends on the distribution. Fedora comes with a manual for users, `user-manual.html`, and a how-to, `howto-index.html`.

Listing 4: Completion and Help

```
01 $ git a
02 add  am  annotate  apply  archive
03 $ git --help
04 Use: git ... <command> [<args>]
```

Listing 5: Project File Status

```
$ echo "new line" >> project.txt

$ git status
On Branch master
Changes not flagged for commit:
  (use "git add <File>..." to flag the changes for the commit)
  (use "git checkout -- <File>..." to discard the changes in the working directory)

        changed:      project.txt

no changes are flagged for commit (use "git add" and/or "git commit -a")

$ git add project.txt

$ git status
On Branch master
changes flagged for commit:
  (Use "git reset HEAD <File>..." to remove from the staging area)

        changed:      project.txt

$ git commit -m "new line inserted"
[master 9d71c8d] new line inserted
1 file changed, 1 insertion(+)
```

from the overview of the Git commands by task.

Continuing with the Project

The `project.txt` file changes as the project progresses. You copy and save new versions with the commands `add` and `commit`. The `git status` command shows the status of the files in the working directory. Listing 5 shows the status of the file after the `git add project.txt` command switches from changes that are not flagged for a commit to changes that are flagged for commit.

The `git add` command lets users specify patterns for files and directories and other options. You can use `git add -u` to move all modified files entered in the index into the staging area. Table 2 shows the commands used so far.

Where to Next?

Git is now managing the project, but what is it actually doing? Let's look at the history first, which you can see with `git log`. The excerpt from Listing 6 shows a project with two commits, which corresponds to two versions.

Each commit is identified by a 40-digit SHA1 hash, which I will simply refer to as the hash. The hash is used for unique identification and as a checksum. For some commands, it is possible to specify the hash as a parameter; the first 8 to 10 digits are often sufficient.

The `git log 77558e4ac` command will only output the log messages up to the specified commit. In the terminal, you can copy and paste the hash with the mouse by double-clicking the hash with the left mouse button and then pasting it again with a single click on the middle mouse button.

Table 3 contains some commands, including possible options for handling the versioned data. The commands include a multitude of options.

The `git difftool` command behaves like `git diff` but starts an external program (Figure 5). Use the command

```
git config --global diff.tool Program_name
```

to define the external program if required.

Remote Repository

So far, the project consists only of a local repository in the project direc-

Listing 6: Two Commits

```
$ git log
commit 9d71c8dd00db5bfb7e21ac8884356d0af284b1b8 (HEAD -> master)
Author: John Doe <john.doe@muster.de>
Date:   Fri May 11 15:22:13 2018 +0200

    new line inserted

commit e29f38d1bc7625090
Author: John Doe <john.doe@muster.de>
Date:   Thu May 10 09:35:53 2018 +0200

    First commit
```

Table 3: Extended Commands

Command	Function
log	Display versions including hash for identification
diff	Display differences between working directory and staging area
diff --staged	Display differences between staging area and last commit
diff hash	Display differences between working directory and commit hash
diff hash1 hash2	Display differences between the specified commits
reset HEAD	Remove files from the staging area
reset --hard	Reset files in the working directory to checked-in state
checkout	Overwrite file in working directory with content from last commit
checkout hash	Check out all files of the specified version. (Note: you cannot modify versions that have already been checked in)

Listing 7: Remote Repository

```
$ cd ~/mproject/..
$ git clone --bare mproject mproject.git
Clone in bare repository 'mproject.git' ...

$ mv mproject.git gitrepo

$ cd
$ mkdir gitrepo

$ mv mproject/ mproject_old

$ cd
$ git clone /home/john/gitrepo/mproject.git
Clone to 'mproject' ...

$ cd mproject

$ git remote show origin
* Remote repository origin
URL for retrieval: /home/john/gitrepo/project.git
URL for sending: /home/john/gitrepo/project.git
Main branch: master
Remote branch:
  master followed
Local branch configured for 'git pull':
  master merges with remote branch master
Local reference configured for 'git push':
  master sent to master (current)
```

tory. However, a typical project usually comes from a remote repository.

To create a remote repository, first create a bare repository from the local data. Unlike the local repo, this does not contain a working directory.

You then have the option to move the bare repository to a corresponding directory, ~/gitrepo in Listing 7. You can then rename or delete the existing project directory. Simply clone the newly created remote repository, and Git creates the subdirectory.

From now on, you will be editing the project in the newly created mproject directory. The local repository is connected to the remote gitrepo/mproject repository. The git push command transfers the data from the local directory to the remote directory.

Conclusions

Using Git is quite simple, even without prior knowledge of version control systems. You can complete your daily work efficiently at the command line with just a few commands. And since Git usually saves the changes locally, commands execute quickly. ■■■

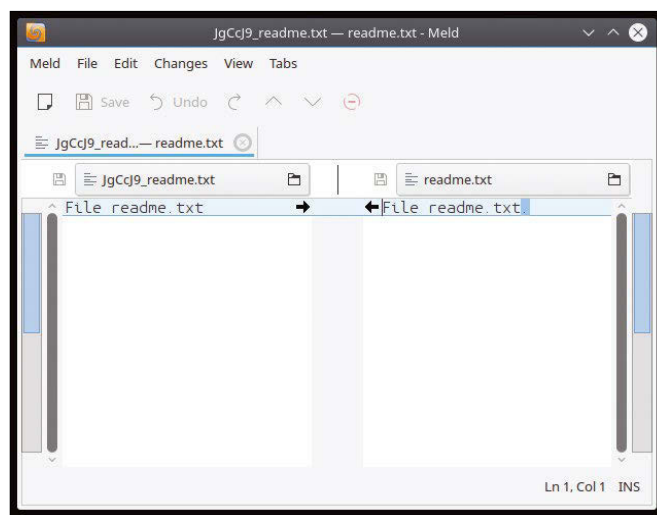
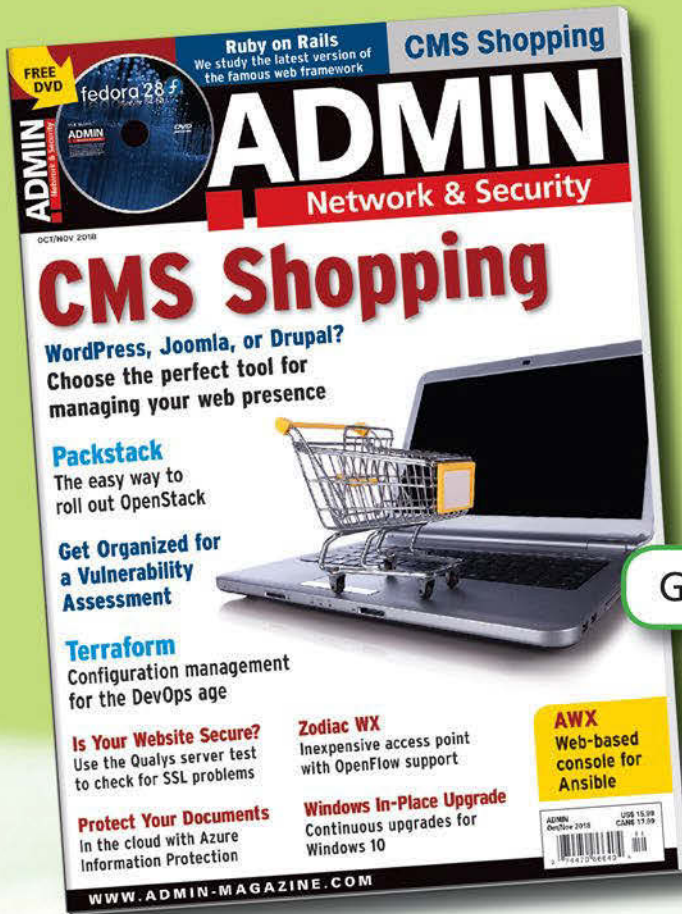


Figure 5: The git difftool command calls an external program to view the differences between files, in this case, the Meld visual diff and merge tool.

Info

- [1] Git website: <https://git-scm.com>
- [2] Wikipedia on Git: <https://en.wikipedia.org/wiki/Git>
- [3] An Interview with Linus Torvalds: <https://techcrunch.com/2012/04/19/an-interview-with-millennium-technology-prize-finalist-linus-torvalds/>
- [4] Wikipedia definition of a software repository: https://en.wikipedia.org/wiki/Software_repository

REAL SOLUTIONS *for* REAL NETWORKS



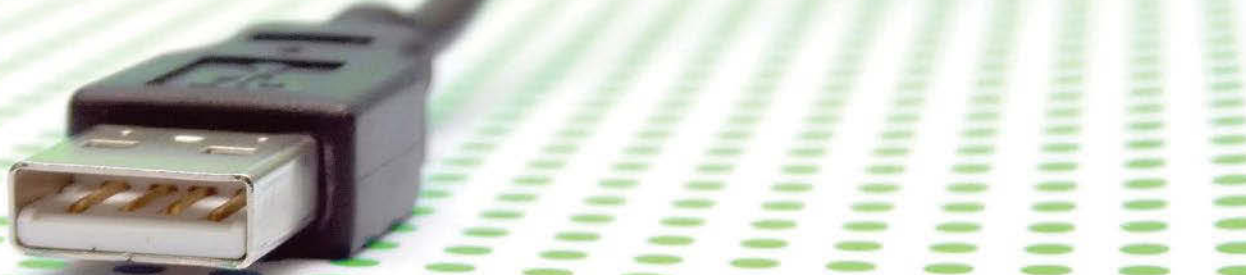
THE NEW IT

ADMIN – your source for technical solutions to real-world problems.

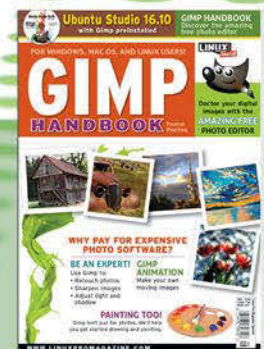
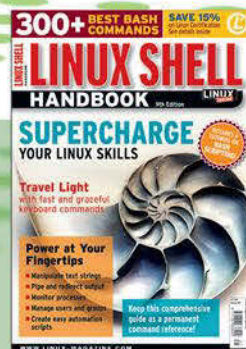
SUBSCRIBE NOW!

shop.linuxnewmedia.com

GET IT FAST with a digital subscription!



Check out our full catalog:
shop.linuxnewmedia.com





Renaming files at the command line

BULK RENAMING

When it comes to renaming multiple files, the command line offers time-saving options in the form of `mv`, `rename`, and `mmv`. *By Bruce Byfield*

Modern computers are full of collections of photos, music, files, and ebooks. These collections increase the need for efficient file management, including the renaming of files. Often, users want to show by naming conventions that a set of files belong together, or, in the case of music files, to ensure that songs are played in a certain order. Similarly, projects may want to indicate file contributors or revision numbers. Fortunately, the Bash shell includes three standard commands for adjusting file names: `mv`, `rename`, and `mmv`.

Desktops routinely include a file manager, in which a file's name can be altered in the same dialog window as its

Author

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

other properties. However this feature is designed for renaming a single file in the simplest of circumstances. Even more importantly, as I found while digitizing music ripped from cassettes, when you work with multiple files, opening and closing the file properties dialog a dozen or more times becomes tedious after a few repetitions. Almost always, it is far easier to open a command line.

At the prompt, you could use a script, especially if you are doing a routine operation like a dated backup. However, even easier is to place all the files to be renamed in the same folder and use their path and a regular expression – probably an asterisk (*) – to rename the files all at once. You do have to be careful that a typo does not leave your file names in confusion, but you can use some of the built-in safeguards and do in a couple of minutes what might take you 10 minutes from the desktop.

People Gotta mv

`mv` [1] is short for “move,” but its inclusion here makes sense when you realize that to move a file is to change at least

part of its path – which is also its name. The command is especially useful for simple backups, and the command is about what you might expect:

```
mv SOURCE-PATH TARGET-PATH
```

However, what you might not expect is that, by default, the original file is deleted. If you want to copy the original file, add the option `--no-clobber` (`-n`) directly after the basic command. The target, of course, can be the same directory, with only the file name being different.

If you are making a backup, you can streamline the process by using `--update` (`-u`), so that only files not already in the target directory are added. You might also use `-stop-trailing-slashes` to delete automatically any unnecessary slashes at the end of the path, especially if you are copying and pasting. You might also want to add `--suffix=SUFFIX` to change the extension, perhaps to `BK` to identify the backup copy.

These can be useful features for renaming a single file or, with a simple regular expression like an asterisk (*),


```
bb@nanday:~/files$ mv -v ./*.pdf ./backup/
'./FINAL-Complaint-Reitman-v.-Ronell-and-NYU.pdf' -> './backup/FINAL-Complaint-Reitman-v.-Ronell-and-NYU.pdf'
'./guidelines-for-submission-format-march-31-2017.pdf' -> './backup/guidelines-for-submission-format-march-31-2017.pdf'
'./International_outgoing_wire_request.pdf' -> './backup/International_outgoing_wire_request.pdf'
'./Intuos4-Users-Manual.pdf' -> './backup/Intuos4-Users-Manual.pdf'
bb@nanday:~/files$ █
```

Figure 1: Running `mv` in verbose mode shows the command's limited options.

```
bb@nanday:~/files$ rename -v 'y/A-Z/a-z/' ./*
./BACKUP renamed as ./backup
./FINAL-COMPLAINT-REITMAN-V.-RONELL-AND-NYU.PDF renamed as ./final-complaint-reitman-v.-ronell-and-nyu.pdf
./GUIDELINES-FOR-SUBMISSION-FORMAT-MARCH-31-2017.PDF renamed as ./guidelines-for-submission-format-march-31-2017.pdf
./INTERNATIONAL_OUTGOING_WIRE_REQUEST.PDF renamed as ./international_outgoing_wire_request.pdf
./INTUOS4-USERS-MANUAL.PDF renamed as ./intuos4-users-manual.pdf
```

Figure 2: `rename` uses Perl expressions, allowing for a variety of ways to rename files. Shown here is a range of characters to be renamed.

for moving a group of files to another directory. Otherwise, `mv`'s ability to manipulate file names is limited (Figure 1).

The Intricacies of `rename`

By contrast, the `rename` [2] command is more complex. It is designed to use Perl expressions [3], the same set of wild card characters used in commands such as `awk`, `grep`, and `sed`. In fact, apart from a few examples, the man page assumes that you already know how to use Perl expressions. However, the basic structure is simple enough to pick up on the fly:

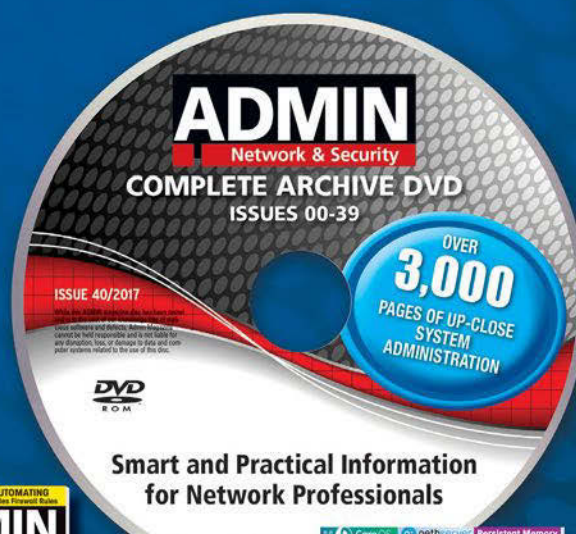
```
rename OPTIONS EXPRESSIONS SOURCE-FILES
```

The first thing to notice about this structure is that, unlike `mv`, `rename` does not include a target file. Instead, the designated files are renamed where they are, based on the structure of the expressions entered.

Secondly, this structure and behavior comes with an increased chance of temporarily losing track of files renamed because of unexpected results. `rename` will not overwrite existing files unless you use the `-force` (`-f`) option, but you further safeguard your files with another

two of `rename`'s handful of options. The first option, `--nono` (`-n`), does a dry run for the command you have constructed, terminating successfully if there are no errors or else listing any syntax problems that need to be corrected. The second option is `--verbose` (`-v`). As in many commands, `--verbose` gives you extra information about the operation as it happens. However, in the case of `rename`, the extra information is a listing of files that have been successfully renamed. This list gives you a way of instantly checking what you have done.

7 Years of ADMIN on One DVD



This searchable DVD gives you 40 issues of ADMIN, the #1 source for:

- systems administration
- security
- monitoring
- databases
- and more!



ORDER NOW!
shop.linuxnewmedia.com

```
bb@nanday:~/files$ mv -v '*intuos4-users-manual.pdf' '#1tablet.pdf'
intuos4-users-manual_.pdf -> tablet.pdf : done
```

Figure 3: `mmv` uses its own syntax to provide a versatile option midway between `mv` and `rename`.

The most important element in a `rename` command's structure is the expressions. Technically, an expression is another option, but its structure is very different from other options. Expressions do not start with one or two hyphens, and they have their own syntax for defining which files are to be renamed and how. Each expression is contained within single quotation marks, indicating that all elements should be processed together, and consists of three parts:

```
'OPERATOR/SEARCH-STRING/OUTPUT-STRING/
MODIFIER'
```

The modifier is optional.

For example, the complete command to convert the name of all files in the current directory from upper case-letters to lower-case letters would be:

```
rename 'y/A-Z/a-z/' ./*
```

With this structure, `rename` offers a huge variety of ways to rename files (Figure 2). In effect, the file name is treated as a file when the expression is applied. Obviously, some Perl expressions are not applicable to a file – for instance, using `\t` to search for a tab would be pointless. However, you can search for ranges of characters and numbers, as in the example above, and search for strings at the start of the file name (^) or the end (\$), two alternatives (a|b), a single character (.), and many more expressions. As you might expect if you have any experience with Perl, there can be more than one structure that will make the same changes.

A complete list of all the possibilities would take pages, but the following list should cover most of the common expressions:

- `'s/ORIGINAL-STRING/RENAMED-STRING/g'`: Replaces one set of characters with another one in every instance. By default, the expression is case insensitive, but adding an `i` as a modifier changes both lower- and upper-case matches. Without the `g`, only the first match is changed.
- `'y/a-z/A-Z/'`: Converts lower-case letters to upper case. By changing the

order of the ranges, you can convert upper case to lower case.

- `'s/\.ORIGINAL-EXTENSION$/.NEW-EXTENSION/'`: Changes the file extension. `$` searches at the end of the file name.
- `'s/STRING-TO-DELETE\./ /' *STRING-TO-DELETE*`: Deletes a series of characters and replaces it with nothing, which is indicated by the blank space between the last two forward slashes.
- `'s/ /_/'`: Replaces a blank space in a file name with an underscore. This example is especially useful if you are ripping or downloading music files or want the files to be used easily from the command line, perhaps for backups. You could also designate a blank space with `\w`. You can use modifications of this structure for removing commas, apostrophes, and other illegal characters in a Bash file name.

Often, the structure to achieve other results may depend on the details of what you want to do. For example, to add a string to file names, you could use `'s/ORIGINAL-STRING/RENAMED-STRING/g'`, adding the new string to the original one in the renamed string. Alternatively, you could start the new string with `.` to place it at the start of the file name, or start the new string with `$` to place it at the end of the file name.

Basically, any renaming that can be done in Perl should be doable using the `rename` command, but it requires more than a passing acquaintance with Perl. For example, incremental file naming can be done using Perl's `sprintf`, although that would be an article in itself.

Other Ways to Rename

If `mv` is too simple for your needs, and `rename`'s use of Perl too complex, you might try `mmv` [4], a newer command that is carried by many distributions, but generally not installed by default. `mmv` has only some of `rename`'s versatility, but is simpler to use. Instead of the command structure having a separate section for expressions, `mmv` incorporates the renaming instructions into the source and target designations. For example, the basic command structure is:

```
mmv '*ORIGINAL-NAME' '#1NEW-NAME'
```

With `*` being the notation for the original name, and `#1` the new name (Figure 3). However, `mmv` can also do more complex operations, such as replacing the first instance of a string in each file name with another string

```
mmv '*ORIGINAL-STRING*' '#1NEW-STRING#2'
```

In all cases, `mmv` offers several options for handling files, such as moving the original file to the new name, copying the original and its permissions and then deleting the original, or renaming the source file without moving it. The examples in the `man` and `info` pages should be enough to get you started, although detailed information about the command structure appears to be missing.

Alternatively, those working with music files might want to return to the desktop and use `pyRenamer` [5], which has all the functions of `rename`, but has the added advantage of reading meta tags directly, which allows the naming of music files in various combinations of elements such as track number, title, and genre. The only disadvantage of `pyRenamer` is that you need to refresh the display manually after adding files to the current directory.

None of the renaming tools I mention are ideal by themselves. However, with a combination of these commands, you can organize your growing collection of files and have a reasonable chance of finding them later. File search utilities can help, but, in the end, nothing beats the organized structure that renaming tools help you to create. ■■■

Info

[1] `mv`: <https://linux.die.net/man/1/mv>

[2] `rename`: <https://linux.die.net/man/2/rename>

[3] Perl expressions: <http://jkorpele.fi/perl/regexp.html>

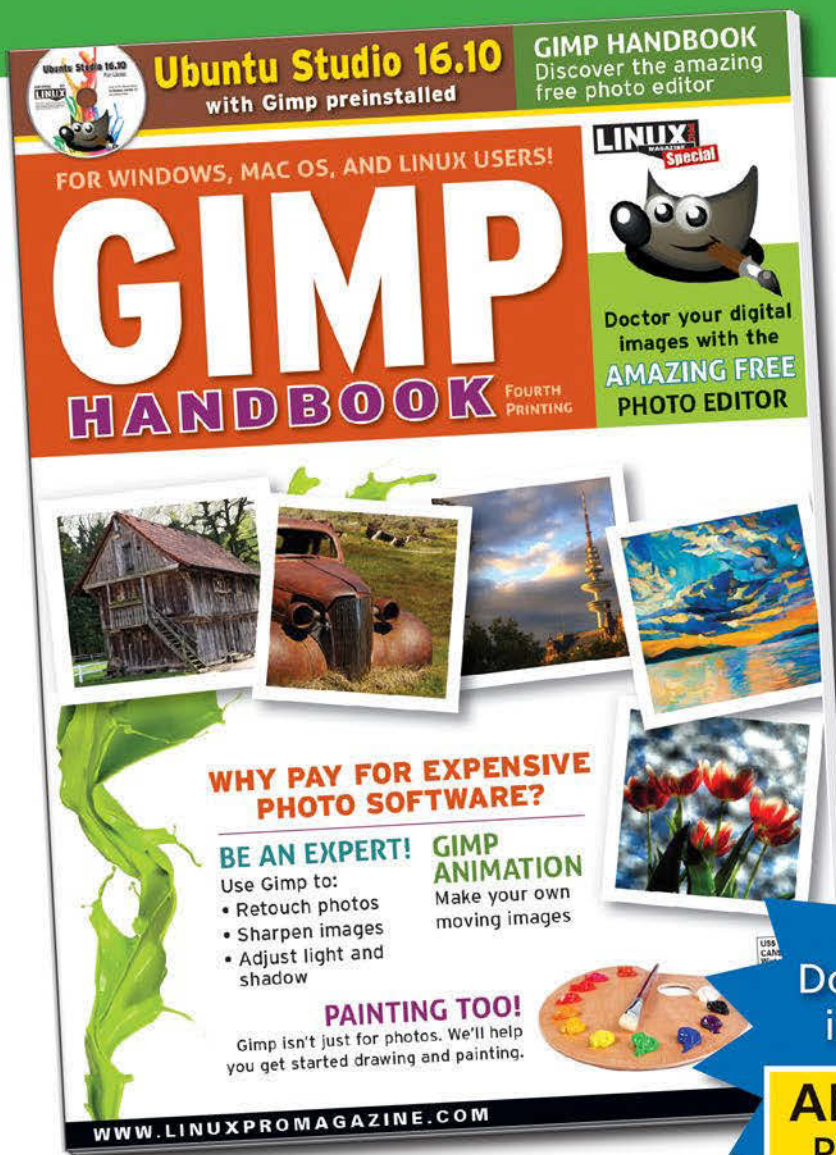
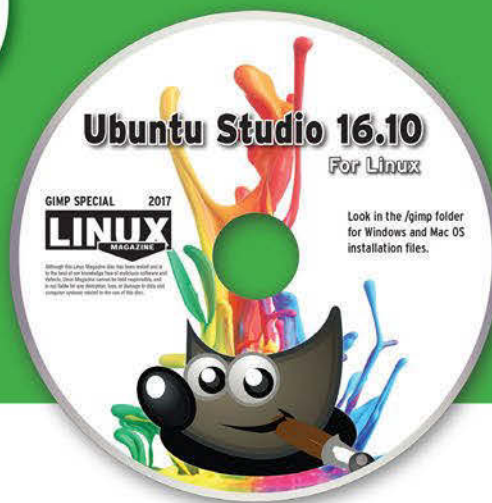
[4] `mmv`: <https://linux.die.net/man/4/mmv>

[5] `PyRenamer`: <https://github.com/SteveRyherd/pyRenamer>

Shop the Shop

shop.linuxnewmedia.com

GIMP HANDBOOK



**SURE YOU
KNOW LINUX...**
but do you know **Gimp?**

- Fix your digital photos
- Create animations
- Build posters, signs, and logos

Order now and become an expert in one of the most important and practical open source tools!

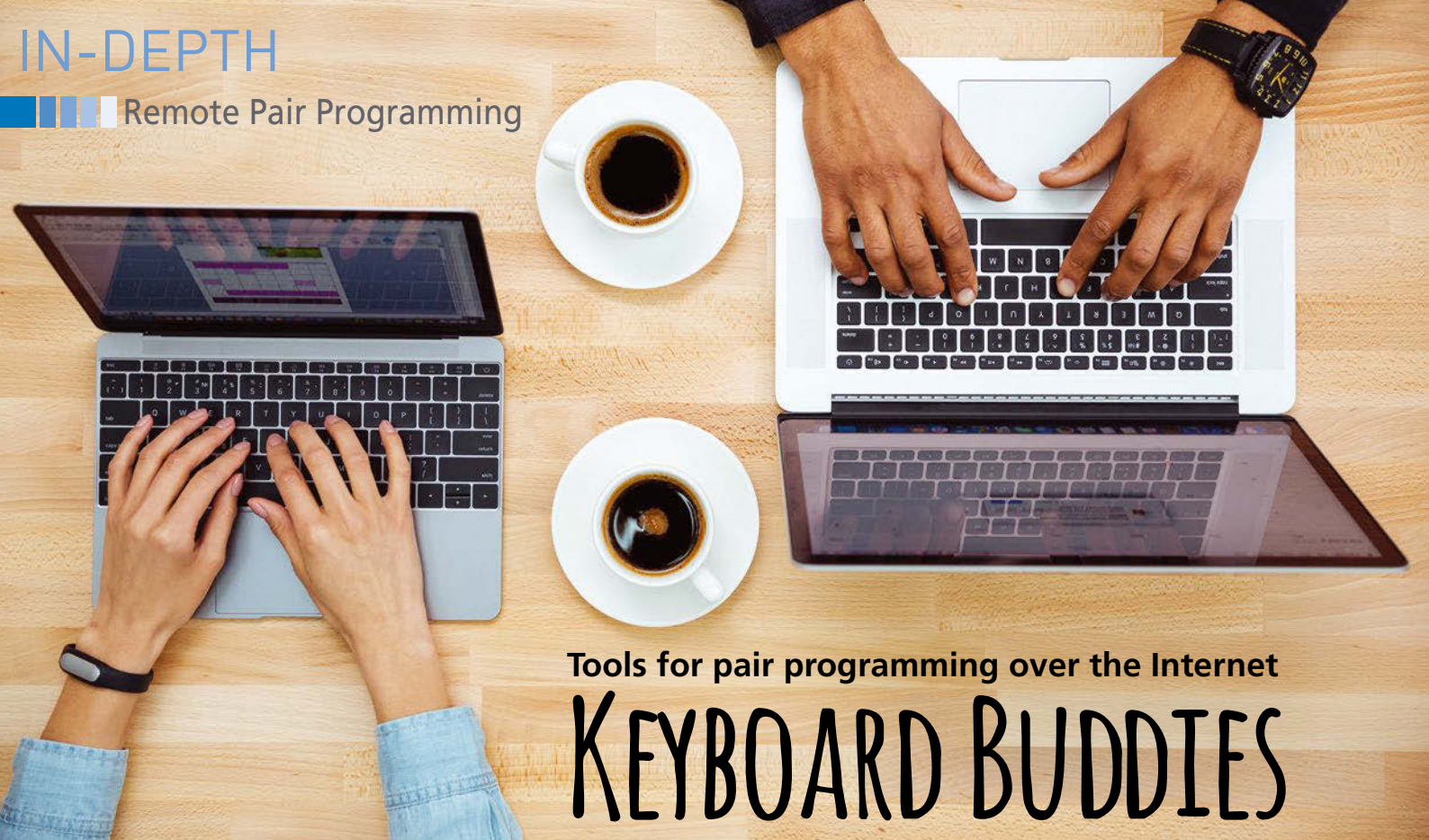
Gimp
Doctor your digital images with the

**AMAZING FREE
PHOTO EDITOR!**

Order online:
shop.linuxnewmedia.com/specials



FOR WINDOWS, MAC OS, AND LINUX USERS!



Tools for pair programming over the Internet

KEYBOARD BUDDIES

The best way to save money on software development is to get it right the first time. Pair programming, an agile technique, saves development costs by putting two coders to work on the same code. Visual Studio Code and *tmate* bring the promise of pair programming to remote workers. *By Stefan Wintermeyer*

As agile programming spreads into the IT space, development teams are increasingly relying on a concept known as pair programming [1]. Pair programming is an agile technique that involves two programmers working simultaneously on one computer. Both programmers use their own keyboard and mouse. Ideally, each also has a separate monitor.

In pair programming, one programmer acts as the *driver* (the person who actively programs), and the other is the *navigator*. Since both programmers have their own keyboards, the navigator can access the code directly at any time. This leads to those “wait a minute, I’ll show you how to do that” moments where the navigator demonstrates something to the driver or simply corrects an error.

Pair programming proponents believe this technique leads to far cleaner code containing fewer errors. However, pair programming does put a strain on both programmers. Experts recommend that companies use pair programming regularly, but not every day

for the entire day. The right chemistry between the two programmers is also important.

The technical requirements for pair programming (two keyboards, two mice, and ideally two monitors) are no problem for most companies; however, the picture looks a little more complicated when you combine pair programming with another recent modern office phenomenon: remote employment.

Pair programming concept can also work for two programmers separated geographically with the help of modern tools, such as webcams and headsets. Part of the concept, however, is that both parties need to be able to intervene directly into events, so the team also needs software that will promote interactive access in a programming context.

This article introduces two popular remote pair programming tools: *tmate* and Visual Studio Code.

tmate

Tmate [2], which is a fork of *tmux* (terminal multiplexer) [3], is a very simple tool that allows two programmers to

work remotely by terminal. You can install *tmate* with your choice of package manager (the package is almost always named *tmate*). If you prefer to compile the source code yourself, you will find it on the project page [2].

The programmer who will manage the source code starts the *tmate* session locally by entering the *tmate* command. The command calls a local *tmate* client and a *tmate* server, which in turn establishes an SSH connection to the centrally hosted *tmate.io* server (Figure 1).

If you have never used SSH before, you first need to generate a key pair with *ssh-keygen*. If desired, you can also host your own *tmate* server, allowing for total control over the communication.

The connection uses *gzip* for compression to make the most of the bandwidth. On the centrally hosted server, the second programmer logs in using a normal SSH client. A *tmate* client then launches automatically. What is impressive about *tmate* is that the second programmer does not require any special software – only SSH. *Tmate* will display the complete SSH command at

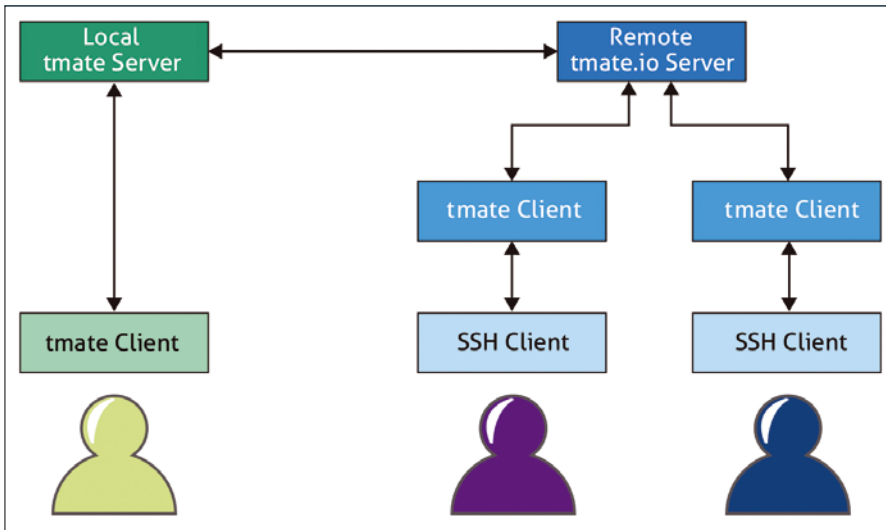


Figure 1: Based on tmux, tmate allows two remote developers to collaborate at the command line.

the bottom of the screen shortly after startup (Figure 2).

Both programmers can work simultaneously in a tmate session. Because they share a terminal, they can use all available editors (e.g., Vim and Emacs) without restrictions. At the end of the work period, the session can be terminated on both sides by typing `exit`. You can also invite other people to join the session, and, if you like, you can restrict the access rights to read-only.

Thanks to the manageable amount of data, the programming partners can still cooperate very well via tmate even with very narrowband Internet connections. If necessary, tmate will even work over a slow 3G mobile phone connection.

Visual Studio Code

Good old terminal editors aren't everyone's cup of tea. Another effective tool for pair programming is Microsoft's Visual Studio Code (VSC) [4], which is open source software and available for Linux.

VSC is well equipped for remote pair programming. With the Visual Studio Live Share extension [5], several people can work on a project at the same time. After installing the extension, they have to log on to a central server; this can be done via a GitHub account, for example. Unlike tmate, however, this server cannot be hosted on a local network. After logging on to the server, the developer starts a share and then sees a URL displayed and copied directly to the clipboard. The developer informs the pairing partner, who then has access.

The driver and navigator can now work on the code at the same time. It is even possible to work with two cursors at different places in the same file. The names of the people involved appear above the cursors. However, this rarely occurs in normal pair programming.

The option to see selections on both sides of the session is very handy. Given simultaneous use of an audio connection, this allows for scenarios in which

the navigator selects code to show that something is wrong, eliminating the need to reference code line numbers.

As an added benefit, the two participants can choose their favorite colors and settings. The text can be white on black for the driver, for example, while the navigator can choose black on pink.

Things get really exciting during debugging. The VSC debugger started by the driver can also be used by the navigator. The navigator doesn't even need to pre-install the appropriate software (e.g., Node). Both can set breakpoints and browse the program bit by bit.

If you develop web applications, you will want to see the results of your work in your browser. For this purpose, the driver can enable the port where its local system is running for the navigator (e.g., 8000). The driver can then access `http://localhost:8000` in their browser and use the application on the driver's system.

From a technical point of view, VSC offers nothing new for Linux professionals, but the simplicity of this solution – and the ability to use it across different operating systems – is impressive.

Conclusions

Tmate and VSC require little bandwidth, and both impress with the perceived speed. Nevertheless, the physical limits mean that a pairing session between New York and Sydney, for example, will always be a touch slower than cooperation between nearby locations. In spite of the delays, however, remote pair programming often works better than working alone. ■■■

Info

- [1] Pair programming: https://en.wikipedia.org/wiki/Pair_programming
- [2] tmate: <https://tmate.io>
- [3] tmux: <https://github.com/tmux/tmux>
- [4] VSC: <https://code.visualstudio.com>
- [5] Visual Studio Live Share: <https://visualstudio.microsoft.com/de/services/live-share/>

Author

Stefan Wintermeyer (<https://www.wintermeyer-consulting.de>) is a consultant, trainer, and book author who writes on the topics of Ruby on Rails, Phoenix, and web performance.

```

stefan — tmate /Users/stefan — tmate — 80x24
Welcome to fish, the friendly interactive shell
stefan@sw ~>
[tmate] ssh session: ssh 0qa6cVFINWvEFGIX9ovob9f7J@ln2.tmate.io
  
```

Figure 2: At the bottom of the screen, tmate shows the session's SSH address.

Using the Electron framework to weed out images

Clever Sampling

Does the private photo archive on your computer just keep on growing without ever seeing any attention? Mike Schilli whips up a home-grown solution to get rid of bad photos with the Electron framework. *By Mike Schilli*

Two months ago, I used facial recognition with artificial intelligence to rummage through my digital shoebox of vacation photos on the quest to discover hidden treasures [1]. That made me realize how little I know about the content of my own photo archive. No doubt this unfortunate state of affairs is caused by my laziness, as every time I come home from a trip, all photos from the mobile phone go directly to a folder on the PC. But once they get there, they tend to grow moss, because there is no keyword attached to them to enable finding them later in a search.

Separating Wheat from Chaff

As a first step, I thought about at least taking out the bad photos before archiving the whole batch. It's hard to do this from the command line, because I have to look at the picture to make a decision.

Now there are a number of programs like Eye of Gnome (eog) for viewing and editing photo collections, but I haven't found one that suits my taste yet. I am looking for a lean application that reads images very quickly and, of course, doesn't prompt me to confirm once I've said I want to delete an image – anything else would be unworthy of an expert. How hard could it be to write something like this myself?

No Exotic Knowledge

Native GUI programming requires exotic expertise – think about the fundamental differences between Gnome, Mac OS, and Windows. A GitHub programmer must have struggled with the same issue to come up with the idea of running local desktop applications in the Chrome browser, which works just the same on all three platforms. And so, GitHub's Electron framework [2] was born.

Electron can bundle GUI programs for all supported target platforms without much additional effort. No comparison to porting a native GUI to other platforms, in which case you might as well start a new project from scratch.

GitHub's Atom editor and Slack – yep, the \$5 billion company that develops and ships its chat client for the desktop as an Electron app – prove that Electron is robust enough to drive commercially successful applications.

Historically Documented

Electron was created when main developer Cheng Zhao linked the rendering engine of the Chromium browser to a Node.js process. This links the browser's restrictive sandbox with the offers of the local operating system and file-system via the Node.js community's extensive range of tools. The main process and renderer always run separately, but somehow have to share a JavaScript context. This is necessary, for example, so that the main process in `main.js` can dynamically change elements on the displayed page in the browser (Figure 1).

These two processes in Electron are running their own event loops and communicate through Inter-Process Communication (IPC). The Node modules that provide the glue between the two sides are `ipcMain` and `ipcRenderer`. For the main process to display in the browser a photo that it has just read from the hard disk, for example, it must first send it to the renderer process, which then refreshes its view of the displayed browser page.

Author

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



Ready, Steady, Go

An article on setting up the Electron Framework [3] was published in *Linux Magazine* a few months ago, so I mention the preparation only briefly and then head on directly to processing the photo folders.

The following commands install the Electron framework on Ubuntu:

```
sudo apt-get install npm nodejs-legacy
```

The additional `nodejs-legacy` package only installs some symlinks that many older node modules need during build and execution. In a fresh directory, then run

```
npm init
npm install electron --save-dev
```

to create a new project that not only installs the Electron framework locally, but also adds its dependencies to its dependency list, helping adopters to modify and rebuild the code to their heart's content. The `npm init` command prompts the user to enter some project parameters, such as the application name, its version, or the author name (Figure 2). The `--save-dev` option of the `npm install` statement appends the name of the in-

stalled package to the `devDependencies` list in `package.json`. For comparison, `--save` would list the package as a run-time dependency.

If you also add the following to the `scripts` section inside `package.json`,

```
"start": "electron ."
```

the application can be launched later by using `npm start`. Electron then initially loads the `main.js` start script in Listing 1 [4] (specified in the configuration file under `main`) and passes it to the Node.js interpreter for execution.

Courage to be Different

As is well known, the asynchronous functional approach used by Node.js means a very different programming style compared with “normal” languages like Python or Perl. Instead of sequentially processing calls, Node.js code often adds a callback

to a function call. The function later resumes execution by calling it at the end.

The GUI code builds a state machine, between whose states the code jumps back and forth, as controlled by events. At the same time, the event loop always needs to watch out for new events such as mouse clicks, to which it must respond promptly. This would not work if the code were just blocked for a while because it was reading a large file from disk.

The code in Listing 1 does not execute anything at first but waits until the

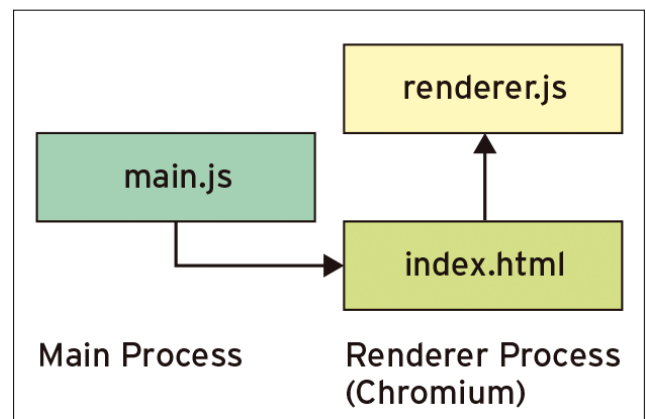


Figure 1: Electron is divided into the main process and the renderer in the Chromium browser code. While the program is running, the two communicate back and forth.

Listing 1: main.js

```

01 const {app,globalShortcut,BrowserWindow} =
02   require('electron');
03 const path = require('path');
04 const url = require('url');
05
06 let win;
07
08 function createWindow(){
09   win = new BrowserWindow({
10     width:800, height:600});
11
12   win.loadURL(url.format({
13     pathname:
14       path.join(__dirname, 'index.html'),
15     protocol: 'file:', slashes: true
16   }));
17
18   win.webContents.openDevTools();
19
20   win.on('closed', () => {
21     win = null;
22   });
23 }
24
25 app.on('ready', () => {
26   createWindow();
27   globalShortcut.register('l', () => {
28     win.webContents.send('nextImage');
29   });
30   globalShortcut.register('h', () => {
31     win.webContents.send('prevImage');
32   });
33   globalShortcut.register('d', () => {
34     win.webContents.send('deleteImage');
35   });
36   win.webContents.send('prevImage');
37 });
38
39 app.on('will-quit', () => {
40   ['h','l','d'].forEach(function(key){
41     globalShortcut.unregister(key);
42   });
43 });
44
45 app.on('window-all-closed', () => {
46   app.quit();
47 });
  
```

```

$ npm init
name: (eg) inuke
version: (1.0.0)
description: Photo Discharge Utility
entry point: (index.js) main.js
test command:
git repository:
keywords:
author: Mike Schilli <m@perlmeister.com>
license: (ISC) MIT
About to write to package.json:
{
  "name": "inuke",
  "version": "1.0.0",
  "description": "Photo Discharge Utility",
  "main": "main.js",
  "scripts": {
    "test": ""
  },
  "author": "Mike Schilli <m@perlmeister.com>",
  "license": "MIT"
}
Is this ok? (yes) y

```

Figure 2: `npm init` creates the JavaScript file package. `js` as the basic Node.js configuration for the iNuke application.

node environment reports the ready event. If this occurs, it starts the callback from line 25 and first submits a web page to the renderer process for display with `createWindow()`. This happens from line 8 and with an object of the `BrowserWindow` class, whose `loadURL()` method is given the path to the `index.html` file in Listing 2.

At the same time, Listing 1 uses the global variable `win` to store a reference to the browser window. It can reset the variable during later callbacks, as being performed by the handler of the `closed` event, which gets triggered by the windowing system, and handles freeing up memory before shutting down the program.

During the debug phase of a

new Electron application, it is extremely useful to open Chromium's debug window in the browser's main window using `openDevTools()` (line 18) and either read the warnings at the command line or analyze the HTML of the dynamically refreshed web page (Figure 3).

Short and Sweet

Intercepting keyboard input is also a task of the main process in `main.js`. The register

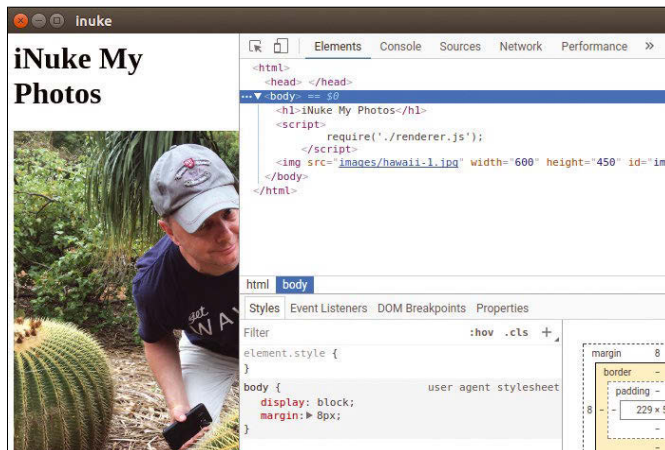


Figure 3: When the debug window is open, the developer can analyze the displayed HTML or track messages on the console.

calls in lines 27, 30, and 33 ensure that the user can move to the next image with `L` and to the previous image with `H` (just as you move left or right in Vim) and delete the displayed image with `D`.

Among other things, these keystroke commands affect the displayed web page, which is why the main process `main.js` sends them as events to the renderer process in Listing 3 via IPC and `win.webContents.send()`. The renderer process starts out at the very beginning of the main process in Listing 1. It loads the `index.html` file (Listing 2) into the browser in lines 12 to 16, which in turn executes the renderer's JavaScript (Listing 3) in line 8 of Listing 2 via `require('./renderer.js')`.

There Is No Going Back

Listing 2 defines the currently displayed photo's image tag in line 11 with the `id` attribute of `image`. This makes it easy for the renderer process to find it later and replace it with a new photo, if the user commands this via keyboard input.

Figure 4 shows the finished application. In it, you can delete an image by pressing `D`, move to the next one with `L`, or go back to the previous one with `H` if you change your mind. However, there is no undelete – that would not make sense in Unix philosophy.

The application is shut down either by the user closing the window or – as usual under Gnome – by pressing `Ctrl + W`. These events are intercepted by the event handlers `will-quit` or `window-all-closed`. They release the key bindings and call the application's `quit()` method, which releases all occupied resources.

Listing 2: `index.html`

```

01 <html>
02   <head> </head>
03
04   <body>
05     <h1>iNuke My Photos</h1>
06
07     <script>
08       require('./renderer.js');
09     </script>
10
11     <img id="image"></img>
12
13   </body>
14 </html>

```


The renderer process in Listing 3 uses the `blueimp` image library. This presupposes that the user previously installs it using

```
npm install blueimp-load-image --save
```

from the `npmjs.com` Node.js repository. The `npm` package manager notices the new addition in the local package. `json` file's dependency list, ensuring that curious adopters only have to clone the project and call `npm install` to retrieve and install all the modules on which the project depends.

The photo files, which were read using `readdir()` from the `images` directory (line 40) when the program started, are stored by the program in the global `images` array variable. The `images_idx` variable (line 6) remembers the index of the currently displayed image within the array.

Fielding Signals

The `scroll()` function from line 19 expects the direction (1 for forward, -1 for back) in which the user wants to shift through the collection as a parameter. It

then sets the global variables to the new photo to be displayed and ensures that no one moves outside the boundaries of the array in the body of the function.

Lines 51 and 52 in the renderer process receive the signals triggered by the main process from Listing 1. The handlers trigger when the user presses the corresponding key. Line 53 reacts to the keypress `D` in the main process and jumps to the `deleteImage()` function from line 29 when activated. The Node.js `unlink()` method from the `fs` class mercilessly removes the file from the disk and displays the next photo.

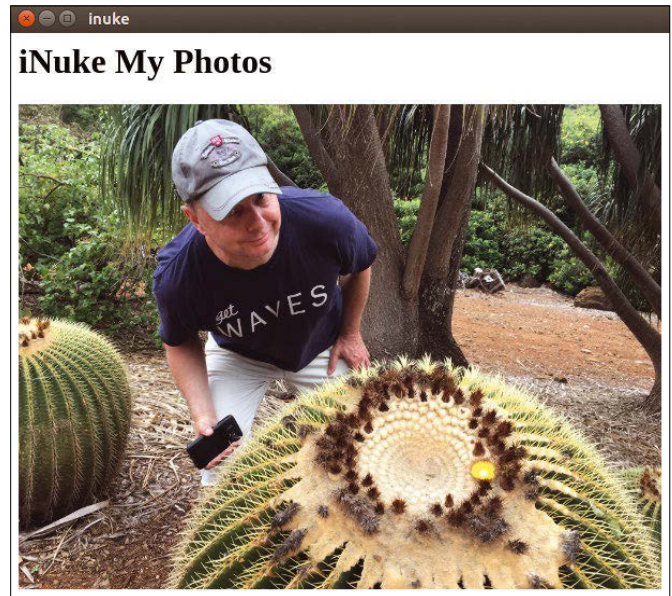


Figure 4: The iNuke app on Ubuntu.

Loading Photos

Loading a photo from disk and displaying it in the browser window is the task of the `displayImage()` function from line 8. It uses the `loadImage()` function exported by the `blueimp` library from the NpmJS repository to

What?!
I can get my issues SOONER?



Available anywhere, anytime!

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

shop.linuxnewmedia.com/digisub

Listing 3: renderer.js

```

01 loadImage = require('blueimp-load-image');
02 fs = require( 'fs' );
03 ipc = require('electron').ipcRenderer;
04
05 images      = [];
06 images_idx = -1;
07
08 function displayImage(file) {
09   loaded = loadImage(file, function(img) {
10     scaled_img = loadImage.scale(
11       img, {maxWidth: 600});
12     scaled_img.id = "image";
13     node = window.document.getElementById(
14       'image');
15     node.replaceWith(scaled_img);
16   } );
17 }
18
19 function scroll(direction){
20   images_idx += direction;
21   if(images_idx > images.length-1){
22     images_idx = images.length-1;
23   }else if(images_idx<0) {
24     images_idx = 0;
25   }
26   displayImage( images[ images_idx ] );
27 }
28
29 function deleteImage() {
30   fs.unlink(images[ images_idx ]);
31   images.splice(images_idx, 1);
32   if(images.length == 0) {
33     console.log("That's it. Good-bye!");
34     require('electron').remote.app.quit();
35   }
36   scroll(-1);
37 }
38
39 dir = "images"; // change to process.cwd()
40 fs.readdir(dir, function(err, files) {
41   if( err ) {
42     console.error("readdir:", err);
43     require('electron').remote.app.quit();
44   }
45   files.forEach(function(file, index) {
46     images.push( dir + "/" + file );
47   });
48   scroll(0);
49 } );
50
51 ipc.on('nextImage', () => { scroll(1); });
52 ipc.on('prevImage', () => { scroll(-1); });
53 ipc.on('deleteImage', deleteImage);

```

load the photo. The `scale()` method then scales it to a maximum width of 600 pixels. The renderer then searches the browser's HTML for an image tag with the `id` attribute `image` and replaces the image found there with the new image (line 15). Before this, Listing 3 also sets the new image's `id` attribute to `image` so that the refresh process will find the tag next time.

Next Steps

Instead of starting the application with `npm start`, the `electron-builder` add-on package bundles it into a binary. This contains everything you need, including the Chromium browser and libraries on which the project depends. However, the `electron-builder` package requires a newer Node.js version than the one provided by Ubuntu 16.04. If you like, you can install it from the official Node.js repository.

The results can be easily copied from host to host in file format and run perfectly on similar architectures. If you want to search for images in the current directory rather than the `images` directory, you will want to set the string in

line 39 of Listing 3 to `process.cwd()` after on-boarding the appropriate module with `require('process')`.

Changing Platform

If you want to transfer it to another platform, for example the Mac, just call `npm install` and `npm start` there in the same directory, and you will be surprised: The whole thing works perfectly with the usual GUI conventions. A bundle of joy.

The book *Cross-Platform Desktop Applications using Electron and NW.js* [5], as well as the blog posting by Jessica Lord [6], offer a good starting point for anyone who now can't resist the urge to delve deeper into the subject of GUI programming with Electron.

One useful addition to the iNuke application would be to insert tags into photos. The good photos of a Hawaii vacation would then be assigned a tag of *hawaii 2018*, and portrait photos tagged with the name of the person depicted, and so on. Following this step, a search program could easily retrieve them later.

A few buttons with tag names on the GUI's right side that insert a tag into the Exif area of the current photo, plus a

New Tag button to help you create new tags – how hard can that be? We may well see. ■■■

Info

- [1] "Programming Snapshot – Facial Recognition" by Mike Schilli, *Linux Magazine*, issue 214, September 2018, p. 56, <http://www.linuxpromagazine.com/Issues/2018/214/Open-source-libraries-for-facial-recognition/>
- [2] Electron framework: <https://github.com/electron/electron>
- [3] "Exploring the Electron Application Framework" by Andreas Möller, *Linux Magazine*, issue 210, May 2018, p. 30, <http://www.linuxpromagazine.com/Issues/2018/210/Electron/>
- [4] Listings for this article: <ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/216/>
- [5] Jensen, Paul B. *Cross-Platform Desktop Applications using Electron and NW.js*. Manning, 2017 <https://www.manning.com/books/cross-platform-desktop-applications>
- [6] "Essential Electron" by Jessica Lord: <http://jlord.us/essential-electron/#stay-in-touch>

The sys admin's daily grind: grepcidr

Thwarting Spammers

Often it is the very simple tools that, when used appropriately, lead to the greatest success. This time, sys admin columnist Charly employs an IP address filter to count the devices in his home and trip up spammers to boot. *By Charly Kühnast*

Although Linux has many `grep` variants, you can always find a new one. I only discovered `grepcidr` [1] a few months ago. As the name suggests, the tool filters input by IP addresses and networks. It works equally well with IPv4 and IPv6. To show `grepcidr`'s capabilities, I will use it to compile a list of all IPv4 addresses on my home network. I got this from the Syslog on the firewall, which is also the DHCP server:

```
cd /var/log
grepcidr 10.0.0.0/24 syslog |>
grep DHCPACK|tail -n 1500 |>
cut -f9 -d" " |sort|uniq > 1stlist
```

The `1stlist` file now contains 46 IP addresses:

```
10.0.0.135
10.0.0.15
10.0.0.150
10.0.0.16
10.0.0.166
[...41 more...]
```

I automated this discovery process with the following command:

```
grepcidr 10.0.0.0/24 -c < ./1stlist
```

In this simple case, it would have been faster with `wc -l`, but `grepcidr` ultimately shows its strengths when you have to filter out different networks from such a file.

After a while, I repeat the game and write the list of IP addresses written to the `2ndlist` file. For my statistics,

```
grepcidr 10.0.0.0/24 -c < ./2ndlist
```

shows that the file is one line shorter, so there is one less device on the network. I can easily find out which one is missing with `diff`:

```
diff 1stlist 2ndlist
2d1
< 10.0.0.15
```

To see how many devices in my house are run 24/7, I send the data to a round-robin database and have a history graph drawn. The interruption at about 6am in Figure 1 is a routine reboot of the DHCP server.

Fighting Off Mail Pests

I also used `grepcidr` to create IP blacklists on my mail server. I have set up some mail addresses that I don't use but that spammers will typically test for their existence: `sales@...`, for example. Twice a day, `grepcidr` plows through the log and extracts all the IPs from the log lines that relate to these mailboxes and writes them to the `harvest` file.

Because these lines also contain my own servers' IPs, I have to remove them with a whitelist, which is also a file with IP addresses.

The ready-to-use blacklist is now built by `grepcidr` in a single step:

```
cat harvest|grepcidr -vf whitelist >
> blacklist
```

This trick has helped me keep many a spammer out of my life. ■■■

Info

[1] `grepcidr`: <http://www.pc-tools.net/unix/grepcidr/>

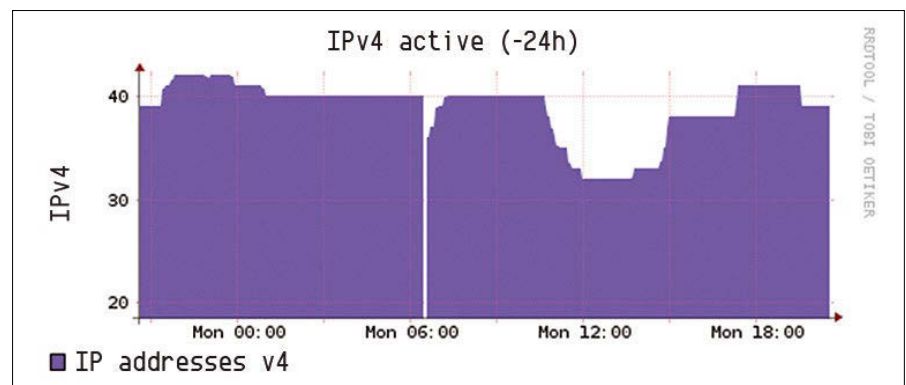


Figure 1: To be or not to be a device – `grepcidr` clarifies the question.

Author

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

MakerSpace

Network monitoring and
GPIO control with SNMP

Pins and Nodes

Monitor disk space and network and CPU loads with SNMP, view and control Rasp Pi GPIO pins remotely with custom SNMP objects, and create web dashboards with Node-RED. *By Pete Metcalfe*

Some of the great full-featured networking packages like Nagios [1] and MRTG [2] can be loaded on a Raspberry Pi. If, however, you are looking for something a bit smaller in scale, a Simple Network Management Protocol (SNMP) installation and Node-RED is a great place to start. Node-RED [3] is a visual programming environment that lets you create applications by dragging and dropping blocks (nodes) on the screen and directing the logic flow by connecting the nodes together.

In this article, I look at some basic SNMP monitoring that will allow you to integrate a Raspberry Pi into a larger network (Figure 1).

SNMP

SNMP is the standard for communicating with and monitoring network devices. Common device information is grouped into management information bases (MIBs). Data items are object identifiers (OIDs), referenced by either their MIB name or their OID numeric name. For example, the SNMP device could be queried by its MIB name (e.g., *SNMPv2-MIB::sysName.0*) or its OID number (e.g., *.1.3.6.1.2.1.1.5.0*).

Install the SNMP monitor and server on your Rasp Pi with:

```
sudo apt-get update
sudo apt-get install snmp snmpd
snmp-mibs-downloader
```

To show meaningful MIB names, modify the SNMP config file by opening it in the Nano editor

```
sudo nano /etc/snmp/snmp.conf
```

and commenting out the first (*#mibs*) line. The SNMP server agent has many configuration options. For testing (not recommended in a real system), begin by opening the configuration file and uncommenting the *agentAddress* line, so all interfaces are open:

```
sudo nano /etc/snmp/snmpd.conf

# Listen for connections on all
# interfaces (both IPv4 *and* IPv6)
agentAddress udp:161,udp6:[::]:161
```

In the *ACCESS CONTROL* section, add a line to give read/write access to the public and comment out all other access control definitions:

```
# ACCESS CONTROL
#
# Set read/write access to public
# anywhere
#
rwcommunity public
```

After saving the changes you've made to *snmpd.conf*, the service needs to be restarted:

```
sudo service snmpd restart
```

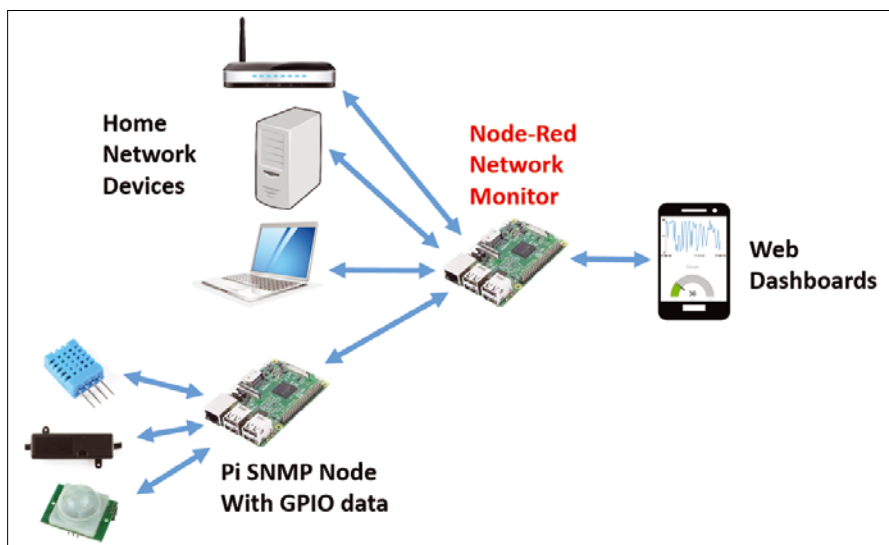


Figure 1: Monitor devices in your home network and various sensors connected to Rasp Pi GPIO pins with Node-RED and a smartphone.

Table 1: SNMP CLI Programs

Program	Function
snmpget	Gets an SNMP message for a specific OID.
snmpset	Sets an SNMP OID (OID needs to be writable).
snmpwalk	Gets multiple OID values in an MIB tree.

A number of useful SNMP command-line programs are available (Table 1). The basic syntax for these commands is:

```
command -c <community> -v <version> <node> <OID>
```

To begin, test SNMP on the Rasp Pi to see if it is working by running the `snmpwalk` command, starting at the top of the tree (.1.3); it should return a long list of all the available SNMP OIDs (Listing 1).

Node-RED

Node-RED has been preinstalled on Raspbian Jesse since the November 2015 version. For this networking example, you will need to load the Node-RED SNMP module, dashboard, and timer library by entering:

```
sudo apt-get update
sudo apt-get install npm
cd $HOME/.node-red
npm install node-red-node-snmp
npm install node-red-dashboard
npm install node-red-contrib-bigtimer
```

At a terminal window, you can start Node-RED with:

```
node-red-start &
```

Node-RED has a web configuration interface that is accessed at `http://localhost:1880` or `http://<pi_ip_address>:1880`.

A good first Node-RED application is to make a web dashboard that shows ping

(node-to-node) delay times. The dashboards are defined in the right panel of Node-RED (Figure 2). Dashboard items are put into groups, and groups are put into tabs. Each tab will be shown as a separate page on your smartphone.

The logic is created by dragging and dropping ping and chart nodes onto the flow sheet (Figure 3) and then wiring the ping outputs to the chart inputs. Next, double-click the ping node and enter the IP address of the node to ping and the scan rate (Figure 4). Similarly, double-click the chart node and enter the dashboard group, size, label, and chart time (Figure 5).

After the configuration is done, click the `Deploy` button on the Node-RED menu bar. The Node-RED dashboard user interface is accessed at `http://<IPaddress>:1880/ui` (e.g., `http://192.168.1.102:1880/ui`). Figure 6 shows the final dashboard. Chart data values are shown by clicking on the chart line.

Node-RED SNMP Example

The ping node is quite simple, returning just the ping value. The snmp node is more complex, returning multiple pieces of information. To use snmp nodes in a Node-RED application, you need some support nodes to parse and pass the payload messages. Sending SNMP data to a

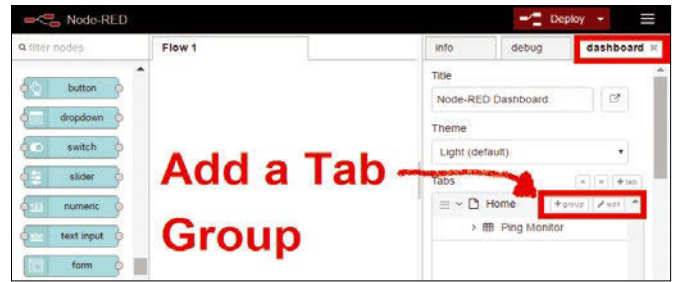


Figure 2: Dashboard configuration.

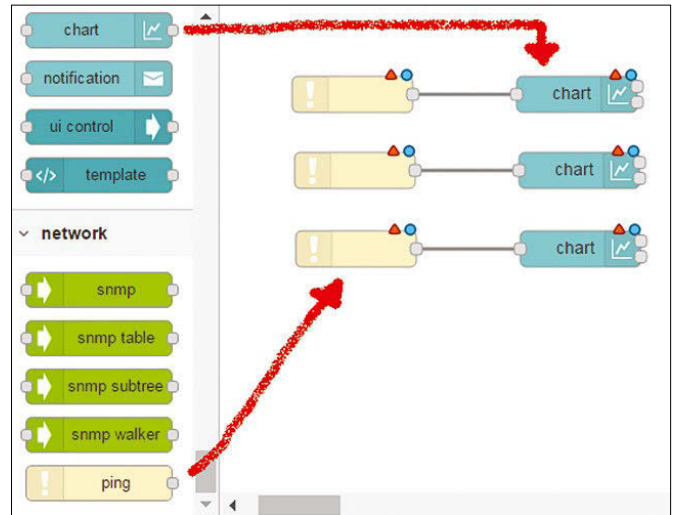


Figure 3: Ping logic.

chart dashboard (Figure 7) uses the following nodes:

- Big Timer – Triggers the polling of data.
- snmp – Gets SNMP and OID information.
- split – Splits the message into addressable variables.
- change – Puts the OID value into the message payload.



Figure 4: The ping node configuration.

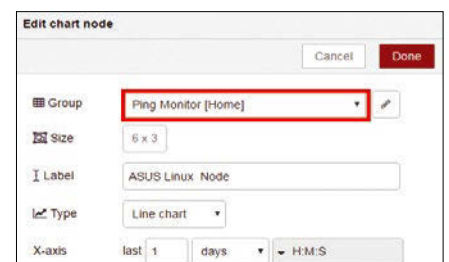


Figure 5: The chart node configuration.

Listing 1: Testing SNMP

```
$ snmpwalk -c public -v 1 localhost .1.3

SNMPv2-MIB::sysDescr.0 = STRING: Linux raspberrypi 4.4.21-v7+ ...
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (319234) 0:53:12.34
SNMPv2-MIB::sysContact.0 = STRING: Me <me@example.org>
SNMPv2-MIB::sysName.0 = STRING: raspberrypi
SNMPv2-MIB::sysLocation.0 = STRING: Sitting on the Dock of the Bay
...
```

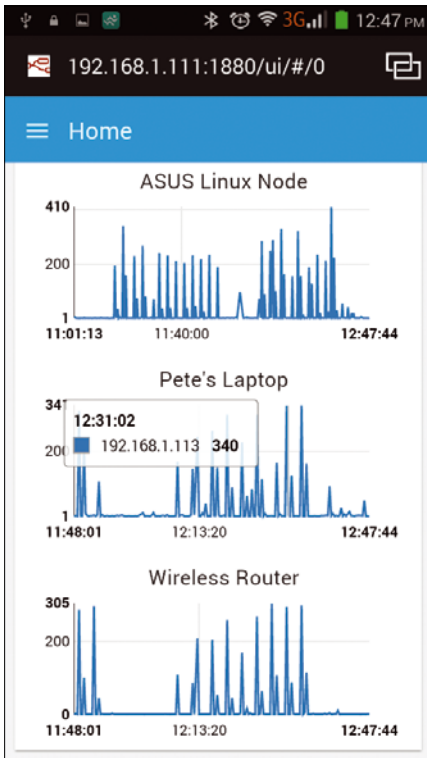


Figure 6: Ping web dashboard.

• chart – Shows the payload.
 The Big Timer node has many useful options. For a simple one-minute monitoring circuit, the middle Big Timer output pin is connected as an input to the snmp node. The snmp node needs a host, community, and numeric OID configuration (Figure 8). The split node should be set up to split on a comma (,).



Figure 9: The change node configuration.

Table 2: Useful SNMP Objects to Monitor

Object	OID
One-minute CPU load	.1.3.6.1.4.1.2021.10.1.3.1
Five-minute CPU load	.1.3.6.1.4.1.2021.10.1.3.2
15-minute CPU load	.1.3.6.1.4.1.2021.10.1.3.3
Idle CPU time (%)	.1.3.6.1.4.1.2021.11.11.0
Total RAM in machine	.1.3.6.1.4.1.2021.4.5.0
Total RAM used	.1.3.6.1.4.1.2021.4.6.0
Total RAM free	.1.3.6.1.4.1.2021.4.11.0
Total disk/partition size (KB)	.1.3.6.1.4.1.2021.9.1.6.1
Available space on disk	.1.3.6.1.4.1.2021.9.1.7.1
Used space on disk	.1.3.6.1.4.1.2021.9.1.8.1

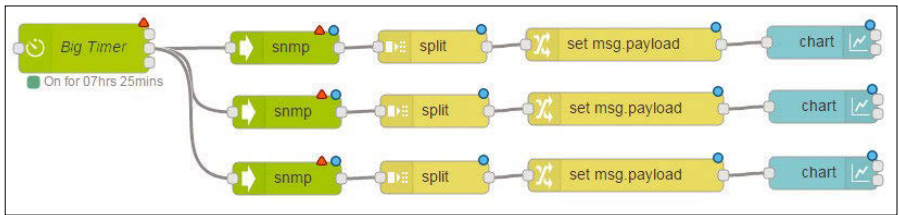


Figure 7: SNMP to chart dashboard logic.

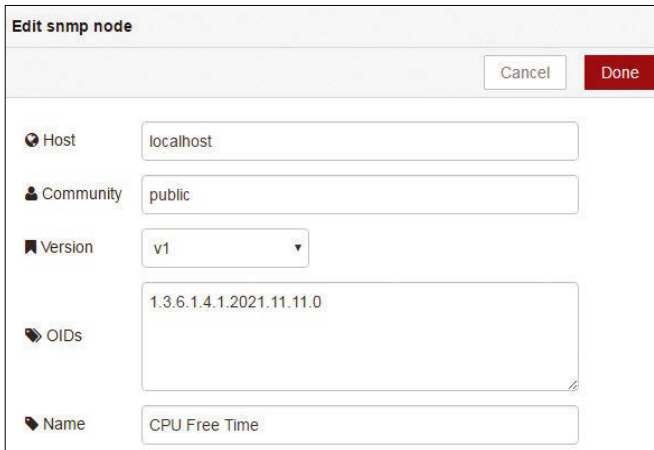


Figure 8: The snmp node configuration.

The powerful change node can adjust and move information within the msg and payload items. For this application, you need to move payload.value to payload (Figure 9). Like the ping example, the chart node is configured as a line chart with the required labels and ranges.

Many other SNMP objects can be monitored. Table 2 shows some of the more commonly monitored objects, and the dashboard in Figure 10 shows the typical output of monitoring CPU load, total RAM free, and free disk space.

Custom GPIO/SNMP Object

A Rasp Pi GPIO pin (pin 18 in this example) is set for read/write access with:

```
gpio mode 18 output
```

You can test setting and reading this pin with the gpio read and gpio write commands:

```
gpio read 18
0
gpio write 18 1
1
```

The Net-SNMP agent snmpd supports the creation

of custom read/write objects (OIDs). The pass-through MIB extension command in snmpd.conf allows you to call script files. Pass-through script files need to follow a few rules:

- An snmpget request passes a -g parameter.
- The snmpget response needs to be

three lines: OID, data type, and value.

- An snmpset request passes a -s parameter and value as the fourth item.
- Listing 2 is a Bash script file that reads pin 18 on an snmpget request (-g), and it will write to pin 18 on an snmpset (-s). This script is made executable by entering

```
chmod +x powerswitch
```

To enable SNMP calls from the powerswitch script file, it needs to be referenced in snmpd.conf. Look for the "Pass-through" section and add a line with the OID and shell you want to use and a path to the script file, such as:

```
# "Pass-through" MIB extension command
#
pass .1.3.6.1.2.1.25.1.8 /bin/bash
/home/pi/powerswitch
```

After the snmpd.conf file is changed, the snmpd service needs to be restarted. If everything is done correctly, you can use the SNMP command-line routines to test reading and writing to the newly created OID (Listing 3). As a future step you can also give your new custom OID a meaningful MIB name.

Node-RED SNMP Set Example

An exec node can be used to run command-line utilities. Figure 11 shows the

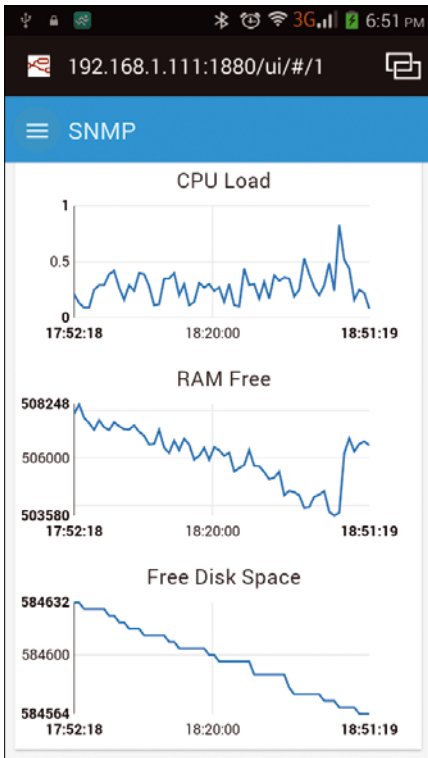


Figure 10: Node-RED SNMP dashboard.

Listing 2: powerswitch Script File

```
#!/bin/bash
if [ "$1" = "-g" ]
then
echo .1.3.6.1.2.1.25.1.8
echo integer
gpio -g read 18
fi

if [ "$1" = "-s" ]
then
gpio -g write 18 $4
fi

exit 0
```

Listing 3: Testing SNMP Commands

```
$ snmpset -c public -v 1 localhost .1.3.6.1.2.1.25.1.8 i 1
HOST-RESOURCES-MIB::hrSystem.8 = INTEGER: 1

$ snmpget -c public -v 1 localhost .1.3.6.1.2.1.25.1.8
HOST-RESOURCES-MIB::hrSystem.8 = INTEGER: 1

$ snmpset -c public -v 1 localhost .1.3.6.1.2.1.25.1.8 i 0
HOST-RESOURCES-MIB::hrSystem.8 = INTEGER: 0

$ snmpget -c public -v 1 localhost .1.3.6.1.2.1.25.1.8
HOST-RESOURCES-MIB::hrSystem.8 = INTEGER: 0
```

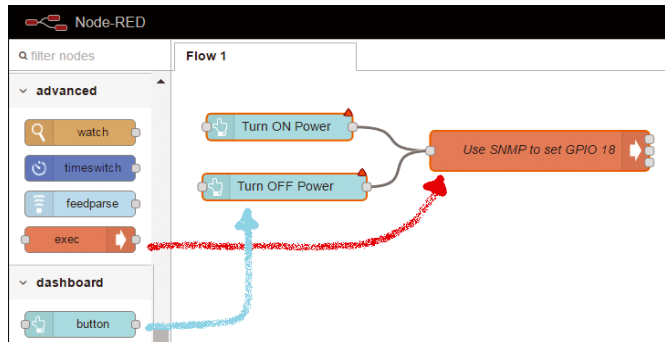


Figure 11: Set SNMP values with buttons.

logic that sets an SNMP value with an *On* and *Off* button. The button nodes are configured to pass a 1 or 0 payload. The exec node contains the snmpset command (Figure 12), which will append the 0 or 1 from the button node. My final web dashboard (Figure 13) includes a gauge to monitor the output status along with the buttons to control pin 18.

Summary

Network monitoring with the Raspberry Pi can be accomplished in a number of ways. With some basic SNMP configuration on the Rasp Pi, you can make data available to higher level packages or create simple standalone solutions with Node-RED.

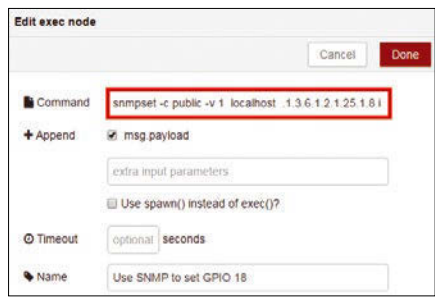


Figure 12: The exec node configuration.

A number of other useful GPIO projects are possible with SNMP, as well. For example, I use a PowerSwitch Tail II (\$26) [4] power cord that can be enabled and disabled through I/O pins to monitor and control power-

ered devices like lights, PCs, and dehumidifiers from SNMP. ■■■

Info

- [1] NagiosPi: <http://everyday-tech.com/nagiospi-server-monitoring-with-the-power-of-pi/>
- [2] MRTG: <https://oss.oetiker.ch/mrtg/>
- [3] Node-RED: <https://nodered.org/>
- [4] PowerSwitch Tail II: <http://www.powerswitchtail.com/>

Author

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

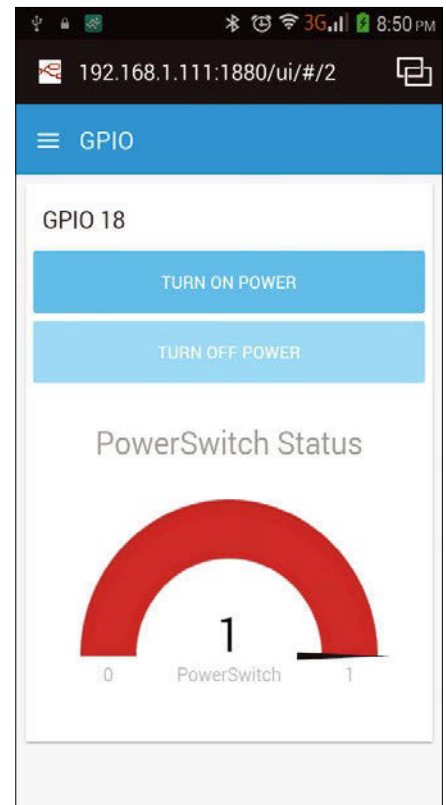


Figure 13: The PowerSwitch dashboard.

MakerSpace

A good cents exit survey

A Penny for Your Thoughts

Get feedback for live events with an exit survey that counts pennies. *By Scott Sumner*

To generate useful feedback for live programs at the Fort Worth Museum of Science & History's Noble Planetarium on the campus of Central Texas College in Killeen, Texas, we conceived, designed, and built a custom feedback system centered around pennies. The goals were (1) ease of use for both guests and staff, (2) a fast participation process, and (3) immediate feedback to the presenters and managers.

Author

Scott Sumner is the assistant manager of the Charlie Noble Planetarium at the Fort Worth Museum of Science and History. He enjoys using Python to solve as many problems as possible.

The Guest Experience

When you attend a planetarium show at the Noble Planetarium, you are handed a penny on the way into the theater. As a

part of the show, we ask you to hold out the penny at arm's length and imagine that you are looking through Abraham Lincoln's eye. That approximates the field of view of the Hubble Space Telescope, which is the frame of reference for the remainder of the program.

As you exit the theater, the presenter has rolled out an interesting device that looks sort of like a coin bank (Figure 1). You are invited to return the penny in one of five slots to indicate how much you liked the show. As the pennies are returned, they are tallied by the machine; once everyone has exited, the presenter



Figure 1: The Penny Counter on the planetarium console ready to accept pennies. Different locations were tested to see whether response levels changed. Staff always step away from the counter as guests are exiting, so they don't influence the ratings.

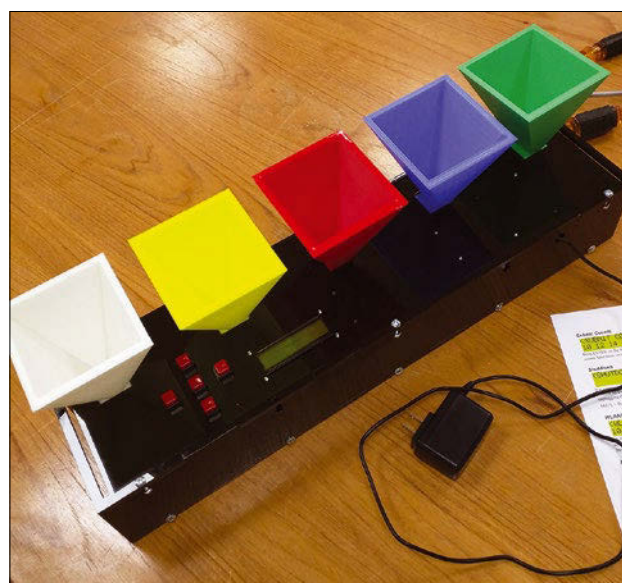


Figure 2: The Penny Counter after final assembly. Note the captured screws along the back and the nuts visible along the bottom panel.

Lead image © olegduenko, 123RF.com

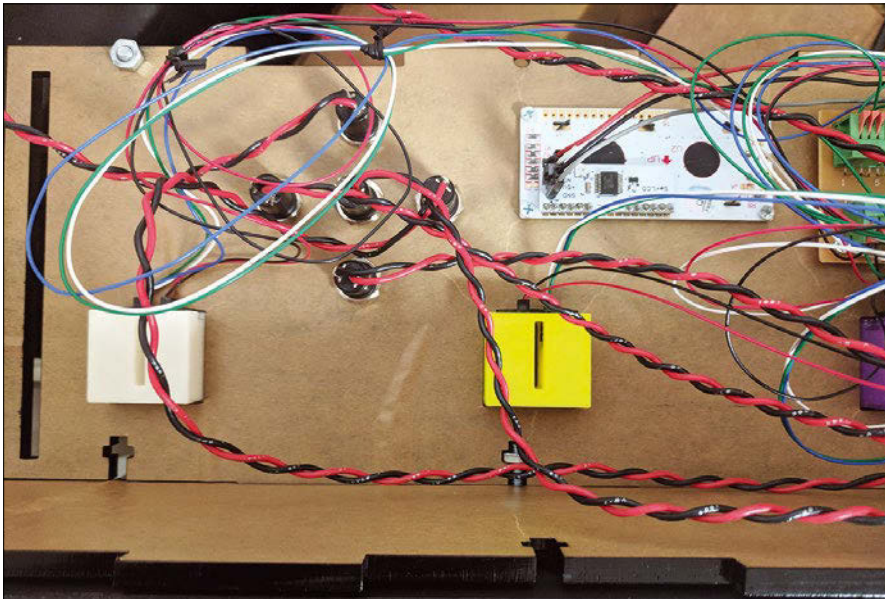


Figure 3: The buttons as seen from inside the Penny Counter. Wires were twisted with a power drill to keep each button pair together. The white board is the 2x16 LCD panel. The smaller bundles of blue, white, and green wires are the phototransistor side of the slot sensor. The small red and black wires visible next to each funnel are for the infrared LED illuminating the sensor on the opposite side.

submits the counts, which are immediately visible on an internal web page.

Hardware and Software

The funnels and case for the penny counter were designed using Blender [1] and printed with a 3D printer [2]. The completed counter is shown in Figure 2.

Under the hood, the Penny Counter's brain is a Raspberry Pi. A two-line by 16-character LCD display shows a menu for the operator; five buttons in a cross pattern provide navigation, and five GPIO pins read infrared sensors at the bottom of each funnel that watch for pennies. The Penny Counter's code is written in Python. It connects to a MySQL database to retrieve show names and post the finished tally after all of the guests have departed.

The Rasp Pi was connected to a generic printed circuit board (PCB) that breaks out the connections from the GPIO header to spring-loaded European-style terminal blocks (Euroblocks), which makes the slot sensors themselves easily interchangeable, as well [3]. The buttons also were wired into spring-loaded Euroblocks. The LCD display was mounted between the buttons and the PCB (Figures 3-6).

Software

The Penny Counter is powered by a Python script that monitors the coin slots

and buttons and displays information to the user on a small LCD display. The complete penny.py script [4] is available at the *Linux Magazine* website.

The script begins with several `import` statements, demonstrating how Python includes libraries with additional functions and methods (Listing 1). The `as` keyword at the end of lines 4 and 6 creates an alias

for easier reference to these modules. The libraries needed for this project are:

- `serial` – Allows access to serial ports. In my case, this is a USB device that emulates a serial port.
- `time` – Gets current time information.
- `curses` – Lets me put text at specific terminal coordinates and captures key-presses. I used this library to simulate the Penny Counter before all of the hardware was built.
- `MySQLdb (mdb)` – Accesses MySQL databases.
- `datetime` – Gets information about the current date and time and supplies functions to perform calculations.
- `RPi.GPIO (GPIO)` – Accesses the 40-pin GPIO header on the Rasp Pi.
- `os` – Talks to the underlying operating system on which Python is running. In my case, it is Raspbian, but it could also be Windows or other Linux distributions, depending on your hardware and host system.

Listing 1: Import Statements

```
01 import serial
02 import time
03 import curses
04 import MySQLdb as mdb
05 import datetime
06 import RPi.GPIO as GPIO
07 import os
08 import subprocess
```

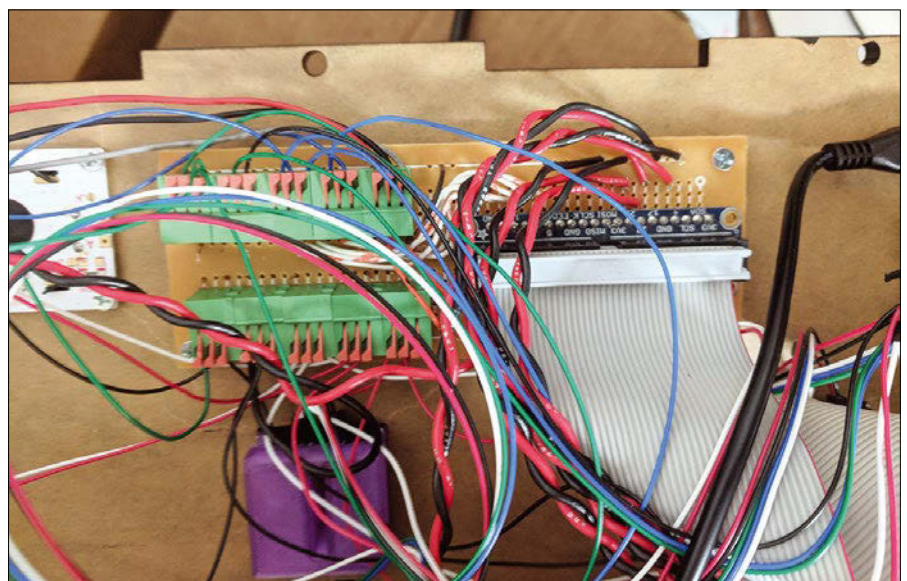


Figure 4: The interconnect PCB. The green blocks on the left are the spring-loaded Euroblock connectors. The blue board on the right is a Rasp Pi GPIO breakout board. The buttons are wired directly to the header because they are not “public-facing” components and therefore not subject to rough handling. The grey ribbon cable connects to the Rasp Pi.

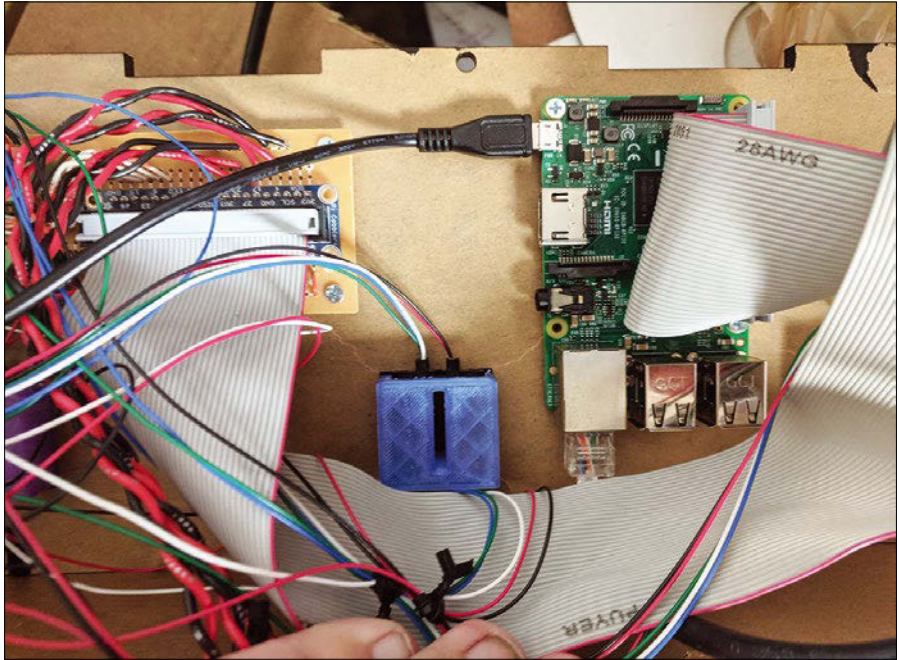


Figure 5: The Rasp Pi and an excellent view of the bottom of the blue funnel. You can just see the network cable in the lower right of the picture. I added the Rasp Pi to the wired network to eliminate connection problems between the Rasp Pi and the museum’s wireless network.

- subprocess – Starts secondary threads for parallel processing.

Display Class

In Python, a class is a group of functions and variables that operate as a set. When a class is created, it is an independent copy that starts with all of the default values of the prototype; then, each copy can be customized as needed. In Listing 2, I use the `display` class to talk to the LCD display. Instead of having to remember all of the hex codes to control the display each time, I can use human-friendly terms (e.g., `clear` and `write`), which make a lot more sense than something like `0xFE` `0x7C` `128`.

The `__init__` function in Listing 2 is called automatically when the class is created. You can do all of your setup here. Notice that all the lines start with `self.`; by default, variables in Python are contained within their parent function.

LCD Display Mapping

The HD44780 LCD driver chip is generically able to control many configurations of text-based LCD displays (up to 80 characters). Specific addresses in the display RAM are mapped to character cells in the display itself. Writing an ASCII-encoded character into the display memory causes that ASCII character to display immediately. However a gap in the end of the first line and the beginning of the second line explains the 64-byte difference (line 17) in the addresses of the first and second lines of 40 bytes each.

Once you exit the function those variables cease to exist, so `self` stores the variables inside the class instead of the function; in this way, other members of the class can access the same values.

The `clear`, `position`, `backlightOn`, and `backlightOff` LCD control functions (Listing 2) work identically, but the data itself is modified for each function. The hex value `0xFE` indicates a control command and `0x7C` indicates a backlight command.

The `clear` command, which clears the display and resets the position to 0,0 (upper left of the display), is hex `0x01`. The `position` command (where to place the cursor) takes the `row` and `column` arguments and calculates the corresponding

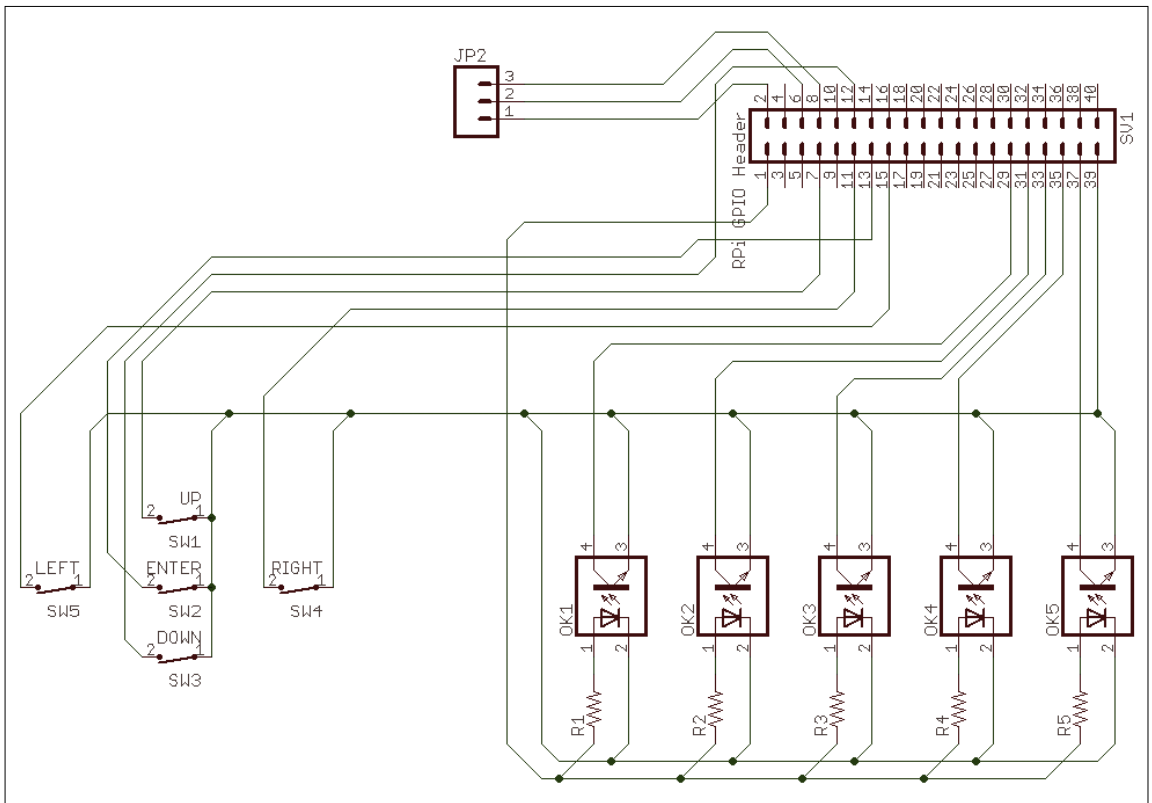


Figure 6: The wiring diagram for the Penny Counter.

Listing 2: The display Class

```

01 class display:
02     def __init__ ( self , port , baud ):
03         self.port = port
04         self.baud = baud
05         self.delay = .01
06         self.serial = serial.Serial ( self.port )
07
08         self.clear()
09
10     def clear ( self ):
11         self.serial.write ( chr ( 0xFE ) )
12         self.serial.write ( chr ( 0x01 ) )
13         time.sleep ( self.delay )
14
15     def position ( self , row , column ):
16         self.serial.write ( chr ( 0xFE ) )
17         self.serial.write ( chr ( column + row * 64 + 128 ) )
18         time.sleep ( self.delay )
19
20     def backlightOn ( self ):
21         self.serial.write ( chr ( 0x7C ) )
22         self.serial.write ( chr ( 157 ) )
23         time.sleep ( self.delay )
24
25     def backlightOff ( self ):
26         self.serial.write ( chr ( 0x7C ) )
27         self.serial.write ( chr ( 128 ) )
28         time.sleep ( self.delay )
29
30     def write ( self , string ):
31         self.serial.write ( string )

```

position. (See the “LCD Display Mapping” box for more information.) On lines 22 and 27, I send either a 157 (backlight on) or 128 (backlight off) for the corresponding action.

All of the functions end with `time.sleep (self.delay)`, which pauses the program for the sleep duration defined on line 5 and gives the LCD enough time to catch up and be ready for the next command.

The `write` function displays text on the LCD and sends `string` out the UART. Without any preceding control characters, the display interprets the string as text to be displayed.

Menu Class

The `menu` class holds all of the major code that allows the Penny Counter to function properly. It integrates the LCD display, the slot sensors, and the buttons that form the user interface.

Listing 3 initializes the `menuItems` list that will become the menu descriptions

Listing 3: Menu Items

```

01     self.menuItems = list()
02     self.menuItems.append ( "SHOW NAME" )
03     self.menuItems.append ( "SHOW HOUR" )
04     self.menuItems.append ( "SHOW MINUTE" )
05     self.menuItems.append ( "LIVE COUNTS" )
06     self.menuItems.append ( "RESET COUNTS" )
07     self.menuItems.append ( "SUBMIT COUNTS" )
08     self.menuItems.append ( "SHUTDOWN" )
09     self.menuItems.append ( "WLANO" )
10     self.menuItems.append ( "ETHO" )
11
12     self.menuIndex = 0
13
14     self.currentShowName = None
15     self.showHour = 10
16     self.showMinute = 00
17     AMPM = "AM"
18
19     self.slotReset = list()
20     self.slotReset.append ( 0 )
21     self.slotReset.append ( 0 )
22     self.slotReset.append ( 0 )
23     self.slotReset.append ( 0 )
24     self.slotReset.append ( 0 )

```

displayed as the user scrolls through and creates each entry with `append`. The `self.menuIndex` holds the menu

item currently being displayed. Initializing it to 0 starts with assigning the

show for which the pennies are to be counted.

Lines 14-17 set a couple of default values that are updated as selections are made in the menu. The time values will be used to build the display strings.

Lines 19-24 set up another Python list to monitor the slot sensors. After creating the list on line 19, I add five zeros,

Listing 4: Setting Up the GPIO

```

01     self.slot1 = 5
02     self.slot2 = 6
03     self.slot3 = 13
04     self.slot4 = 19
05     self.slot5 = 26
06
07     GPIO.setmode ( GPIO.BCM )
08     GPIO.setup ( self.slot1 , GPIO.IN , pull_up_down=GPIO.PUD_UP )
09     GPIO.setup ( self.slot2 , GPIO.IN , pull_up_down=GPIO.PUD_UP )
10     GPIO.setup ( self.slot3 , GPIO.IN , pull_up_down=GPIO.PUD_UP )
11     GPIO.setup ( self.slot4 , GPIO.IN , pull_up_down=GPIO.PUD_UP )
12     GPIO.setup ( self.slot5 , GPIO.IN , pull_up_down=GPIO.PUD_UP )
13
14     self.buttonLeft = 22
15     self.buttonRight = 17
16     self.buttonUp = 4
17     self.buttonDown = 18
18     self.buttonEnter = 27
19

```

Listing 4: Setting Up the GPIO (continued)

```

20     GPIO.setup ( self.buttonLeft , GPIO.IN , pull_up_down=GPIO.PUD_UP )
21     GPIO.setup ( self.buttonRight , GPIO.IN , pull_up_down=GPIO.PUD_UP )
22     GPIO.setup ( self.buttonUp , GPIO.IN , pull_up_down=GPIO.PUD_UP )
23     GPIO.setup ( self.buttonDown , GPIO.IN , pull_up_down=GPIO.PUD_UP )
24     GPIO.setup ( self.buttonEnter , GPIO.IN , pull_up_down=GPIO.PUD_UP )
25
26     self.mode = "MENU"
27     self.showCurrent()

```

Listing 5: countPennies

```

01     def countPennies ( self ):
02         if GPIO.input ( self.slot1 ) == 0 and self.slotSeen [ 0 ] == 0:
03             self.slotCount [ 0 ] += 1
04             self.slotSeen [ 0 ] = 300
05             self.slotReset [ 0 ] += 1
06             if self.menuIndex == 3: self.showCount()
07         else: self.slotReset [ 0 ] = 0
08
09         if GPIO.input ( self.slot2 ) == 0 and self.slotSeen [ 1 ] == 0:
10             self.slotCount [ 1 ] += 1
11             self.slotSeen [ 1 ] = 300
12             self.slotReset [ 1 ] += 1
13             if self.menuIndex == 3: self.showCount()
14         else: self.slotReset [ 1 ] = 0
15
16         if GPIO.input ( self.slot3 ) == 0 and self.slotSeen [ 2 ] == 0:
17             self.slotCount [ 2 ] += 1
18             self.slotSeen [ 2 ] = 300
19             self.slotReset [ 2 ] += 1
20             if self.menuIndex == 3: self.showCount()
21         else: self.slotReset [ 2 ] = 0
22
23         if GPIO.input ( self.slot4 ) == 0 and self.slotSeen [ 3 ] == 0:
24             self.slotCount [ 3 ] += 1
25             self.slotSeen [ 3 ] = 300
26             self.slotReset [ 3 ] += 1
27             if self.menuIndex == 3: self.showCount()
28         else: self.slotReset [ 3 ] = 0
29
30         if GPIO.input ( self.slot5 ) == 0 and self.slotSeen [ 4 ] == 0:
31             self.slotCount [ 4 ] += 1
32             self.slotSeen [ 4 ] = 300
33             self.slotReset [ 4 ] += 1
34             if self.menuIndex == 3: self.showCount()
35         else: self.slotReset [ 4 ] = 0
36
37         if GPIO.input ( self.slot1 ) == 1: self.slotReset [ 0 ] = 0
38         if GPIO.input ( self.slot2 ) == 1: self.slotReset [ 1 ] = 0
39         if GPIO.input ( self.slot3 ) == 1: self.slotReset [ 2 ] = 0
40         if GPIO.input ( self.slot4 ) == 1: self.slotReset [ 3 ] = 0
41         if GPIO.input ( self.slot5 ) == 1: self.slotReset [ 4 ] = 0
42
43         if self.slotReset [ 0 ] >= 5: self.stuck ( self.slot1 , 0 )
44         if self.slotReset [ 1 ] >= 5: self.stuck ( self.slot2 , 1 )
45         if self.slotReset [ 2 ] >= 5: self.stuck ( self.slot3 , 2 )
46         if self.slotReset [ 3 ] >= 5: self.stuck ( self.slot4 , 3 )
47         if self.slotReset [ 4 ] >= 5: self.stuck ( self.slot5 , 4 )

```

which increments repeatedly when the slot sensor is blocked (i.e., a penny is stuck in the slot).

Listing 4 sets up the GPIO to monitor the slot sensors and user interface buttons. Lines 1-5 define the GPIO pins to which each sensor is attached; then, line 7 sets the numbering mode of the GPIO pins. Lines 8-12 configure each pin with `GPIO.setup`, which takes three arguments: the GPIO pin of interest, the direction of pin traffic (`GPIO.IN` defines it as an input), and whether it is a pull-up or pull-down resistor (`pull_up_down=GPIO.PUD_UP` specifies a pull-up resistor connected to the input). These settings are the electrical equivalent of a default value. In the absence of an external signal, the input will read as high, or logic 1.

For the user interface buttons, lines 14-18 mirror lines 1-5, and lines 20-24 mirror lines 8-12 for the slot sensors.

Finally, lines 26 and 27 initialize the user interface so that button presses are processed relative to the main menu and update the LCD display to reflect any changes made with button presses (or the initial menu).

Counting Pennies

The `countPennies` function (Listing 5) monitors the slot detectors and increments the counts whenever a penny is deposited into the counter. Each block of code is identical, except for the slot number and associated indexes, so I'll just describe the first one.

First, I check to see whether the sensor is currently blocked by a penny (line 2). `GPIO.input` returns 1 for clear or 0 for blocked. I also check `self.slotSeen`. If this is not 0, then I've already seen this penny. Assuming both values are 0, I process the count.

On line 3, I increment the counter by one; on line 4, I assign `self.slotSeen` the value 300; then, I increment `self.slotReset` (line 5). I set it to 300 based on experimentation with the hardware. This is how long (in loop iterations) to ignore the slot once any given penny has been detected. This is the time for the penny to pass through the slot. Just like any switch or detector it will bounce momentarily as the penny enters (and leaves) the slot. The penny is counted when initially detected and then all fluctuations are ignored until the penny has

Listing 6: The stuck Function

```

01 def stuck ( self , slot , slotNumber ):
02     self.LCD.clear()
03     self.LCD.write ( "STUCK PENNY" )
04     self.LCD.position ( 1 , 0 )
05     self.LCD.write ( "SLOT " + str ( slotNumber + 1 ) )
06
07     while 1:
08         if GPIO.input ( slot ) == 1: break
09         self.LCD.backlightOff()
10         time.sleep ( .25 )
11         self.LCD.backlightOn()
12         time.sleep ( .25 )
13
14     self.slotCount [ slotNumber ] -= 5
15     self.LCD.backlightOn()
16     self.showCount()

```

passed completely through. Then it's ready for the next penny to repeat the process. Finally, if the menu item currently being displayed is 3, I call `self.showCount` to update the display with the current counts (line 6).

If either condition from the `if` on line 2 wasn't 0, I clear `self.slotReset` on line 7. If this number gets too high, the counter assumes a penny is stuck and alerts the operator.

Lines 37-41 check to see that each slot sensor is clear (returning 1). If so, the associated `self.slotReset` is reset to 0; however, if the return value is greater than 5, `self.stuck()` is called (lines 43-47).

Stuck Pennies

The `stuck` function (Listing 6) alerts the staff that a penny is stuck in a slot by displaying the slot number in which a penny is stuck and flashing the backlight. Line 2 clears the entire display, instead of just the second line, as in the other cases; then, I output the *STUCK PENNY* message (line 3), move to the second line of the display (line 4), and show the offending slot number (line 5).

Now that the display is showing the proper warning message, I get the user's attention by flashing the backlight in an infinite loop (line 7). In each iteration, I check `GPIO.input(slot)` (line 8). If it equals 1, the jam has been cleared and I break out of the loop; otherwise, I turn the backlight off and

on, with a quarter second pause between states (lines 9-12).

Once the jam is cleared, the false counts are decremented, the backlight is turned back on, and the display is returned to normal.

Display Update

In Listing 7, the `showCount` utility function rewrites the LCD display with the current slot counts by positioning the cursor on

the second line of the LCD and generating the count string with `.format`. (Note that I do not have to clear the display because I rewrite the entire 16-character line each time.) The `resetCount` utility

function sets all slots to 0 counts in a for loop, which is handy for testing the counter at the beginning of the day or after maintenance.

Populating MySQL

Once the pennies have been collected, the operator can submit the count, which commits it to the database. In Listing 8, lines 2 and 3 clear the display then show a status message before attempting to connect to the database.

Line 5 gets the current date, and line 7 generates a date-time string in the appropriate format for MySQL. I connect to the database and ask for a dictionary cursor before generating a SQL statement by using `format` to concatenate all of the current slot counts into a SQL query (lines 9-11). Finally, I submit the query and `commit` the change.

Ready to Go

Listing 9 is where the code actually begins execution. Line 1 initializes an instance of

Listing 7: showCount

```

01 def showCount ( self ):
02     self.LCD.position ( 1 , 0 )
03     self.LCD.write ( "{0:02d} {1:02d} {2:02d} {3:02d} {4:02d}".format
04         ( self.slotCount [ 0 ] , self.slotCount [ 1 ] , self.slotCount [ 2 ] ,
05           self.slotCount [ 3 ] , self.slotCount [ 4 ] ) )
06
07     def resetCount ( self ):
08         for i in range ( 5 ):
09             self.slotCount [ i ] = 0

```

Listing 8: Database Commit

```

01 def submitCount ( self ):
02     self.LCD.position ( 1 , 0 )
03     self.LCD.write ( "Submitting..." )
04
05     today = datetime.datetime.today()
06
07     mysqlDT = "{5:d}-{3:d}-{4:02d} {0:d}:{1:02d}:00".format ( self.showHour ,
08         self.showMinute , "" , today.month , today.day , today.year )
09
10     con = mdb.connect('DB_SERVER_IP', 'DB_USERNAME', 'DB_PASSWORD', 'DB_NAME');
11     cursor = con.cursor(mdb.cursors.DictCursor)
12
13     sql = "INSERT INTO `pennies` ( `Show` , `ShowDateTime` , `Slot1` , `Slot2` ,
14         `Slot3` , `Slot4` , `Slot5` ) VALUES ( \"{0}\" , \"{1}\" , {2} , {3} , {4} ,
15         {5} , {6} )".format ( self.currentShowName , mysqlDT , self.slotCount [ 0 ] ,
16         self.slotCount [ 1 ] , self.slotCount [ 2 ] , self.slotCount [ 3 ] , self.
17         slotCount [ 4 ] )
18
19     cursor.execute ( sql )
20
21     con.commit()

```

the `display` class and passing in the address and baud rate of the serial port I want to open. This instance of the `display` library now talks via `/dev/ttyS0`. I could

add a second display connected to a different serial port and initialize another instance of the class to run two displays independently if I desired.

On line 2, I create an instance of the `menu` class and pass a reference to `LCD` that I set up earlier. Line 3 then draws the initial LCD display.

Lines 5-9 set up the `curses` library, which is designed to display text on character cell displays. I used it while developing the software before all of the hardware was ready for an embedded design. Even after the hardware was finished, I kept the `curses` display so I could plug a monitor into the Rasp Pi and see debug information. Table 1 describes the roles of the `curses` functions used here. On line 11, I initialize `idleCount` to 0 to shut off the LCD backlight after a period of no activity.

The main loop of the program starts on line 13. On line 14, it checks to see whether the left interface button has been pressed and that it is not currently being ignored to wait for bounces to die down. If both conditions are OK, it turns on the LCD backlight, calls the `LCDmenu.left()` function, and sets `LCDmenu.bounceCount [0]` to 50. Lines 18-33 do the same thing for the other four buttons.

Note that I am calling the function with `LCDmenu` instead of `self`. Because I am calling the function from outside the class, I reference its functions via the instance – the variable I stored it in when I called the class name on line 2.

Listing 9: Beginning Execution

```
01 LCD = display ( "/dev/ttyS0" , 9600 )
02 LCDmenu = menu ( LCD )
03 LCDmenu.showCurrent()
04
05 screen = curses.initscr()
06 curses.noecho()
07 curses.cbreak()
08 screen.keypad ( 1 )
09 screen.nodelay ( 1 )
10
11 idleCount = 0
12
13 while 1:
14     if GPIO.input ( LCDmenu.buttonLeft ) == False and LCDmenu.bounceCount [ 0 ] == 0:
15         LCDmenu.LCD.backlightOn()
16         LCDmenu.left()
17         LCDmenu.bounceCount [ 0 ] = 50
18     elif GPIO.input ( LCDmenu.buttonRight ) == False and LCDmenu.bounceCount [ 1 ] == 0:
19         LCDmenu.LCD.backlightOn()
20         LCDmenu.right()
21         LCDmenu.bounceCount [ 1 ] = 50
22     elif GPIO.input ( LCDmenu.buttonUp ) == False and LCDmenu.bounceCount [ 2 ] == 0:
23         LCDmenu.LCD.backlightOn()
24         LCDmenu.up()
25         LCDmenu.bounceCount [ 2 ] = 50
26     elif GPIO.input ( LCDmenu.buttonDown ) == False and LCDmenu.bounceCount [ 3 ] == 0:
27         LCDmenu.LCD.backlightOn()
28         LCDmenu.down()
29         LCDmenu.bounceCount [ 3 ] = 50
30     elif GPIO.input ( LCDmenu.buttonEnter ) == False and LCDmenu.bounceCount [ 4 ] == 0:
31         LCDmenu.LCD.backlightOn()
32         LCDmenu.enter()
33         LCDmenu.bounceCount [ 4 ] = 50
34     else: idleCount += 1
35
36     if idleCount == 90000:
37         idleCount = 0
38         LCDmenu.LCD.backlightOff()
39
40     while GPIO.input ( LCDmenu.buttonLeft ) == False: pass
41     while GPIO.input ( LCDmenu.buttonRight ) == False: pass
42     while GPIO.input ( LCDmenu.buttonUp ) == False: pass
43     while GPIO.input ( LCDmenu.buttonDown ) == False: pass
44     while GPIO.input ( LCDmenu.buttonEnter ) == False: pass
45
46     for i in range ( 5 ):
47         if LCDmenu.bounceCount [ i ] > 0: LCDmenu.
48             bounceCount [ i ] -= 1
49
49     LCDmenu.countPennies()
```

Table 1: Curses Functions

Statement	Function
<code>screen = curses.initscr()</code>	Initialize the <code>curses</code> library. <code>screen</code> will be the reference to the display to which I'm writing.
<code>curses.noecho()</code>	Do not echo characters to the screen as they are typed.
<code>curses.cbreak()</code>	Turn off input buffering and pass all character codes directly to the input function without processing.
<code>screen.keypad (1)</code>	Convert escape sequences generated by some special keyboard keys to unique scan codes so that special keys can be handled as regular keystrokes.
<code>screen.nodelay (1)</code>	Do not wait for input if none is available.

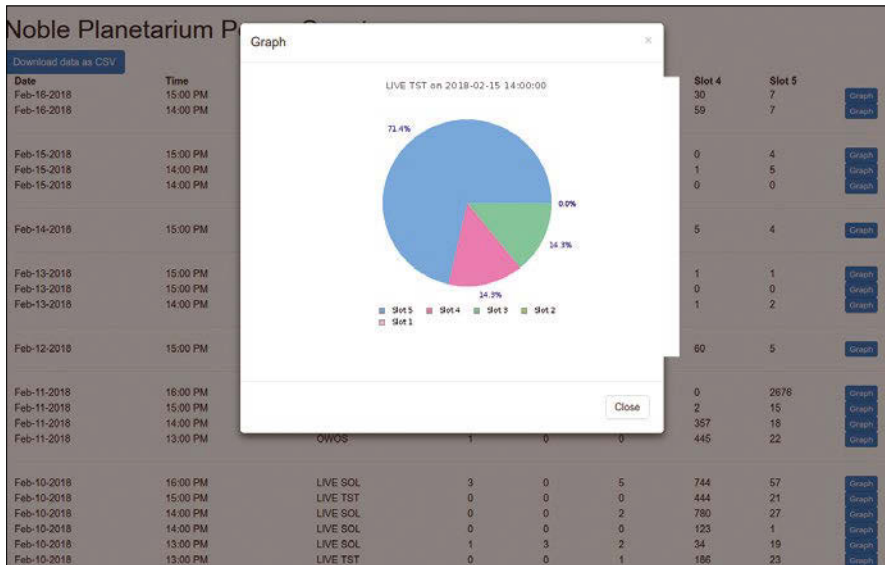


Figure 7: A pie chart showing the distribution of pennies.

Line 34 is the catch-all for the string of if/elif statements that monitor the buttons. If no button has been pressed, I increment the idle counter. If it reaches 90,000, I reset it to 0 and turn off the LCD backlight.

Each button has its own while loop on lines 40-44. The loop continues to run until its associated button is released, preventing a button press from being processed more than once.

Lines 46-47 decrement the bounceCount counters if they are greater than 0, and line 49 calls LCDmenu.countPennies() to monitor the penny slots.

After the Show

After the pennies have been counted, you'll naturally want to see how well you performed. Once the usher has submitted the show's counts, it is available immediately on the planetarium's web dashboard.

The pennyGraph.php module (Listing 10) generates a pie chart of the penny ratings (Figure 7). Line 1 tells Apache to let the PHP

interpreter handle this block of code, and line 3 includes db.php, which has connection information for the database server.

Pennies in the Bank

Overall, the Penny Counter has performed well. Aside from the occasional

penny getting jammed in the slot, it has successfully counted to the entire capacity of the theater (80 people).

One unanticipated outcome of the design that was discovered during beta testing was that children are more interested in putting their penny in a slot of their favorite color than they are in rating the show. We solved this problem by placing black covers on all of the funnels and fronting the counter with a sign to block the funnel colors. Reprinting all the funnels in the same color will eliminate this problem. ■■■

Info

- [1] Blender 3D Modeling and Creative Suite: <https://www.blender.org>
- [2] PolyPrinter 3D printer: www.polyprinter.com
- [3] Photomicrosensor: https://www.mouser.com/datasheet/2/307/en-ee_sx3096_w11_4096_w11-805875.pdf
- [4] penny.py script: <ftp://ftp.linux-magazine.com/pub/lists/linux-magazine.com/216/>

Listing 10: Pie Chart

```

01 <?php
02
03     include "db.php";
04
05     $sql = "SELECT * FROM `pennies` WHERE `ID` = ?";
06     $query = $db->prepare ( $sql );
07     $query->bindValue ( 1, $_REQUEST [ 'pennyID' ], PDO::PARAM_INT );
08     $query->execute();
09     $show = $query->fetch ( PDO::FETCH_ASSOC );
10
11     require_once ('jpgraph/src/jpgraph.php');
12     require_once ('jpgraph/src/jpgraph_pie.php');
13
14     $data = array(intval ( $show [ 'Slot1' ] ), intval ( $show [ 'Slot2' ] ), intval
15     ( $show [ 'Slot3' ] ), intval ( $show [ 'Slot4' ] ), intval ( $show [ 'Slot5' ] ) );
16
17     $graph = new PieGraph(600,400);
18     $graph->SetShadow();
19
20     $graph->title->Set ( $show [ 'Show' ]." on ".$show [ 'ShowDateTime' ] );
21
22     $p1 = new PiePlot($data);
23     $p1->SetLegends ( array ( "Slot 5" , "Slot 4" , "Slot 3" , "Slot 2" , "Slot 1" ) );
24
25     $graph->Add($p1);
26     $graph->Stroke();
27 ?>

```



MakerSpace

Making open hardware a reality

The Rise of Open Hardware

Changes in funding, manufacturing, and technology have helped move open hardware from an idea to reality.

By Bruce Byfield

Like free software, open hardware was an idea before it was a reality. Until developments in the tech industry caught up with the idea, open hardware was impractical. Even now, in 2018, open hardware is at the stage where free software was in about 1999: ready to make its mark, but not being developed by major hardware manufacturers.

As late as 1999, Richard M. Stallman of the Free Software Foundation (FSF) downplayed the practicality of what he called free hardware. In “On ‘Free Hardware’” [1], Stallman suggested that working on free hardware was “a fine thing to do” and said that the FSF would put enthusiasts in touch with each other. However, while firmware is just software, and specifications could be made freely available, he did not think that either would do much good because of the difficulties of manufacturing, writing:

We don't have automatic copiers for hardware. (Maybe nanotechnology will provide that capability.) So you must expect that making fresh a copy of some hardware will cost you, even if the hardware or design is free. The parts will cost money, and only a very good friend is likely to make circuit boards or solder wires and chips for you as a favor.

Although the comments on Stallman's short essay mention an early attempt to clone a Pentium II, most of the responses agreed with his conclusions. The idea of open hardware continued to float around, but for another decade almost no progress was made. For many years, the greatest signs of interest were the release of refurbished computers and mobile devices with proprietary firmware replaced – a useful, but self-limiting trend.

However, during that decade, technological trends began to lay the groundwork for open hardware. During that time, free software changed from a fringe idea to a technology mainstay. Other endeavors began to imitate free software's licensing and collaborative development. For instance, the open access movement began to advocate making published academic papers freely available, instead of restricting readership to the subscribers of expensive journals [2]. As open source software proved itself a valuable tool for slashing development time for commercial software and other products, the idea of open hardware gradually became less far-fetched.

Still, open hardware would not have become more commonplace without solutions to the problems observed by Stallman and others. During the early

years of the millennium, these solutions gradually fell into place.

The Crowdfunding Alternative

As Stallman pointed out, although free software can be developed for no more than the cost of a computer and an Internet connection, manufacturing hardware can be expensive. Raw material, hiring experts, and storing and shipping hardware add up to a cost that is beyond most people's means. In fact, they are beyond the means of many small companies, as well. These costs also mean that small, practical equipment or products with a limited market may not be profitable enough to be worth developing and bringing to market.

However, these traditional manufacturing economics began to be challenged with the popularity of crowdfunding. With crowdfunding, potential developers can test the audience for a product by asking for donations to fund development. More importantly, developers do not have to have the money themselves or sell some control of their efforts to traditional investors. As a bonus, crowdfunding seems to work best with the transparency and interaction typified by free software, where many open hardware developers have been active for years.

Admittedly, more crowdfunding campaigns fail than succeed. In 2014, I estimated that only 7.5 percent of Indiegogo campaigns succeeded [3]. However, targeted campaigns have a higher success rate, and expertise has accumulated to improve the odds. One site predicts campaign success from a dozen questions [4], and sites like Crowd Supply [5] have greatly increased the success rate of open hardware by working closely with would-be entrepreneurs. Although considerable effort and time go into successful crowdfunding campaigns, funding of open hardware is now more of a possibility.

Perhaps the largest problem that remains with crowdfunding is that campaigns for open hardware seem more likely to succeed if the target is under \$2 million. More can be raised, as proven by the Ubuntu Edge project [6] – an open source software campaign, although not an open hardware one. The Ubuntu Edge phone received

\$13 million in pledges, but it also fell well short of its goal of \$32 million, the level of funding that might be raised for such a project through venture capital. Possibly, open hardware is for now limited to small companies, even with crowdfunding.

The Supply Chain Changes

At the same time as crowdfunding was getting started, alternative manufacturing was being established. In the last decade, the do-it-yourself Maker Movement [7] has taken form, increasing the average developer's access to expertise. Developers who lack experience themselves can often find the information they need online or at a local meeting.

The Maker Movement has also given rise to a supporting supplier cottage industry. Originally aimed at hobbyists, sites like Adafruit [8] have proven equally useful for beginning entrepreneurs, especially since they sell smaller quantities than some traditional suppliers.

One of the major products sold by these new suppliers are single-board controllers, such as the Arduino [9] and Raspberry Pi [10]. Over the last few years, these controllers have gradually become more powerful, so much so that the CPUs are available for less than \$50. Should a single board be insufficient for a product, others can be chained as needed. Arduino boards are particularly useful for open hardware developers because they include the software for flashing firmware. Additionally, both Arduino and Raspberry Pi have sites for community exchange of knowledge and specifications, much like open source projects.

Still another development that has aided open hardware is the creation of 3D printers. Now, for a few thousand dollars, developers can not only produce their own parts, but do so on demand, which means that they do not have to keep track of large inventories or find a place to store them. Parts produced by 3D printers are not always durable or heat resistant, yet they can sometimes free developers from dependency on traditional manufacturers.

Open hardware development remains more expensive than free software development and probably always will. However, today, the infrastructure is beginning to support it.

Since about 2010, open hardware has become steadily more possible to an extent that was unimaginable at the turn of the millennium.

Inexperience vs. Monopolies

Even with more favorable conditions, another obstacle remains: bringing a product to market. Bringing any new product to market is difficult, but delivering open hardware is even more so because of the simple fact that most open hardware entrepreneurs have a technical rather than a business background.

This situation has been a constant theme with open hardware. For instance, in 2012, Aaron Seigo of KDE fame began attempts to produce Vivaldi, a KDE-based tablet [11]. Two years and \$200,000 of his own money later, he abandoned the attempt. Even a last ditch attempt to salvage something from his efforts by producing an engineering board failed.

Vivaldi's troubles began with the impracticality of manufacturing in Europe. Financially, manufacturing in Asia seemed a solution. Unfortunately, communication problems required constant, hands-on supervision. For example, manufacturers were not oriented to free software and hardware and would replace free-licensed parts with cheaper proprietary ones without bothering to inform Seigo.

Additionally, the small scale of Vivaldi's production run was of limited interest to most manufacturers, who were used to runs of several hundred thousand units. At times, parts were unavailable, having been reserved for large manufacturers like Apple.

Similar problems and delays are reported by Luke Leighton, who has been struggling since 2016 to bring a line of recyclable computers to production [12]. Keyboardio's Jesse Vincent and Kaia Dekker did manage to bring their open hardware keyboards to market, but it took several years of constant trips and setbacks to do so, as described in detail on the company blog [13]. Among the memorable moments Vincent and Dekker describe is a supplier's inability to understand their refusal to ship substandard parts, and their discovery that no business in China was possible during the Lunar New Year celebrations.

Some manufacturing problems can be overcome by persistence. But the problem of near monopolies in manufacturing hardware can only be worked around. Even more importantly, as Seigo observed, “There isn’t much business experience in the free software communities,” so such issues seem likely to continue.

Is the Future of Hardware Open?

Despite all the challenges, more open hardware products probably exist today than ever before. Keyboardio, for instance, has established itself as an innovative keyboard manufacturer (Figure 1) and is currently in the early stages of developing other products. Similarly, Purism has established itself with a line of high-end laptops (Figure 2) and has attracted attention for some months with its development of a free phone. Other open hardware manufacturers may not have launched the next IBM or Apple, but their small businesses do appear to be enjoying modest successes. For the most part, the long-standing challenges of open hardware production appear to have been alleviated in the last few years, even if they have not been overcome.

The next challenge for open hardware production is to scale and become a known presence in manufacturing. This next stage may already be happening in prosthetics, where open hardware can produce alternatives for a fraction of the cost of proprietary methodologies (Figure 3). However, to



Figure 1: After almost three years of struggling with production problems, Keyboardio has become an open hardware success story.

date, most open hardware makers seem to have a boutique niche, producing high-quality products at high prices for a relatively small number of discerning clients.

Perhaps the Internet of Things (IOT) will alter this situation. To be fully realized in the short time that is pre-

dicted, the IOT requires millions of gadgets as soon as possible, and conventional manufacturing is unlikely to keep up with the demand for product development. Now, all that can be said is that, one way or the other, open hardware is on its way to becoming a reality at last. ■■■

Info

- [1] “On ‘Free Hardware’”: <https://www.linuxtoday.com/infrastructure/1999062200505NWL/>
- [2] Open access: https://en.wikipedia.org/wiki/Open_access
- [3] Crowdfunding success rate: <https://www.datamation.com/applications/crowdfunding-and-open-source.html>
- [4] Predicting crowdfunding success: <http://crowdfunding.io/>
- [5] Crowd Supply: <http://www.crowdsupply.com>
- [6] Ubuntu Edge: <https://www.indiegogo.com/projects/ubuntu-edge#/>
- [7] The Maker Movement: https://en.wikipedia.org/wiki/Maker_culture
- [8] Adafruit: <https://www.adafruit.com/>
- [9] Arduino: <https://www.arduino.cc/>
- [10] Raspberry Pi: <https://www.raspberrypi.org/>
- [11] Vivaldi tablet: <https://lwn.net/Articles/606100/>
- [12] Recyclable bamboo computers: <https://www.crowdsupply.com/eoma68/micro-desktop/updates>
- [13] Keyboardio blog: <http://blog.keyboard.io/>



Figure 2: Purism has become known for its line of high-end, open hardware laptops.



Figure 3: The high demand for prosthetics make them a field where open hardware has made inroads. Shown here is a design from Social Hardware.

A computer is a tool for practical problems, and yet, sometimes software gets so full of high-end features that we lose the practical in the versatile. This month we feature a household finance tool called Eqonomize that keeps the emphasis on the personal experience, without a lot of unnecessary features that only a business or corporate user would need. Elsewhere in this month's LinuxVoice, we look at the Polo file manager, which is only a year old but offers some impressive new features.

Our tutorials this month offer lessons in practical programming. We show you how to create a custom tool that will access the sensors on your cell phone, and we build a script that simplifies the tedious task of manually deleting email attachments.



Image © Alexandr Moroz, 123RF.com

LINUXVOICE ▶

Doghouse – Continuity	69
<i>Jon "maddog" Hall</i>	
Developing an exit strategy can ensure continuity for a FOSS project.	
Eqonomize	70
<i>Erik Bärawaldt</i>	
Most accounting programs for Linux are aimed primarily at businesses. Eqonomize focuses on personal use offering a smart solution for getting a handle on your household budget.	
FOSSPicks	74
<i>Graham Morrison</i>	
Graham reviews Thunderbird 60, Stress-Terminal UI, Taskbook, SolveSpace, Star Ruler 2, and more!	
Polo File Manager	80
<i>Christoph Langner</i>	
If you expect more from a file manager than the ability to move files, Polo might be for you.	
Tutorials – Cordova Sensor	85
<i>Paul Brown</i>	
A new universal standard makes it easy to create mobile applications that access your phone's sensor data.	
Tutorials – Attachment Extraction	90
<i>Marco Fioretti</i>	
Retrieving email attachments manually can be a tedious task. We'll show you a script that fetches attachments automatically and can even save the email as a text file.	





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

Expand your Linux skills with:

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

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

Subscribe now!
shop.linuxnewmedia.com/subs

GET IT NOW!
FAST DELIVERY WITH OUR PDF EDITION

MADDOG'S DOGHOUSE



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

Developing an exit strategy can ensure continuity when it comes to FOSS projects. BY JON "MADDOG" HALL

Passing the baton

In a conversation with a chief technical officer (CTO) of a web-hosting company the other day, he mentioned that a large piece of FOSS that his company depended on was being removed from the Linux kernel, because there was no one to develop and support it. While he did not really come out and make the accusation, I inferred from his remark that he felt parts of the Linux kernel were not receiving proper attention, given their importance to the FOSS world.

This does happen, and it has been noted in the past. Typically at the last minute, the FOSS community will belly up to the bar, and a few more developers will be found. Or some company will fund the primary developers, so they can spend full time supporting the software that they had been supporting and developing in their spare time.

Yes, the FOSS community has a coverage problem from time to time. Yes, sometimes software that we depend on goes stagnant, with the developers either leaving the project or sometimes (unfortunately) dying. This is why software projects need to spend as much time "building community" around their projects as they do writing code. The project leaders have to attract new talent, both building enthusiasm for the project, as well as building expertise in those who will become the new architects and leaders of tomorrow.

The fact that closed source software has the same problem did not occur to my CTO until I mentioned it, and the look on his face was priceless. I pointed out that not only do whole companies (sometimes very large ones) disappear and their products become useless, but companies drop product lines or features when they are deemed "not profitable" (or not profitable enough).

The difference between "open" and "closed" projects is that typically "open" projects give warning signs that are visible for those who know how to look for the signs. The rate of code submissions, the last date of code release, the number of developers, activity on the mailing list – all can be indications of a project's strength. Of course, to ask "Mom&Pop" to do this analysis is typically beyond their capabilities, but the CTO could certainly have done that task.

Also in the case of the CTO, if his business depended on the functionality that was being dropped from the kernel, he could have volunteered resources to keep that functionality alive or

switched to a long-term kernel that had the functionality in it while he devised a plan to recover. The functionality being dropped from *future* kernels did not mean it would suddenly disappear from *existing* kernels.

Still, the issue of "coverage" is an important one and touches more than just the software and hardware itself.

In the "good old days," a mainframe computer typically had only a few applications on it at one time. The mainframe might be dedicated to a specific task or tasks, and the number of layers and dependencies in "the stack" were small. In the modern world of servers and the cloud, the number of dependencies seems to grow without bound, and a "dependency nightmare" pushes us toward containers, which allow the nightmare to grow even faster. To try and keep up with all of the different pieces of code that your business depends on is a difficult, if not impossible, task.

Part of this issue is the graying of the FOSS development world. When I started with the Linux project, I was "only" 43 years old. Linus was 24 when I met him. Next year, I will be 69 (itself an interesting number), and Linus will be 50. There were discussions a long time ago about what would happen to Linux if something happened to Linus, and he put the issue to rest, pointing out that many of his lieutenants would be perfectly capable of taking over from him. Nevertheless, the conferences and meetings that used to have sandal-wearing twenty-somethings now have graybeards attending. These graybeards need to work extra hard in "passing the baton."

It is not the Linux kernel that worries me, but the hundreds of thousands of FOSS projects on which we all depend. Each project leader should develop an exit strategy with regards to their project. Who takes over if something happens to the project leader? Who controls the domain name and other intellectual property of the project? Does anyone know all the passwords necessary to keep developing the project? Is someone identified (even tentatively) who would be able (and willing) to take over the project? Does this person have what they need to take over? Waiting until the project absolutely needs this information is often too late.

Go out and find those Padawans and work hard to develop them into the next generation of Jedi Knights. ■■■

Penny Pincher

Most accounting programs available for Linux are aimed primarily at businesses. Eqonomize focuses on personal use, offering a smart solution for getting a handle on your household budget. **BY ERIK BÄRWALDT**

Although many accounting programs populate the Linux universe, most provide features that private households do not need. Additionally, these applications usually rely on a database in the background, which requires time-consuming installation and configuration.

Eqonomize [1], a KDE program specially designed for private households, offers a different approach that dispenses with unnecessary features and brings order to personal accounting in no time at all. The program, published under the GPLv3, can be found as a variant for 64-bit systems, as well as a DEB package for Ubuntu and its derivatives, on the Eqonomize website [2].

Source code for manual compilation and an AppImage that runs on most distributions and does not require installation also are available. To start AppImage, which weighs in at almost 30MB, you need to assign execution rights to it the after downloading by entering

```
chmod +x Eqonomize-1.3.AppImage
```

To start the program with a simple mouse click, move the AppImage to a suitable subdirectory (e.g., /opt/) to integrate it into the system's menu structure.

Ready, Steady, Go!

On startup, Eqonomize opens a window with a center section that shows a large list area of various accounts and their balances. Above is a buttonbar and the obligatory menu. In the bottom area, you can set the display period and, if enabled, the budget data (Figure 1).

First, open the *Accounts / New Account* dialog (Figure 2) and create a new account, which then appears in the main window's *Accounts & Categories* tab. In addition to space for entering the necessary data, the Accounts dialog lets you select an account-specific currency.

Figure 1: The clearly structured program window is largely self-explanatory.

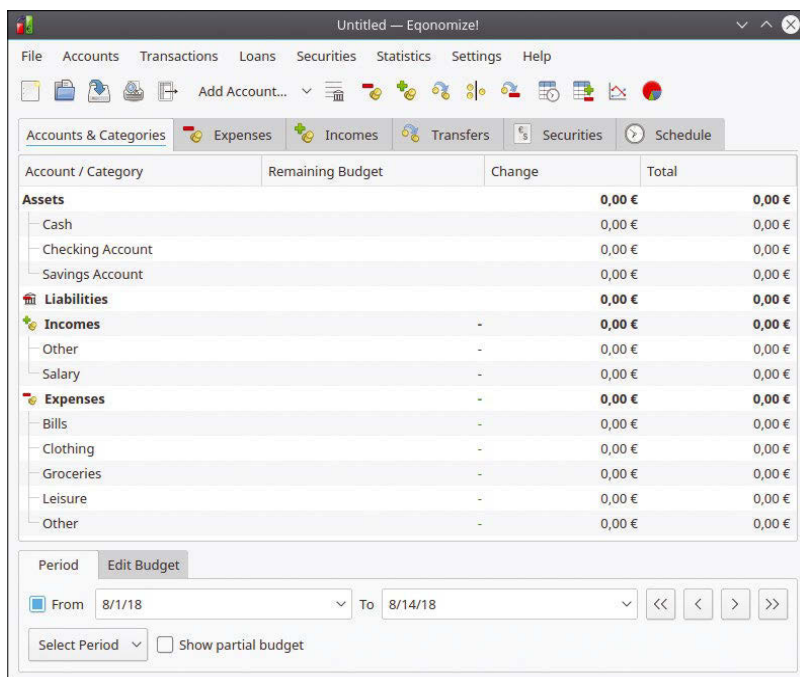
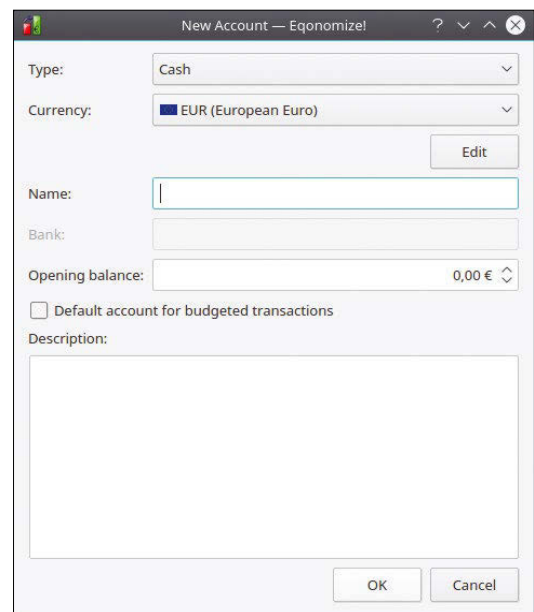


Figure 2: Create new accounts in a simple dialog.



After entering the initial inputs, the data immediately appears in the main window's list overview. In the *Assets* group, you only see a few account types in the basic setting; you will need to add most of the additional account types.

In the *New Account* dialog, you can select additional types such as *Credit Card* or *Securities*. Alternatively, right-click the *Assets* group and select *Add Account* from the context menu. To edit an existing account, right-click on it to open the context menu and select *Edit*.

In the *Liabilities* group, you then enter liabilities, such as loans or mortgages, against which you make regular payments. Since this group does not contain a category, you can right-click on the group name to open the context menu and choose *Add Loan*. Then enter the relevant data.

The *Incomes* and *Expenses* groups deal with recurring business transactions. The *Incomes* group already contains the *Salary* account and a category named *Other*. The *Incomes* group can also be customized, for example, to include an account for receiving regular rental payments.

You can create a new subcategory by right-clicking on the desired category and selecting *Add Category* from the context menu. Creating such subcategories is particularly useful if, for example, you receive your salary or fee payments from different sources and want to enhance the *Salary* category by adding corresponding subcategories.

The *Expenses* group, which already contains some categories for daily needs, is much more extensive in the default setting. However, categories for capital goods and for irregular expenses, such as home repair or renovation, are missing. Similar to the *Incomes* group, you can also enter these costs either by creating new categories or allocating them to subcategories.

Amounts

To calculate your income and expenses correctly, use the two tabs *Expenses* and *Incomes* in the main window. The dialog window for both of these tabs splits into two areas. In the list area, individual transactions are listed based on specifications, such as a time period. In the input area at the bottom of the window, you can enter the transaction along with information such as recipient, category, amount, and the debited account.

Note that *Cash* is also an account type. Clicking on *Add* moves the transaction to the main *Accounts & Categories* section; at the same time, the tool recalculates the corresponding balances (Figure 3).

Entries only appear in the *Expenses* and *Incomes* list view when you click on the *Filter* tab next to *New/Edit Expense* or *New/Edit Income* tab and search for the corresponding expenses

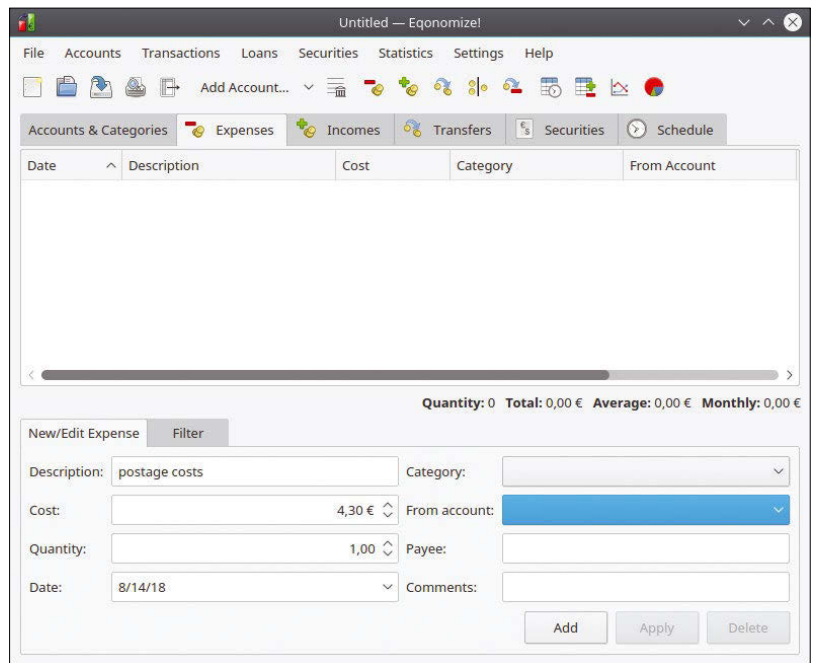


Figure 3: The list view displays all processes clearly.

or incomes according to the specified criteria. These then appear in the list above for the defined period.

To exchange amounts correctly between individual account balance sheets, use the *Transfers* tab. For example, you can transfer funds from your checking account to your cash account. The *Transfers* tab is divided into a list and an input area in the same way as the other options; *Filter* lets you display specific transfers. Transactions then immediately appear in the main window allocated appropriately so that you always have an overview of your account balance sheets.

Securities

The *Securities* tab lets you manage a securities account. First, you need to create security groups. To do this, click on the *New Security* button above the empty list view.

In the input dialog, you can then specify whether the security is a stock or a mutual fund. The *Other* category can be used for other securities, such as cooperative shares. The input dialog also expects you to create an account that the software uses to process transactions.

I recommend creating a stock market account to specifically use for trading securities. To purchase or sell securities, use the *New Transaction* button. The dialog lets you buy, sell, and exchange shares, as well as post dividends.

In the corresponding dialogs, you can also specify a target account: For example, dividends can be allocated to your checking account. In the list view, you can also see security account fees, giving you an exact overview of your investments (Figure 4).

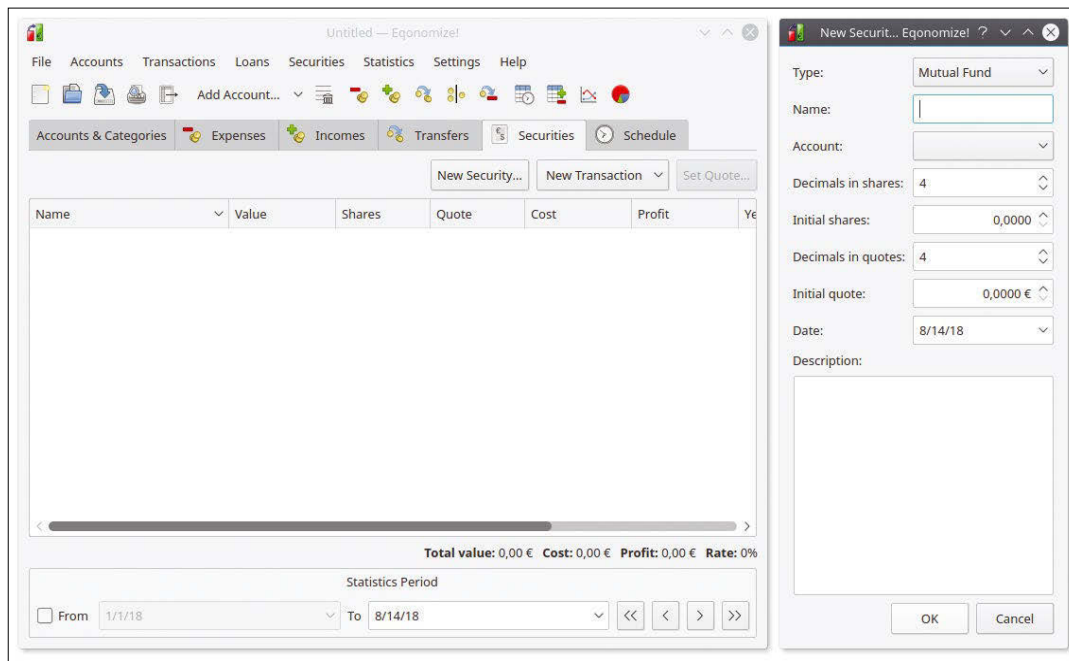


Figure 4: Eqonomize also manages any existing securities in a simple but efficient manner. Dividends can be transferred directly to your checking account if desired.

Deadlines

The *Schedule* tab lets you plan regular transactions in advance. In this way, you can define recurring payments and income or define a loan due date.

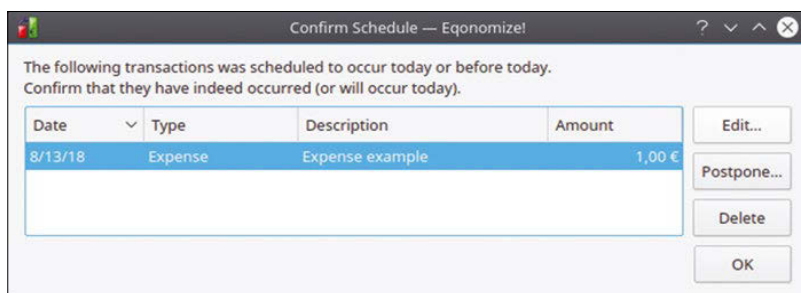
Eqonomize then lists these transactions, but does not perform them automatically. When a specified payment due date arrives, a small reminder window opens if you launch the software on the due date. It informs you of the planned transaction, which you then either confirm or reject (Figure 5). The system incorporates confirmed transactions immediately.

Foreign Currencies

Accounts can be managed independently in different currencies. The software offers a small currency converter under *File | Currency Converter* so that the conversion rates correspond to real conditions. Here, you can select the source and target currencies from a list that covers virtually all available currencies worldwide, making the list unwieldy. Either a search function or a summary of the most important currencies at the top of the list (Figure 6) would be helpful.

The software automatically queries the exchange rates for your current conversion online and enters them in the conversion dialog. To

Figure 5: Eqonomize has a reminder function to notify you when payments are due.



have these updated daily in your accounting, click *File | Update Exchange Rates*. Eqonomize now takes the current exchange rates into account when posting the currencies concerned.

Security

Handling sensitive financial data requires a way to secure it. Eqonomize therefore offers routines that allow for fast data backup, as well as exporting to other file formats.

Start the file backup from *File | Save as*. In the small file manager,

navigate to the location of the backup file with the program-specific extension (.eqz). You can make subsequent backups simply by choosing *File | Save*.

If you want to use the data in another program, the current spreadsheet can be saved as a QIF file with *File | Export As QIF File*, making it readable by other financial management programs.

With optional automatic backups, Eqonomize ensures increased data security. To set backup intervals or turn off the automatic backup function, select *Settings | Backup Frequency*. Eqonomize then overwrites old backup datasets with new ones at the selected time.

Picture Book

Thanks to numerous graphical format options, Eqonomize allows you to output your financial data in a visually appealing format or in a simple list form. You can access the simplest form of list output for further processing under *File | Export View*.

This option saves the list view shown in the dialog window to an HTML file by converting the lists into clear-cut table views, thus offering a superior overview compared with the list view on the screen, especially for large volumes of transactions. Another graphical format option can be found in the *Statistics* menu, where four different types of tables and charts are available.

Choosing *Development Over Time Report* lets you can generate a table showing the overall development of your income or expenses on a monthly basis. The table appears in a new window and can be saved as an HTML file with the *Save As* button in the upper-right corner. In the lower part of the window, you specify in a selec-

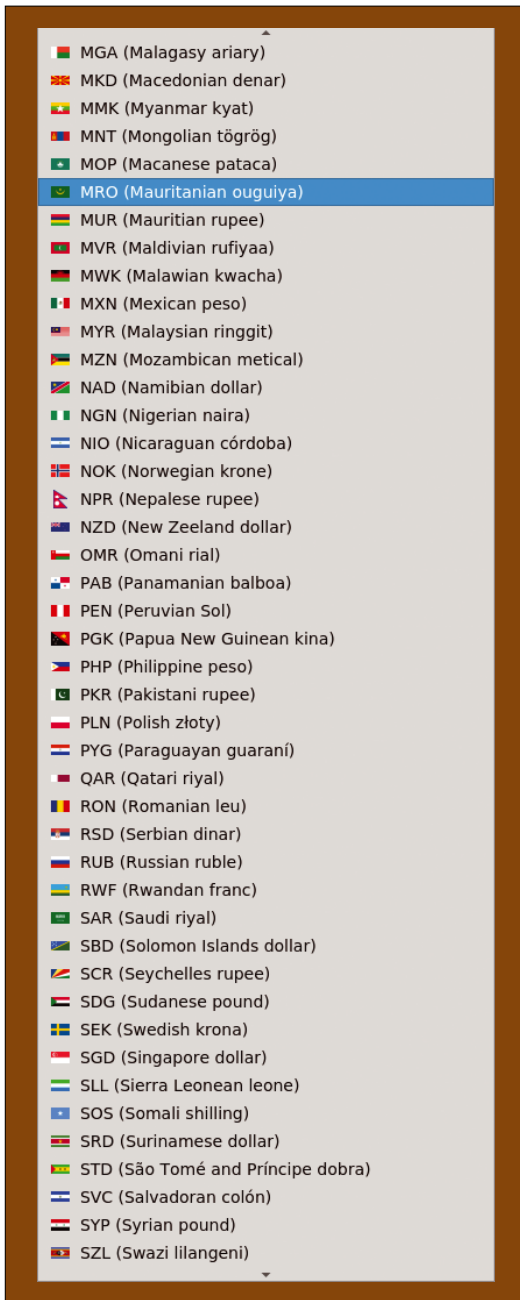


Figure 6: The software supports almost all available currencies, but navigating the currency list can be unwieldy.

tion box whether you want the table to include incomes or expenses and select one of the existing accounts.

The *Categories Comparison Report* displays a complete tabular overview of all transactions, with the underlying data filtered by category, account, and date. These tables are, of course, quite extensive (Figure 7).

The options *Development Over Time Chart* and *Categories Comparison Chart* display the same data, but as graphs. They offer the same selection criteria as the table views, but use bar or line graphs to display the values. The category comparison also offers pie charts. All graphs can be saved as image files or printed (Figure 8).

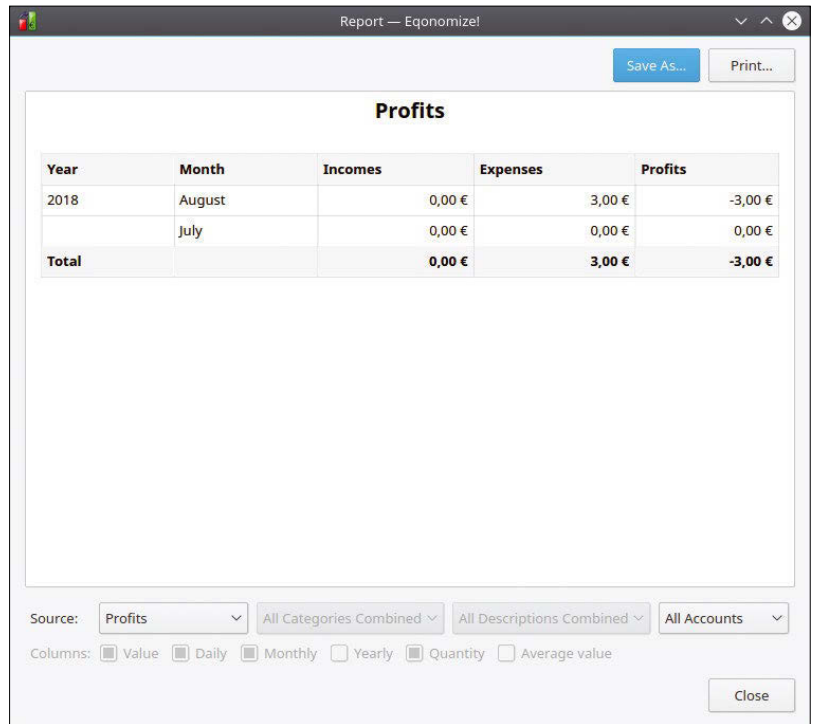


Figure 7: The table format clearly displays the data.

Conclusions

Eqonomize is an excellent accounting program for personal use. Not only does it display the data clearly onscreen, but Eqonomize also can display your data in a table or graph format for printing or as an HTML file for further processing. Finally, the program requires virtually no training thanks to its intuitive interface. ■■■

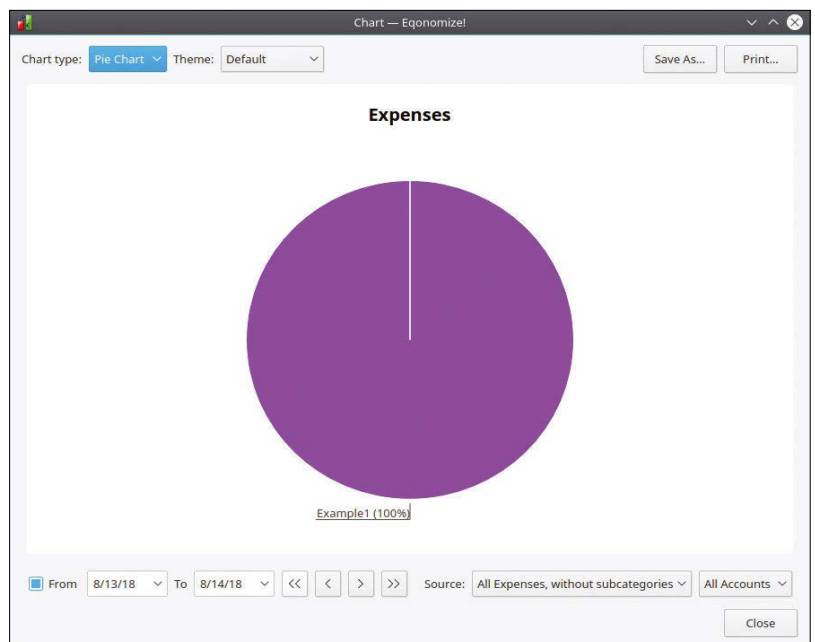
Info

[1] Eqonomize: <https://eqonomize.github.io>

[2] Eqonomize download:

<https://eqonomize.github.io/downloads.html>

Figure 8: Thanks to various graph and table formats, your financial status can be displayed attractively at the push of a button.



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**

Email client

Thunderbird 60

As much as online proprietary services would like old-school email to go away, it's not dead yet. The great thing about email is that it's truly peer-to-peer and open. It enables any of us to run our own mail domain and send and receive messages from our own servers or computers, which causes the major problem with email too – anyone includes spammers, and there are thousands of them. There are solutions to spammers (SpamAssassin and Rspamd), and email is still amazingly useful. In the end, we still

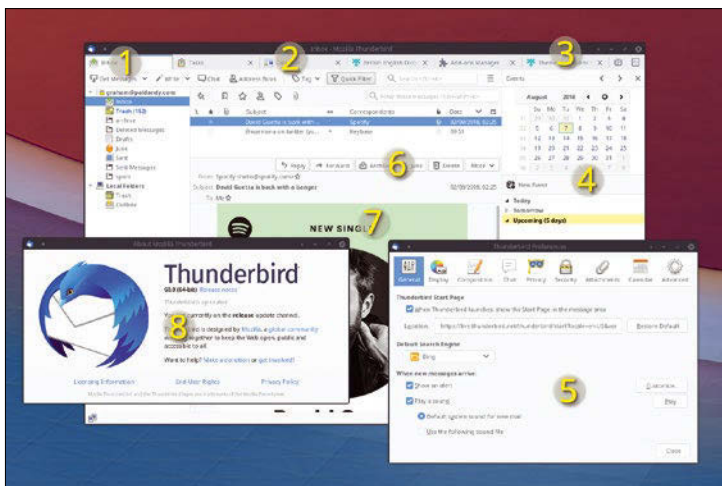
need a desktop email client. Roundcube and other online services are great, but they can't compete with the desktop integration and offline access of a proper application like Mozilla's Thunderbird.

Thunderbird used to be the go-to desktop email application, regardless of your operating system and desktop environment. Its development stalled. Fortunately, there was enough community concern for Thunderbird and its pivotal role as one of the only usable open source email clients that devel-

opment has restarted. This is the first major Thunderbird release under this new regime, and one hopes the first of many as Mozilla rewrites the codebase, drops the old Firefox technologies, and builds an email client fit for the future. This doesn't mean that this release doesn't include lots of updates – it does. After a long period of stable release stasis, version 60 really does contain many new features and fixes. For that reason, it doesn't automatically upgrade from old versions. Keeping with the times, there are now light and dark themes thanks to the use of Firefox's Photon design and excellent FIDO U2F support for two-factor authentication with various devices. There's also experimental support for the conversion between MBOX and Maildir mail storage formats, which is particularly useful for Linux users who have historically started with one and now want to switch to the other.

When composing messages, there are several improvements to the way attachments are handled, allowing you to reorder them. The attachment pane appears when you first start writing an email, along with a hidden but non-empty attachment pane showing a paperclip. You can also remove recipients by clicking on a delete button that's displayed when you move your cursor over the To/Cc/Bcc selector, and you can save a message as a template for other messages, creating them with the *New Message from Template* command. Native Linux notifications have been also reinstated. Besides these changes, there are lots of fixes that aren't obvious. The calendar now allows for copying, cutting, and deleting across a single or recurring event, and it's now much easier to see event locations in the week and day calendar views. Thunderbird is starting to feel alive again. While there are still some major features we'd like to see, such as integrated and simplified OpenPGP to strengthen Thunderbird's privacy credentials, we're just pleased the project is being worked on at all. Here's to the next release!

Project Website
<https://www.thunderbird.net/>



1. Tabbed UI: Based on the Firefox browser, different activities open in their own tabs.
2. Calendar: Aside from email, Thunderbird is an excellent event, calendar, and organization platform.
3. Themes: New for this release is the ability to switch between light and dark themes, with even more themes for the chat client.
4. Event view: Now you can copy and paste multiple events alongside CalDAV email scheduling.
5. Preferences: You can now switch between MBOX and Maildir email storage.
6. Templates: Save email as a template.
7. Remove addresses: A cross appears for easy deselection.
8. Release notes: Check to see if Thunderbird is up to date.

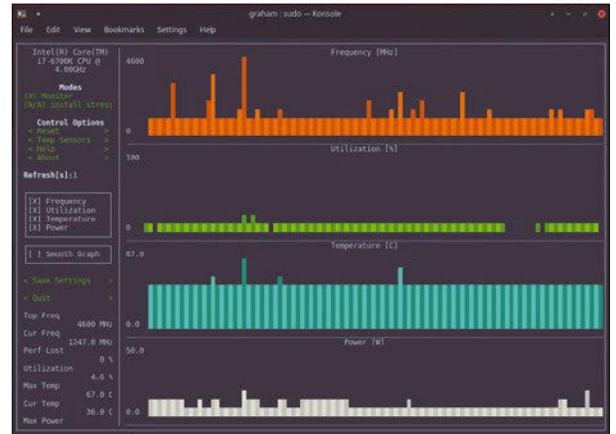
Power monitor

Stress-Terminal UI

We just can't get enough of system monitors that run from the command line. They just seem like such a natural fit, not only aesthetically, but also functionally. From a design perspective, the command line forces the developer to put usefulness up front rather than hidden behind endless tabs or options. ASCII's limitations help enforce this, because there's only so much you can do. Use of color can't be superfluous, and you need to be careful about the number of words or amount information you present, as well as making it obvious how the user interacts with your application. Plus, you're effectively working toward a fixed resolution. Of course, there are exceptions – Vim springs to mind – but most command-line

developers understand the advantages that come from environmental limitations.

Stress-Terminal UI (s-tui) is a great example of this. It monitors your system's CPU utilization, but also shows its changing frequency and temperature alongside power consumption in watts. This is perfect for monitoring the effectiveness of your system cooling, as well as the kind of power consumption you can expect from your system under load. What's even better is that **stress-ng**, a popular stress tester, can be run directly from within s-tui, so you can monitor the effects of high CPU usage directly. You navigate around the user interface using the Vim direction keys, which allow you to toggle stressed and regular operation



Thanks to being a terminal application, s-tui lets you easily monitor how much power your system is consuming even over a remote connection.

and select different temperature sensors, as well as provide a toggle for each chart. You can even output the statistics to a JSON or CSV file with a launch argument. s-tui is a tiny tool that does everything you require, and it's just as good monitoring your local system as it is checking over a remote machine – just be careful that the stress test doesn't cause a remote crash!

Project Website

<https://amanusk.github.io/s-tui/>

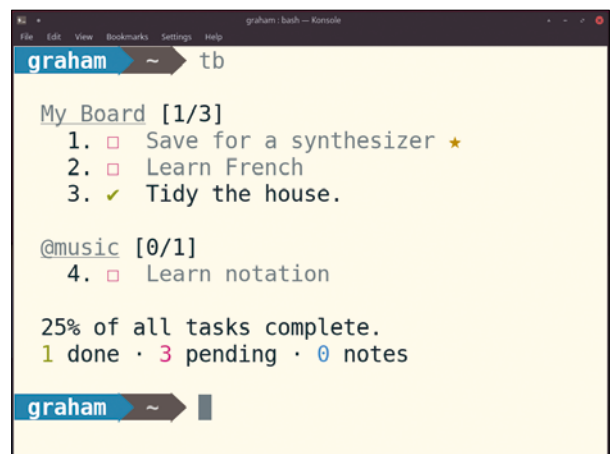
Task management

Taskbook

Soon we'll be able to give up our OpenGL-accelerated desktops entirely and return to the framebuffer from whence we came. It will solve all kinds of problems, from needing to upgrade your machine because you need to run Slack desktop, to being more productive because YouTube isn't nihilistic when watched as ASCII. One thing we won't be short of is tools to keep us productive. Vimwiki is excellent, for example, if you want a quick and easy way to create a small wiki directly from your editing environment (Vim). With a simple shortcut key, you can switch from whatever document you might be editing to your own note and linking environment using simple mark-

down language. Press Enter on a title, and it becomes a link to a new page where you can continue, and all of this can easily be exported as HTML or accessed directly.

Taskbook is a little like Vimwiki in that it uses a minimal syntax to add and manage notes and tasks. Unlike Vimwiki, it doesn't have a steep learning curve, because you don't need to learn Vim first. Instead, you run it as a single command from the terminal. To add a task, just type `tb -t Save for a synthesizer`. As your reward, you'll get a little green tick and a quick message to say the task has been added. What's particularly clever is that you can pin tasks to specific "boards," which work like categories or tags. `tb -t @music Learn`



If you work on the command line, it makes sense to bring as many tools as possible to the same environment.

notation will add a task to a "music" board, for instance. Type `tb` to lists your tasks, or add `-i` to see them on a creation-date timeline. It's easy to create huge lists of these, but `tb` is quick and simple enough to make managing and navigating between them a breeze.

Project Website

<https://github.com/klauscfhq/taskbook>

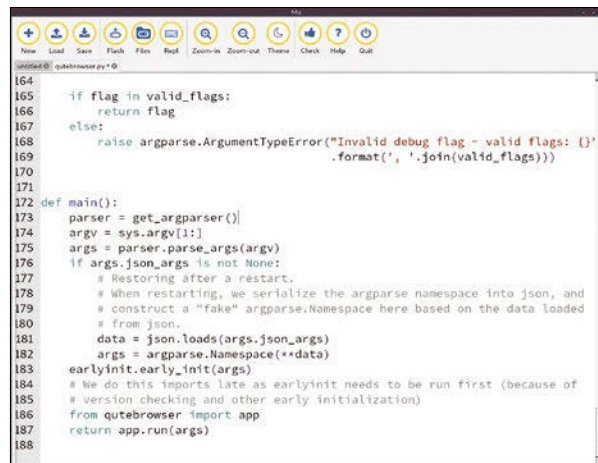
Python IDE

Mu

Considering how many people (millions?) use the Python programming language, it's surprising that no one has yet created the perfect IDE for people who want to code. If the perfect IDE were to exist, it would have to cater to one of Python's most important and unique demographics: the beginner. It's this specific demographic to whom the Mu editor aims to cater, both in terms of beginners and those teaching beginners. As a good example of the latter, the user interface is remarkably clear. The toolbar, for example, features large and easy-to-distinguish icons. Not only are these going to be easy for people unfamiliar with programming to use, see, and understand, they're also going to be easy to see when projected in

front of a room of students. This may also be why there are such prominent *Zoom in* and *Zoom out* buttons, allowing the teacher to easily scale the view according to the amount of wall space.

Behind the graphical niceties is a well thought out IDE with some unique features. You can change the editing mode, which defaults to writing code for the BBC micro:bit, but it can be switched to a mode more suitable for writing CircuitPython for Adafruit's hardware, making games with Pygame Zero, or simply writing a vanilla project for Python 3. As you change modes, Mu will understand automatically how to communicate with your hardware or provide extra tools, such as the visual debugger when using the Python 3 mode. There's



If you're a Python beginner, especially if you're working with hardware like the BBC micro:bit, the Mu editor is a great place to start.

even a very neat data plotter. If you're using a micro:bit, the *Repl* button lets you run your code interactively on the external hardware. It even runs on the Raspberry Pi, making it the perfect application for students in the lab and to run at home.

Project Website
<https://codewith.mu/>

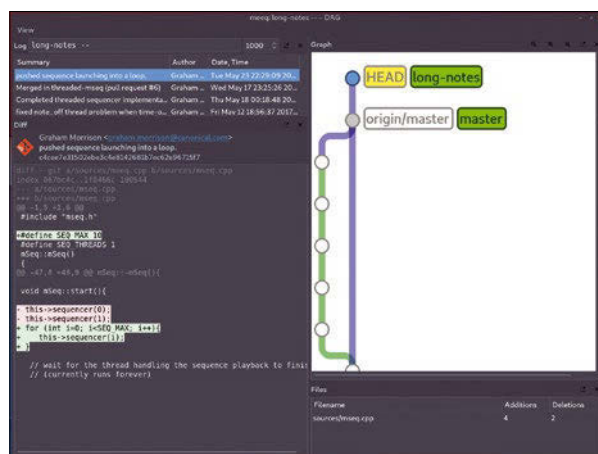
Git GUI

Git Cola

In the same way that we're all perhaps looking for an ideal Python IDE (see above), we're all waiting for a good Git GUI that can make sense of all of those weird options that keep being added. Git is brilliant and wonderful and not too difficult to understand at a high level. However, as soon as you scratch the surface, there are just so many options and ways of accomplishing the same task. A GUI should help resolve these problems by offering a canonical process for all of the most common tasks, as well as a way to visualize exactly what's happening in your own branches, as well as the branches they track on some distant online repository. Git Cola gets close, although, if you're a beginner, it won't help

you learn Git. This is because its default view assumes you already know what you're doing and most likely are accessing a project with a comprehensive set of Git history.

The main view on the left is for showing "diffs," the textual differences between one version of a file and the next. You choose files in various stages of development from the Branches pane, as well as the status pane that shows the paths for those files. A file browser lets you view all the files in the project, along with their status and their last commit message; our favorite view is the DAG visualiser. This looks a little like the visualizer in GitHub, only the timeline for a file is vertical rather than horizontal. It shows how the file



If you find working with Git on the command line a struggle, checkout Git COLA to see if it makes it easier to understand.

diverges and remerges throughout its lifecycle in the project, and consequently, how the entire project grows and changes over time. This could offer some valuable insights to more established projects, helping projects understand how to use various developers to the best of their capabilities.

Project Website
<https://git-cola.github.io>

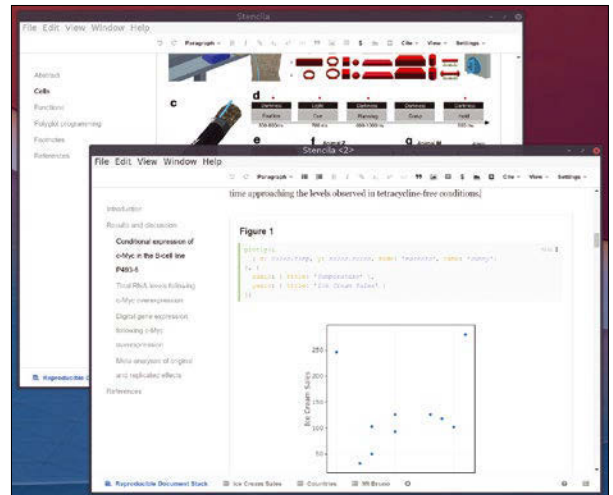
Scientific office suite

Stencila

While the wider world argues about open access to a research paper written invariably about taxpayer-funded research, there's a smaller revolution happening in software. Stencila is an office suite designed specifically for writing the results of reproducible research. Its primary user interface is just like any other office suite, where you create a document and start typing. It includes all the features you'd expect, such as excellent text support and better than average citation support. But in Stencila, these features are augmented by the "execution engine," an embedded reactive programming model that feels a lot like a spreadsheet. It enables you to enter data and process that data within your research document, but when that data changes, so too does the output

generated in the remainder of the document. You can even mix code from various programming languages used to process the data within your documents to show how results are processed. This is exactly what you'd expect with an academic paper, but the way this is now interactive and verifiable feels brilliant. It lets you or your peers look through the code for how a plot has been generated, for instance, rather than trusting the authors' word.

The programming languages most commonly used are R, Python, JavaScript, SQL, and Stencila's own Mini. You can share custom functions, data validation, and custom types. Code is often written in snippets, with data loaded via XML – the entire document is typically a series of XML files, and you can convert from Excel



Illustrations take on a whole new level when they include the interactive code that generates the output.

spreadsheets, Jupyter notebooks, Markdown, Word documents, and LaTeX. Many of these converters are "lossless," which means you can save your work back out again in the same format and send it to colleagues working with those different software tools. When you're ready to publish, Stencila supports Journal Article Tag Suite (JATS), commonly used by publishers to provide finer output control.

Project Website
<https://stenci.la>

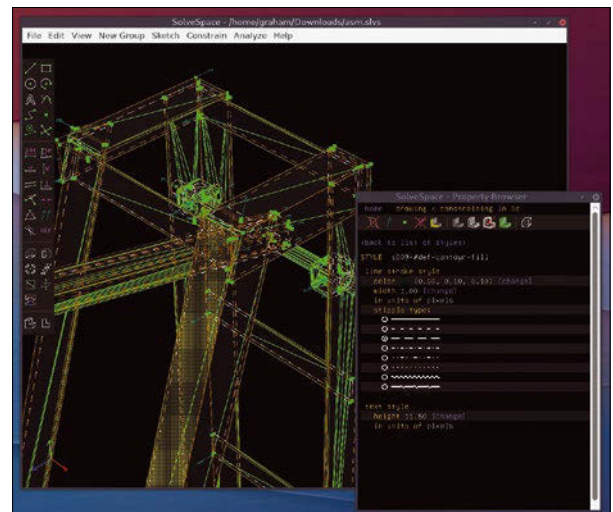
Computer-aided design

SolveSpace

The 3D printing revolution has brought many more users to the world previously dominated by engineers: computer-aided design (CAD). It's a world where software seldom moves rapidly: Many applications like LibreCAD, QCad, FreeCAD, or the proprietary VariCAD, would see intermittent releases, often over a period of years. 3D printing has upped the pace on some of the projects, and that's true of SolveSpace, which has just been awoken from its slumber to add high-DPI support. SolveSpace doesn't usually get mentioned alongside other CAD applications, but it should. SolveSpace, as its name implies, is a parametric modeler, as opposed to a direct modeler. Parametric means the relationship

between each element is mathematical, rather than hand drawn, despite mouse or controller manipulation of the various points and lines in your model during the design phase. The result is exacting and formulaic designs that are ideal for manufacturing.

There are too many important features to list, but objects can be created using lines, rectangles, datum lines, and points. You can draw curves, arcs, and scalable vector text, which can be saved, too. Solid models can then be extruded and processed with boolean operations before measuring factors such as volume or internal area. Its best feature is the property browser, which lists all the items in your scene and allows them to be selected and then manipulated as



Design your own objects and send them to a laser cutter!

groups. Because it's parametric, this manipulation is mostly visual, as you decide how grouped elements are shown, from the line types that connect them to the way normals are displayed. It's always clear and easy to see what's going on and to get a good insight into your model's construction. And it now looks great on high-DPI displays, too!

Project Website
<http://solvespace.com>

DX7 emulator

Dexed

There aren't many synthesizers that people uninitiated with the world of oscillators and filters are likely to name. But if there's one, perhaps mentioned after the oft-mispronounced Moog (pronounced to rhyme with "rogue"), it's the Yamaha DX7. This synth from the early 1980s dominated that decade and changed the sound of popular music. Before the DX7, analog was at the top of its game. But analog was expensive, sensitive to environmental conditions, and difficult to integrate with a studio. The DX7, on the other hand, was digital. It had MIDI so that notes could be played from early computers, obviating any need to play, and it did away with tactile controllers, replacing them with a minimal set of buttons and a tiny LCD. But most importantly, it sounded like nothing else that came before it. It was a digital synth in a world of analog, and it wasn't even trying to sound like an analog synth. Instead, it implemented something Yamaha called "frequency modulation."

Frequency modulation is a process where the frequency of one waveform, such as a square or sawtooth wave, is modulated (changed) by another similarly pitched waveform. You can hear the effect in an analog synth if you modulate the frequency of one oscillator with another – it creates metallic percussive sounds, similar to a bell. But analog components are too variable to control with any degree of certainty. That's where the DX7 succeeded. Not only could it create those sounds using commodity digital components, it could generate sounds using code that would always generate the same sounds on the hardware, regardless of the humidity or the amount of smoke in the studio. And it could even do this while playing more than one note at a time and, later, more than one instrument at a time, and for a price lower than the average polyphonic analog.

Dexed is an exacting and perfectionist software recreation of the DX7. It aims to capture all the peculiar and specific nuances of the original, brought



The FM7 was the sound of the 80s, and you can choose from all the same algorithms in Dexed.

about because of the relative newness of digital technology of the time. It is these nuances that give the DX7 such character, despite nearly every PC sound card including an FM sound generator since the 90s and CPUs being capable of the processing involved for almost as long. It loads and saves sounds from the original synth, loads cartridges, and can even control the original parameters with the various sliders and buttons on the software interface – a huge advantage over the limited input of the original. There's also the option to downgrade the modern 24-bit sound generator with designs based on the original OPL 8-bit chips for a more authentically noisy output. However, this is also more than nostalgia. The synth sounds wonderful, and it's completely open source. The electric pianos and pads are still perfect in today's music, especially with a little processing, and FM synthesis remains one of the most complex to get your head around if you want to get into sound design. There still is no other set of synthesizers that sound the same, and you can get it on your Linux desktop without having to worry about leaking batteries or beer damage to the keyboard.



Not only does Dexed sound just like a Yamaha DX7, it can even control a real DX7 and load its patches.

Project Website
<https://asb2m10.github.io/dexed/>

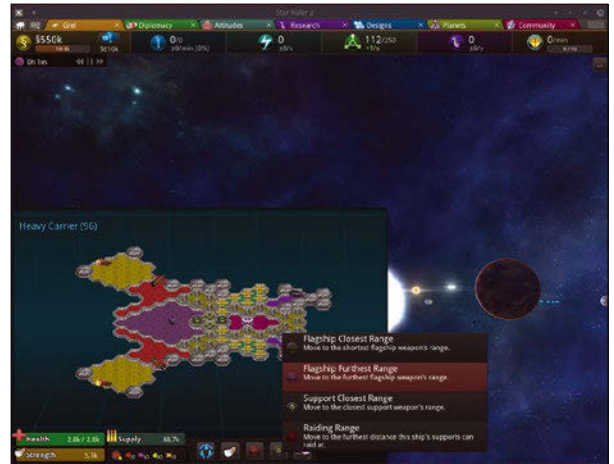
Real-time strategy

Star Ruler 2

While it's sad when a company gives up on gaming, if they choose to do the right thing with the code, it benefits a much wider audience. Star Ruler 2 is a great example. This is a popular real-time strategy (RTS) game from 2015, and it's still available on Steam, but the studio behind it, Blind Mind Studios, has been inactive for a few years. Rather than drop off the radar completely and alienate its players, the studio decided to open source both its game and the expansion pack. The code drop didn't include binaries, but you can already install the game with just a single command, `sudo snap install starruler2`, if your system supports snaps. This is great news

for anyone who wants to study the game's engine or modify it to make their own game, but it also means we get a major multiplayer RTS title on Linux for free, and that's never a bad thing.

Set in space, the game itself is known as a "4x" RTS game, meaning your role as the ruler of the galaxy is to explore, expand, exploit, and exterminate. There are seven different races with different attributes, or you can create your own. With that done, you're dumped into space where you need to manage your finances and local resources to build an empire. You map resources between systems by dragging lines between them, creating networks for imports and exports. You use your fleet



You can drag lines between systems to create trade routes and even design your own warships for total galactic domination.

of ships to explore and colonize just as you do with other RTS games. As you play, you acquire points of influence that can be used in diplomacy, and you can even design your own ships. The game is a lot of fun, and it's polished, just as you'd expect with a commercial game. And hopefully, this is the beginning of a new era for Star Ruler 2.

Project Website

<http://starruler2.com>

Games development

Pyxel

This isn't a game, but it could be your gateway to getting into game programming. Pyxel is a game-specific module/API and extended environment for Python. What makes it different from similar APIs, or simply using graphics fundamentals yourself, is that Pyxel is purposefully limited. It limits the number of colors that can be displayed onscreen to 16, it can only play four sounds at a time, and it only manipulates three 256x256 image banks. These limitations aren't to help Pyxel games run on old hardware; they're designed to limit your options in much the same way that old games consoles' limitations, such as the SNES, forced the games programmer to think creatively. And it's this forced cre-

ativity that led to many of the greatest games of the 80s, many of which are still being played or updated today.

The other advantage to this limitation is that it's easy to get started. The Pyxel module just needs to be imported into your own code, and just a few elements are needed to make your game run on the platform. It's then a case of using the functions provided by Pyxel to write your game. For instance, `pal()` switches color palettes, `circ()` draws a circle, `tone()` plays one of four tones (triangle, square, pulse, and noise, just like those old machines). There's only about 40 of these functions to learn, and they're all very easy for even a nonprogrammer to understand. Pyxel then handles all the



Learn to code and have fun creating the kind of games popular in the 1980s with Pyxel.

rest, letting you play or analyze your game and eventually share it. There are several good examples included with the installation, and it seems a brilliant way to get started – a little like the 4k demo scene only easier to start, and playing around with it is a lot more fun.

Project Website

<https://github.com/kitao/pyxel>

One After the Other

If you expect more from a file manager than the ability to move files from A to B, Polo might be for you. **BY CHRISTOPH LANGNER**

When it comes to file managers, everyone has their own opinion about what is best. Some want a file manager with tabs; others require a divided view with two or more windows, like the Norton Commander. For some, basic functions like copy, move, and delete are sufficient, but others want a universal tool with viewers for different formats and an integrated editor.

If you want to adapt a file manager to match your working methods and expect a solution that makes many everyday tasks easier, then Polo File Manager [1] might be right for you.

Polo File Manager, which is a little over one year old, brings a fresh perspective to the file manager landscape: Tabs, multiple views, an integrated terminal, and simple editing functions for images or PDF documents round out the package. Even in terms of financing, the project breaks new ground.

Installation

Because of its youth, Polo is not yet available in the major distributions' package sources. You need to install the software manually. For Ubuntu and Ubuntu-based distributions (e.g., Linux Mint or elementary OS), the developers offer a package source (Listing 1).

Under Arch, you build the application using PKGBUILD from the Arch User Repository (AUR) – either *polo* or *polo-bin*. The first entry builds the application from the source code [2]; the second

uses the precompiled DEB files as a basis. For users of other distributions, the developer provides an installation script [3].

When first launched, Polo opens a wizard that configures the file manager's layout with just one click. You can choose from one of three layout options: *Classic Icons*, which mimics Gnome Files (formerly Nautilus); *Commander Icons*, which turns Polo into a Norton Commander clone; or *Extreme*, which turns Polo into a four panel monster (Figure 1). Try out the different options; you can always restart the wizard with *Tools | Style Wizard* if necessary.

Listing 1: Ubuntu Package Source

```
$ sudo apt-add-repository -y ppa:teejee2008/ppa
$ sudo apt-get update
$ sudo apt-get install polo-file-manager
```

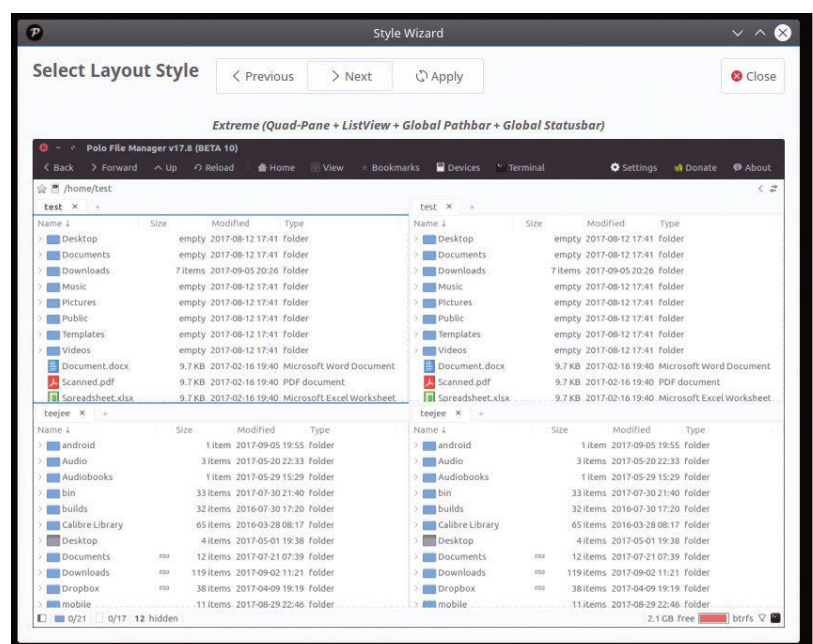


Figure 1: The Style Wizard dialog lets you select file manager layouts with a mouse click. The *Extreme* option with four panels is shown here.

Configuration

Before getting started with the file manager, you should briefly devote some time to the Settings dialog, which is accessed by the *Settings* icon at top right. If you prefer to work with a current Gnome desktop (including the Ubuntu 17.04 interface, minus Unity), activate the header bar by clicking *UI* and checking the *Enable (R)* box under Headerbar (see Figure 2).

After a reboot, Polo uses the window bar to display the menus and the pathbar. For devices with small displays, this modification makes room for the application's content.

Under the *General* tab, enable *Bourne again shell (bash)* in the Terminal section. By default, Polo tries to load the Fish shell, but very few users actually have it installed on their computers.

Other configuration settings, such as icon size depending on the view, the Gtk theme, and the Kernel-based Virtual Machine (KVM) settings, allow you to control many details.

Polo File Manager

The Polo application window is usually divided into three areas. The sidebar on the left shows the most important directories, bookmarks, and mounted drives. The middle section contains the file manager, which can be divided into two or four panels. If necessary, you can show or hide a vertical buttonbar between panels if you are in the dual-pane Commander

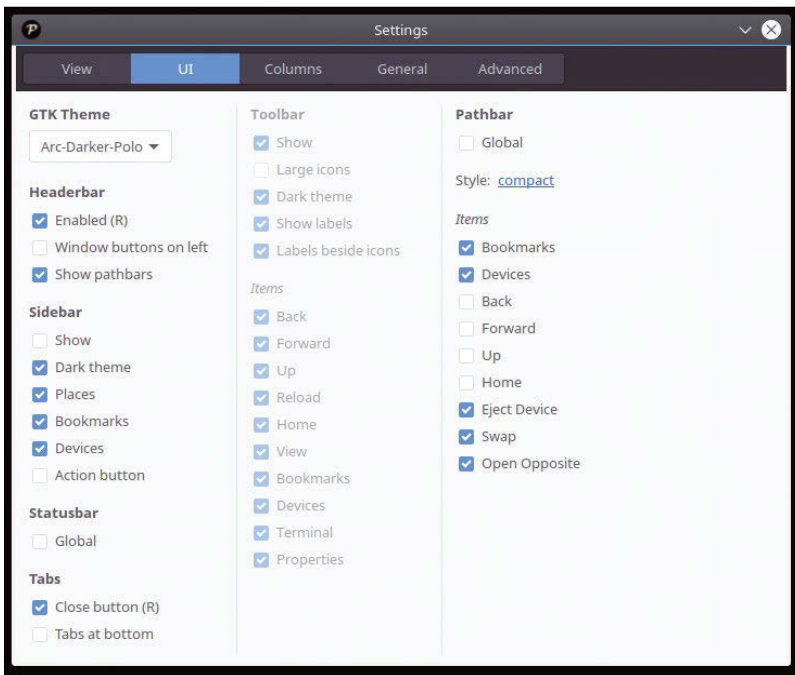


Figure 2: Use the Settings dialog to configure Polo down to the smallest detail. The Headerbar options move the menus and buttons, typical for modern Gnome applications, into the window bar.

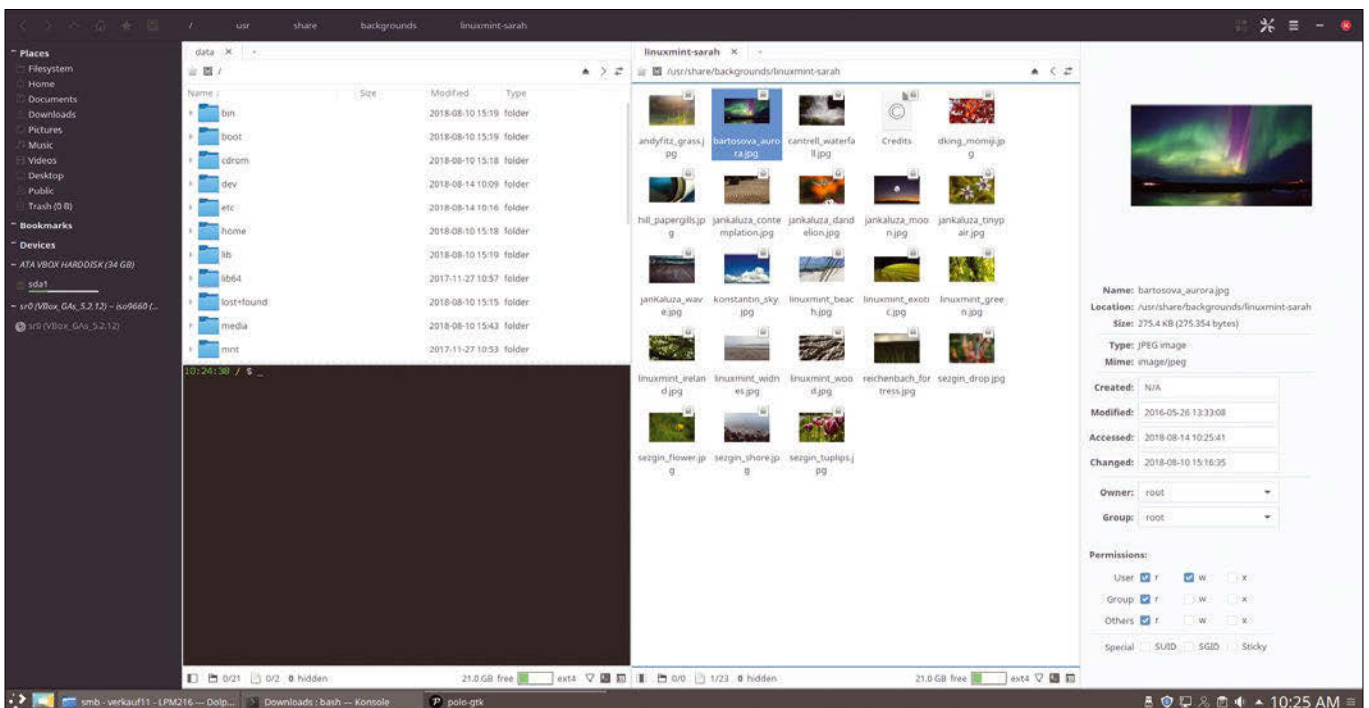


Figure 3: Polo in practice: Two panels allow convenient handling in the filesystem. The terminal automatically follows the file manager to the active directory. The right sidebar provides information about the currently selected item.

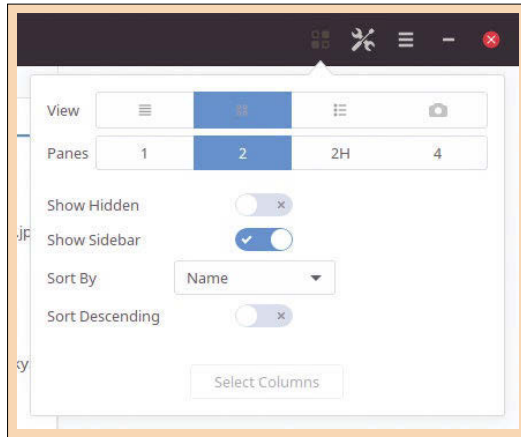


Figure 4: You can switch between the different views with a mouse click. In the same menu, activate or deactivate additional fields or specify the sorting mode.

view. The sidebar on the right displays information about the currently selected file or directory along with a preview (Figure 3).

In the window bar (after activating the Headerbar option in Settings), you will find the usual system navigation buttons and the current path's buttons. To the right of these buttons, click on the *View* button to change the view (*List*, *Icons*, *Tiles*, or *Media*) or the pane division in the center section, or adjust the default setting for sorting and displaying hidden files (Figure 4). The *Gear* button next to the *View* button opens the Settings dialog, and the hamburger button shows the menu.

The status bar shows and hides the sidebars and the center buttonbar, which accesses file operations between the panels. The status bar also informs you about the number of selected files and directories in the current folder, as well as the size, permissions, and type of partition. The three buttons on the status bar's far right let you activate a filter, show or hide the terminal, and show or hide the right sidebar.

Donations Pay Off

In addition to the usual file manager functions such as copying, deleting, or opening files, Polo can do tasks that

would otherwise require additional applications. These include writing ISO files to USB sticks or memory cards, simple image editing functions (e.g., rotating images, converting to other formats, or scaling to a specific resolution), and PDF editing functions (Figure 5). For example, you can split or merge PDF documents, reduce the file size of the document (practical for emailing or for other applications with file size limits), or password-protect documents.

However, these functions are not free of charge. The developer, Tony George [4], tries to finance his work with this extended range of functions. George, who also works on other Gtk3-based applications, asks for a minimum donation of \$10 via PayPal [5]. Alternatively, you can support George regularly via Patreon [6]. After making a donation, you will receive an email with the *polo-donation-plugins* package in the form of DEB files (suitable for installation under Ubuntu) or as a script for installation.

Polo does not reinvent the wheel with these extended functions, but uses external programs such as 7z for unpacking archives, pngcrush for optimizing images in PNG format, or Pdftk for manipulating PDF documents. While Polo assumes that these programs are installed as dependencies on your system, this does not happen automatically when installing the plugins.

To find out which applications are missing, *Tools | External tools* provides a wizard to check for these external helpers (Figure 6). If the wizard

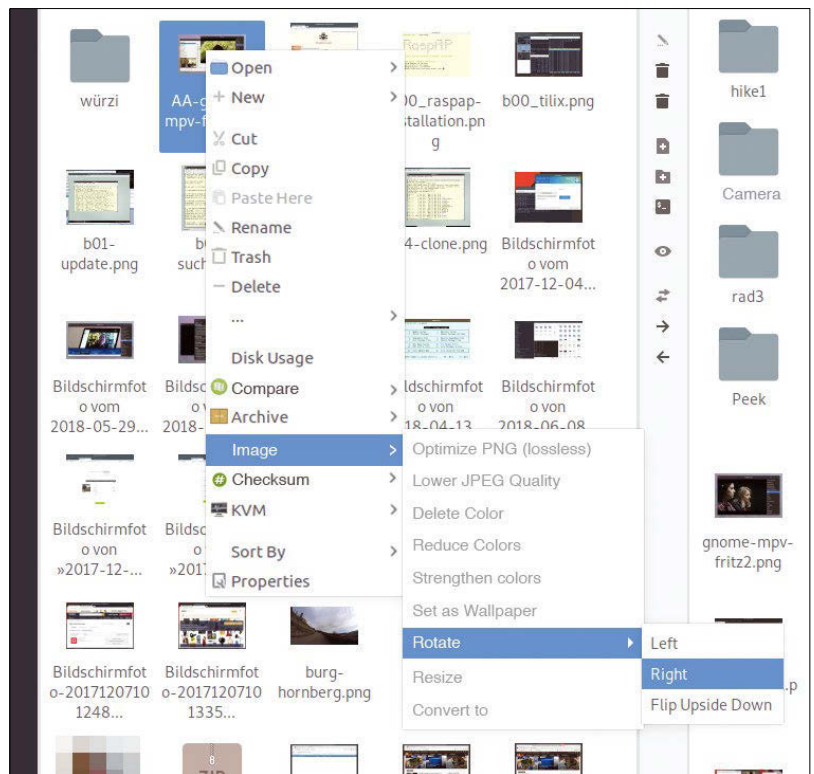


Figure 5: For advanced functions, such as editing images or PDF documents, you need to make a donation to the developer.

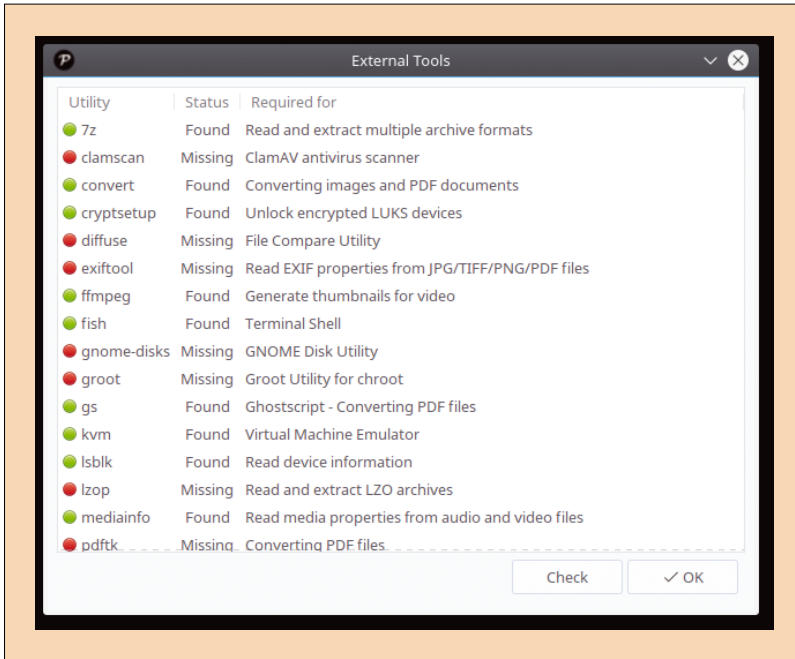


Figure 6: An integrated wizard checks whether the dependencies required for the various functions are installed. Usually, you can find the missing programs in the system's package management.

reports a utility *Missing*, use the package manager to reintegrate the program into the system.

Cloud and KVM

Many users store data not only on their home PC, but also on services such as Dropbox, Google Drive, or Yandex Disk (to name just a few). Despite concerns about privacy and data protection, such online storage is practical in many cases (e.g., to send large volumes of data to friends or relatives).

Whereas Dropbox has had a native Linux client to synchronize data in the background for years,

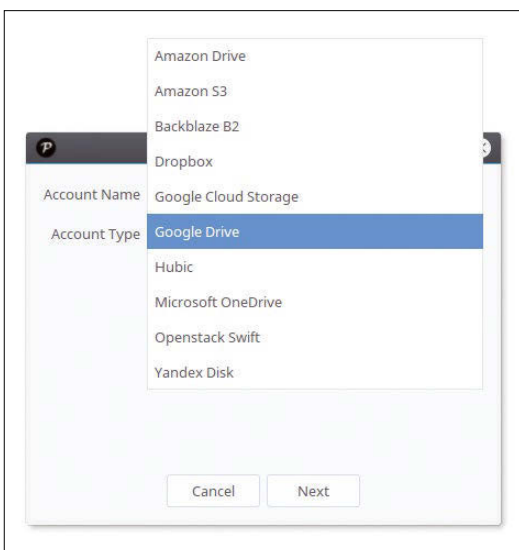


Figure 7: Polo integrates cloud storage such as Dropbox or Google Drive. It uses the Rclone command-line tool under the hood.

Rclone in the package sources.

You set up access to Dropbox and others directly from within Polo (Figure 7). During testing, this worked reliably with Dropbox, but I had to configure access to Google Drive manually in the terminal (see the "Setting up Rclone for Google Drive" box).

Setting up Rclone for Google Drive

In the test, access to Google Drive with Rclone could not be set up in Polo. Instead, this had to be done manually in the terminal: If Rclone is installed, start the interactive configuration with the command `rclone config`; then, enter *Google Drive* or any other designation as the name. In the next step, enter *drive* to select Google's cloud storage as the back end.

After that, much of the information is optional: *client_id* may remain empty, as well as the *client_secret* option (hit Return). For full read/write access, select option 1 (*Full access all files*) in the next step. The entries for *root_folder_id* and *service_account_file* can also remain empty.

Now Rclone should open a browser with a Google page that allows Rclone to access the Google data. Confirm the request and return to the Rclone configuration. Finally, deny the configuration as a *team drive*, save the setting with *Y*, and exit the wizard with *Q*.

Google doesn't have one. In response to requests for Linux support, Google said "we're working on Linux support. Hang tight!" However, it's been over six years since Google made that statement [7].

With tools like Rclone [8], you can easily integrate cloud storage into the system. Polo directly integrates the configuration of the text-based application in the Cloud menu. Additionally, the software offers a wizard (under *Tools | Install Rclone (Cloud Storage Support)*) for installing the tool if your distribution does not yet contain

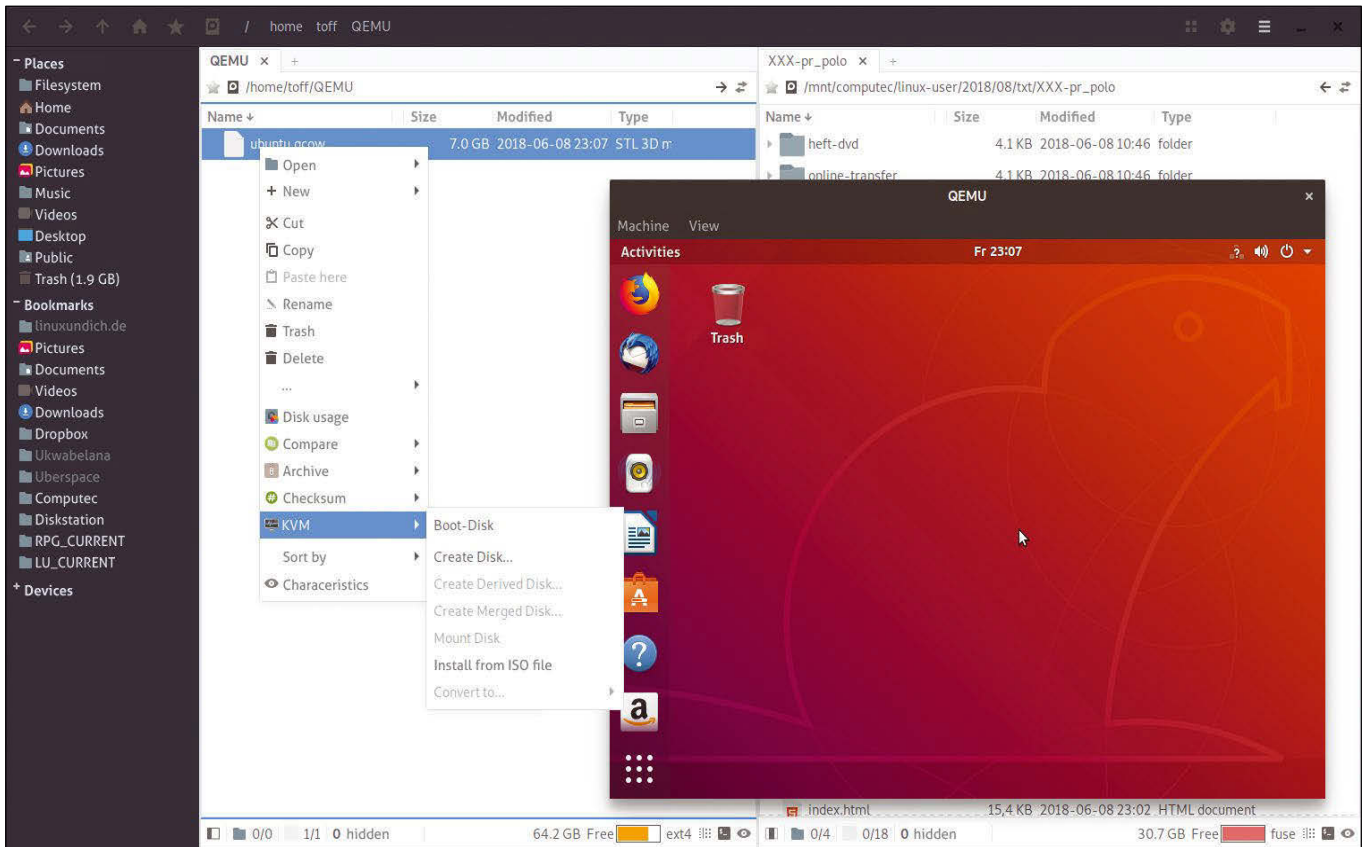


Figure 8: KVMs can be controlled directly from the file manager. In the test, however, the function proved to be faulty, especially under Arch Linux.

If you develop software or experiment with the system, you probably want to use virtual machines (VMs). VM software, such as VirtualBox or VMware, is not required; the Linux kernel comes with everything needed to boot a KVM virtualized operating system virtualized [9].

Polo supports booting VMs and loading ISO images into the VM. Simply click on the corresponding file and open the entries under KVM in the context menu (Figure 8). In a test under Arch Linux, Polo used the `kvm` command, which no longer exists under Arch. A symlink created with `ln -s /usr/bin/qemu-system-x86_64 ~/bin/kvm` works around this problem until the bug is fixed in Polo.

Conclusions

Despite its newness, Polo impresses with functions that cannot be found elsewhere. Anyone who works with multiple documents will appreciate the PDF functions. The ability to rotate or scale images easily relieves tedious work in everyday life. Even managing archives is easy with Polo: ZIP files and other formats are displayed like directories – working with them in the archive is completely transparent.

Some shortcomings and weaknesses are still on the developer's to-do list: For example, dragging and dropping files and folders to a new folder in the current window or to a new destination in another panel [10] is still missing. Additionally, in

the tested version, 18.3.1, Polo does not remember a session's field and sidebar layout when you close and restart the program. These small details hardly cloud the positive picture, however. Polo is still worth a try. ■■■

Info

- [1] Polo File Manager: <https://teejee2008.github.io/polo>
- [2] Source code: <https://github.com/teejee2008/polo>
- [3] Installation script: <https://github.com/teejee2008/polo/wiki/Installation>
- [4] Tony George on GitHub: <https://github.com/teejee2008>
- [5] Advanced functions: <https://github.com/teejee2008/polo/wiki/Donation-Features>
- [6] Tony George on Patreon: <https://www.patreon.com/teejeetech/overview>
- [7] Linux support for Google Drive: <https://abevoelker.github.io/how-long-since-google-said-a-google-drive-linux-client-is-coming>
- [8] Rclone: <https://rclone.org>
- [9] KVM: <https://www.linux-kvm.org>
- [10] Drag-and drop-support: <https://github.com/teejee2008/polo/issues/27>

Sensor & Sensorability

Frameworks like Cordova make creating simple mobile apps quite easy. Making apps that use your phone's sensor is slightly trickier, but, thanks to a new universal standard, things are not as hard as you may think. **BY PAUL BROWN**

In the May 2018 issue of Linux Magazine, you learned how to read and transfer GPS data from a phone to a computer [1]. That made me wonder if you could also play with data collected from other sensors on your phone. The answer I knew immediately would be “yes,” but could it be done in simple enough way that would allow me to explain it in a shortish tutorial written in plain English?

My first instinct was to use an existing app. I found several apps that looked promising but ended up having to discard them all, because they were proprietary and tended to leak data, or they were unstable and crashed, or the code was very old and no longer maintained.

There was one that caught my eye: SSJ Creator [2], which is open source, and has an interesting node-based interface (Figure 1) that lets you configure which sensor gets read and where you pipe its output (Figure 2).

However, it is a bit of overkill: The amount of options and settings make it confusing, and it also crashed consistently when I tried to pipe output to a socket that would allow me to read it from my laptop.

Running out of options, I had to ditch the idea of using an off-the-shelf utility altogether. I have always believed that one way of better understanding something is to build it yourself. I mean, how hard could it be? Besides, in last month's issue, we already talked about the Docker-Cordova combo [3] and how it helps create apps for Android easily.

So down the rabbit hole of integrating sensor data into a Cordova-based app I went.

Sensors

Apart from the geolocation sensor you saw in the article about GPS [1], most devices

come with several other sensors. In mobile phones, accelerometers are used to detect the device's orientation. Manufacturers use them to determine when to put the screen in landscape or portrait mode, for example.

Older phones (and by older, I mean pre-2011 handsets) will not have gyroscopes, but most modern devices will. Gyroscopes do something similar to what an accelerometer does, but there is an important difference: Gyroscopes also detect rotation. Sit in a swivel chair and hold your phone stationary in front of your face. Swivel around on your chair and the accelerometer will not detect anything, but the gyroscope will. Gyroscopes are also useful for gestures: If you have

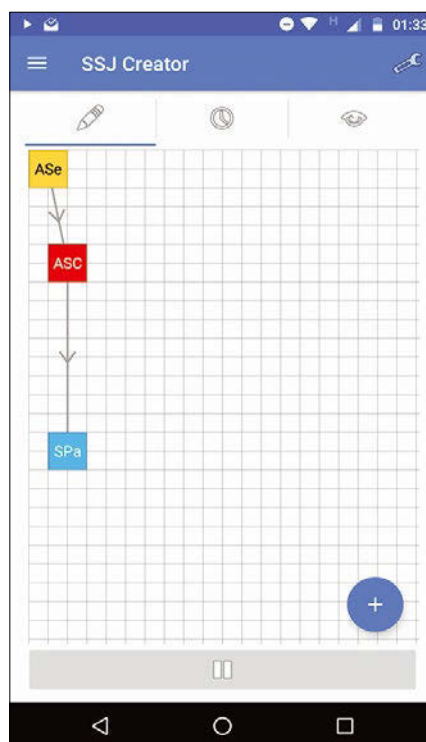


Figure 1: SSJ Creator uses a node-based graphical system to read data from your phone's sensors...

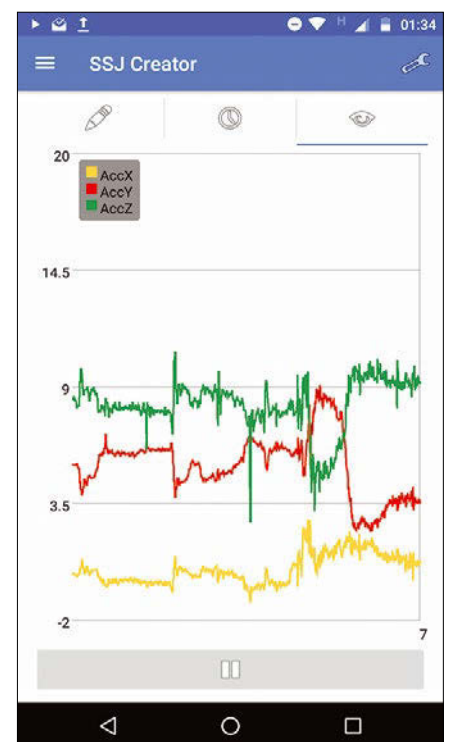


Figure 2: ... and output it to a socket, log file, or graph.

one of those phones that you flip quickly to activate the camera, that's the operating system checking the gyroscope.

Another difference is that gyroscopes are more jittery than accelerometers. While an accelerometer will not emit a lot of different data if your phone is at rest, the gyroscope will. Ultimately, many applications usually use data sent by both

sensors to detect how your phone is positioned and where it is going.

The linear acceleration sensor is very simple by comparison: It calculates the phone's acceleration in any given direction. If you are holding your phone upright in front of your face and you drop it, for example, it will register 9.8 m/s^2 on the Y axis. For the record, the X axis is, again if you are holding your phone upright, from side to side; the Z axis is the horizontal line moving away from you (Figure 3).

These are the most common sensors on mobile devices, and the three we are going to talk about in this article, although there are many more. There are sensors to detect ambient light intensity, magnetic north (i.e., a compass), magnetic fields, proximity, humidity, and so on. Smartwatches may also come with heartbeat and blood pressure sensors.

Not all devices will come with all sensors, but you are more or less guaranteed that even the cheapest phone will have the three mentioned above.

Visiting Cordova

As saw in last month's issue, you can use a Docker image to make everything simpler – especially when it comes to writing Android applications, so install Docker:

```
sudo apt install docker
```

Start the docker daemon

```
sudo systemctl start docker
sudo systemctl enable docker
```

and grab the Cordova/Android Docker image

```
docker pull beevolop/cordova
```

and you are ready to go.

To start creating a Cordova app, first you must create a skeleton application. `cd` into the directory you want your application to live in and run:

```
docker run --rm -i -v /$PWD:/workspace \
-w /workspace \
--privileged beevolop/cordova \
cordova create sensors \
com.LPM.sensors Sensors
```

This will create a `sensors/` directory containing the basic files you need to create your app.

`cd` into `sensors` and run:

Listing 1: index.html (partial)

```
01 <body>
02 <div class="container" style="padding: 10px">
03 <h2>Choose a sensor</h2>
04 <select id="sensor" style="width:100%">
05 <option value="Accelerometer">Accelerometer</option>
06 <option value="Gyroscope">Gyroscope</option>
07 <option value="LinearAccelerationSensor">LinearAccelerationSensor</option>
08 </select>
09 <br />
10 <button id="start">Start</button>
11 <br />
12 <br />
13 <h2>Sensor data</h2>
14 <textarea id="data" rows="10" readonly style="width: 100%">
15   Sensor data shows up here.
16 </textarea>
17 </div>
18 <script type="text/javascript" src="js/index.js"></script>
19 </body>
```

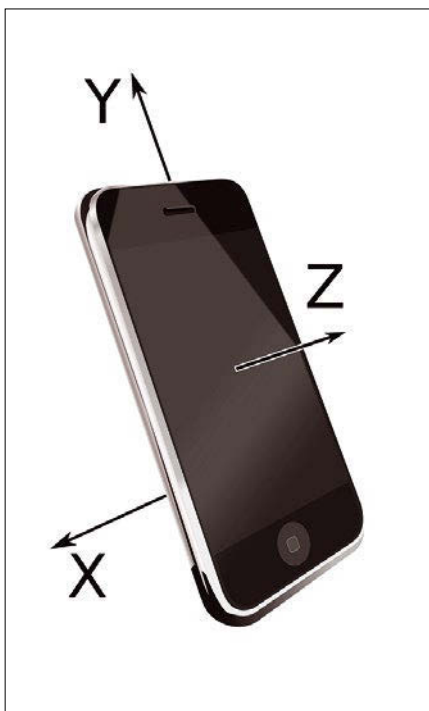


Figure 3: The X axis runs from left to right on your phone, the Y axis runs from top to bottom, and the Z axis runs from back to front.

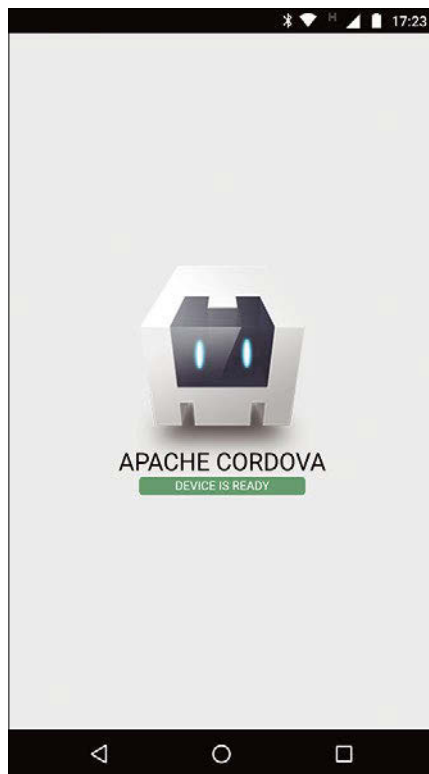


Figure 4: The stock Cordova application.

```
docker run --rm -i -v /$PWD:/workspace ↗
-w /workspace ↗
--privileged beevelop/cordova ↗
cordova platforms add android browser
```

to pull all the tools you will need to create an Android application. It is also a good idea to download what you need for a browser application, as it is easier to debug your HTML, CSS, and JavaScript in a browser than directly on your Android device.

You Got the Look

Cordova applications are a combination of HTML5, CSS, and JavaScript, so the front end (what your users are going to see) is going to be a web page with its CSS. Having created a skeleton app, you already have a demo application that, when loaded on an Android device, looks like Figure 4.

The web part of the application is the `index.html` file that lives in the `www/` directory.

You are going to exchange the Apache Cordova logo for a drop-down selection field (from which you will be able to choose the sensor you want to read), a button (that will start the reading from the sensor), and a text area field (that will show the data read from the sensor). It will look like Figure 5.

Open `index.html` in your favorite text editor. You don't have to touch anything in the `<head> ... </head>` section, but rip out everything between the `<body>` and `</body>` tags and substitute it for what you can see in Listing 1.

This is pretty basic HTML. From lines 1 to 8, you have the drop-down field with the sensor names you can choose (Figure 6); line 10 is the `Start` button and line 14 is the text area. Finally, on line 18, you load in a JavaScript file that is going to give you the code that makes everything work.

But before that, you may want to tweak the CSS. The CSS file you are looking for is in `www/css/`; it is called `index.css`. You can change the `body` section so it looks like what you see in Listing 2. Your app will look, if not beautiful, a bit better than if it used the default CSS.

Reading Sensors

The next step is actually making the app do something; this is when things got hairy. Cordova relies heavily on plugins to extend functionality. There is a nicely stocked repository online [4], and searching for "sensors" coughed up some interesting results.

Besides, you can install plugins quite easily with:

Listing 2: index.css (partial)

```
[...]
body {
  -webkit-touch-callout: none;
  -webkit-text-size-adjust: none;
  -webkit-user-select: none;
  background-color:#F8F8F8;
  background-attachment:fixed;
  font-family:'HelveticaNeue-Light', 'HelveticaNeue', Helvetica, Arial, sans-serif;
  font-size:12px;
  height:100%;
  margin:0px;
  padding:0px;
  width:100%;
}
[...]
```

```
docker run --rm -i -v /$PWD:/workspace ↗
-w /workspace --privileged beevelop/cordova ↗
cordova plugin add <plugin location>
```

But I found that some of the sensor plugins were surprisingly old and unmaintained. Others had emptied out repositories, or the plugin list links that led to 404 errors. And then there were those loudly labeled as obsolete.

It was very confusing. It made no sense that really no one was interested in maintaining plugins that accessed a mobile phone's sensors.

Unless there was no point, of course.

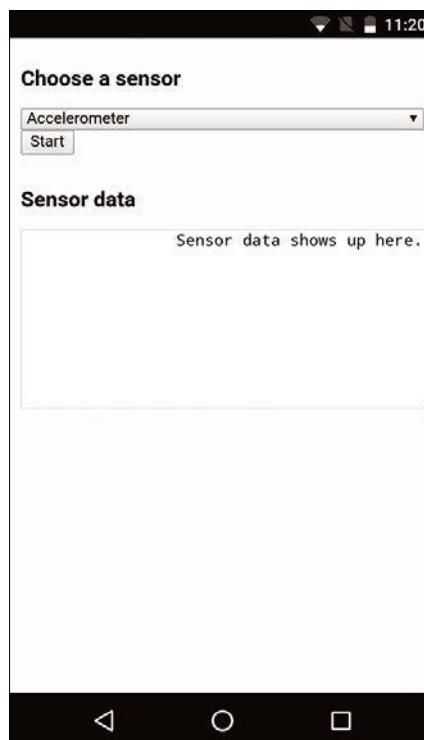


Figure 5: Your app's front end: a drop-down selector field, a button, and a text area.

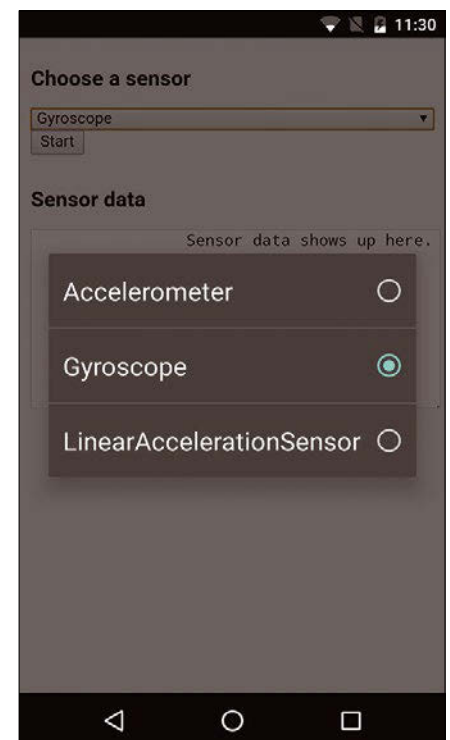


Figure 6: A drop-down selection field as it looks on your mobile phone.

The clue came from one plugin that said the work had been obsolete ever since the World Wide Web Consortium (W3C, the organization that establishes what goes into HTML, CSS, and JavaScript) had decided to take on the standardization of how to interact with mobile-device sensors [5]. That's right: The reason there is no need for plugins is because reading sensors is now as part of the web as the `<blink>` and `<marquee>` tags.

This means two things:

1. You can now have web pages like, (Figure 7) [6] that integrate sensor data from the visitor's device into the content
2. You can integrate sensor data into your Cordova apps.

The good news is that initializing, starting, and reading from a sensor using JavaScript is mercifully simple. First, you create a sensor object by initializing the sensor you want to read from:

```
var <sensor_object> = new <Sensor_name>();
```

For example, to initialize the accelerometer, you would do:

```
var my_sensor = new Accelerometer();
```

The W3C spec lists 10 sensors you can pick from: Accelerometer, AmbientLightSensor, Magnetometer, Gyroscope, OrientationSensor, LinearAccelerationSensor, AbsoluteOrientationSensor, RelativeOrien-

tationSensor, GeolocationSensor, and ProximitySensor. Note that not all sensors will work on all devices, and there are some that won't work on any device, since there are no web engines that have implemented them yet.

You can include as a parameter in the initialization the frequency with which you will be polling the sensor. To poll the accelerometer 60 times a second, you can do:

```
var my_sensor = new Accelerometer({frequency: 60});
```

Next, you have to define a callback function for the sensor, that is, the function that will run every time there is new data from the sensor:

```
my_sensor.addEventListener( 'reading', <function>;
```

`addEventListener()` is a method that is a standard part of JavaScript. It listens for the events you specify. You could add a 'click' event to a listener for a button, or a 'change' event to a text box for when someone types something new in it. For a sensor, you use the special 'reading' event.

You can call the callback function by name, like in:

```
sensor.addEventListener( 'reading', listener);
[...];
function listener( event ) {
[...];
```

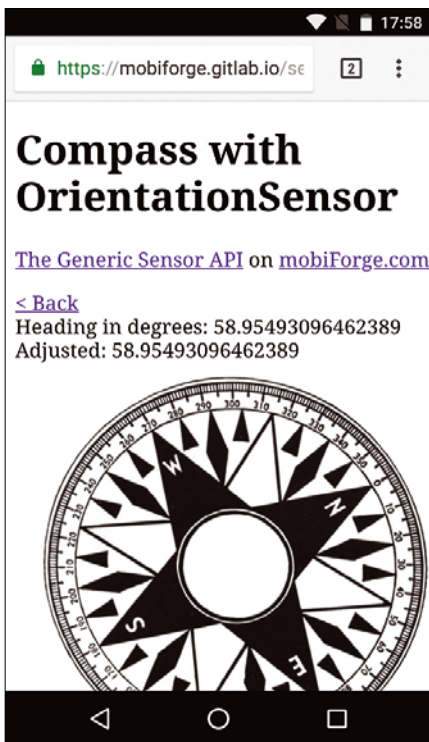


Figure 7: A web page with an integrated working compass. Warning: This will not work on devices without magnetometers.

Listing 3: index.js

```
01 var sensorSelect = document.getElementById("sensor");
02 var startButton = document.getElementById("start");
03 var dataText = document.getElementById("data");
04
05 startButton.onclick = function() {
06   if (startButton.innerHTML == "Start") {
07     startButton.innerHTML = "Stop";
08     try {
09       var sensor = new window[sensorSelect.value]();
10       dataText.innerHTML = sensorSelect.value;
11
12       sensor.addEventListener('reading', function(event) {
13         dataText.innerHTML= 'x: ' + event.target.x + ' y: ' + event.target.y + '
14         z: ' + event.target.z;
15       });
16       sensor.start();
17     } catch(error) {
18       dataText.innerHTML = 'Error creating sensor';
19     }
20   } else {
21     startButton.innerHTML = "Start";
22     dataText.innerHTML = "";
23   }
24 };
```


Or you can do like in Listing 3, line 12, and embed the function directly into the `addEventListener()` method.

Finally, you start listening to your sensor with:

```
sensor.start();
```

There's not much more to it. Listing 3 shows how things would work when you put everything together for your app. Lines 1 to 3 map HTML elements (the select field, button, and text area) to JavaScript variables; then, on line 5, you listen for a click event on the button.

When a user clicks the button, you take the value from the selector (`Accelerometer`, `Gyroscope`, or `LinearAccelerationSensor`) and use it to initialize the sensor (line 9).

As you are only going to print the data out, the callback function (lines 12 to 14) is short. The callback function takes a parameter (`event`) that holds the data from the sensor.

The `event`'s `target` retrieves the element that triggered the event, in this case, the sensor itself. The `x`, `y`, and `z` attributes are common to the three sensors in the list – which is why they are in the list, as other sensors have different properties. The point is you print out the sensor's `x`, `y`, and `z` values on line 13.

And that's it. All told, the HTML for the front end is just 30 lines long, including the head section that you haven't even touched. And the code is fewer than 25 lines of pretty straightforward, uncomplicated JavaScript.

Trial Run

To build the app and launch it on your phone, first make sure your phone is in developer mode by going to *Settings | About phone* and scrolling down until you see the *Build number* section. Tap on that several times until your phone says you are a developer.

Connect your phone to your computer using a USB cable and move back to *Settings*. You will see there is a new submenu called *Developer options*. Tap on that and scroll down until you see the *USB debugging* option. Activate it.

Now check that Cordova can talk to your device by running the following instruction:

```
docker run --rm -i --privileged \
-v /dev/bus/usb:/dev/bus/usb \
beevolve/cordova adb devices
```

Cordova is using the Android Debug Bridge (`adb`) to try and locate your phone. The `devices` option shows a list of connected devices.

The first time around, your device may show up as unauthorized. This is normal. Go into *Settings | Developer options* again and make sure you have

enabled *USB debugging*. While you are there, run

```
docker run --rm -i --privileged \
-v /dev/bus/usb:/dev/bus/usb \
beevolve/cordova adb devices
```

again, and a dialog will pop up on your phone asking you to authorize your computer. Give your computer permission, and try listing your devices again. Your phone should now appear as available.

Now you can push your app to your phone with:

```
docker run --rm -i \
-v /$PWD:/workspace \
-w /workspace --privileged \
-v /dev/bus/usb:/dev/bus/usb/ \
beevolve/cordova cordova run \
android
```

Cordova will automatically install and run the app. The final result will look like Figure 8.

Conclusion

The universal sensor API combined with Cordova makes building sophisticated, sensor-enabled mobile apps ridiculously easy. If you know HTML and some basic JavaScript, you have all you need to get started.

In my next installment, you will learn how to extend your Cordova app, so it can transfer data to your computer. You will also learn how to integrate the data into a desktop application, so you can use your phone as a controller. ■■■

Info

- [1] "Tutorial – GPSD" by Paul Brown, *Linux Magazine*, issue 210, May 2018, p. 90: <http://www.linux-magazine.com/Issues/2018/210/Tutorial-gpsd/>
- [2] SSJ Creator: <https://play.google.com/store/apps/details?id=hcm.ssj.creator>
- [3] "Tutorial – Docker 101" by Paul Brown, *Linux Magazine*, issue 215, October 2018, p. 90: <http://www.linux-magazine.com/Issues/2018/215/Tutorials-Docker>
- [4] Cordova's plugin repository: <https://cordova.apache.org/plugins/>
- [5] The W3C's Generic Sensor API: <https://www.w3.org/TR/generic-sensor/>
- [6] A web page with a compass: <https://mobiforge.gitlab.io/sensors/compass.html>

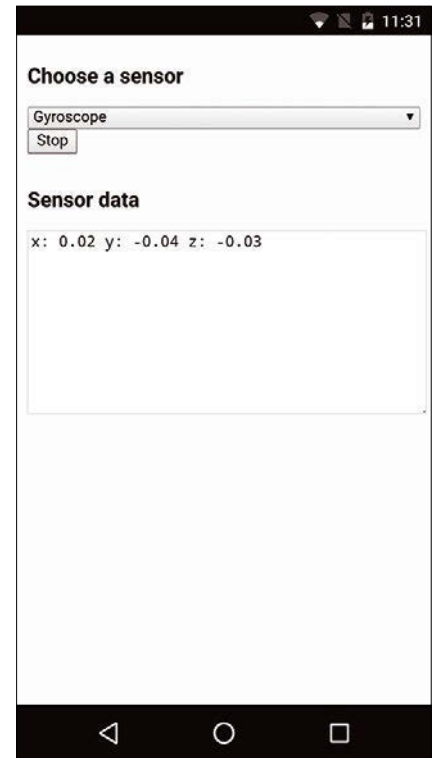


Figure 8: Your first sensor-aware Cordova app.

Needle in a Haystack

If your inbox is full of email messages with important attachments, retrieving those attachments manually can be a tedious task. The script presented in this article does this task automatically and can even save the email as a plain text file.

BY MARCO FIORETTI

Do you ever find yourself urgently searching for a file that you know you received as an email attachment but do not remember who sent it or when? Has your company saved all the important documents received via email somewhere easily retrievable? Would you like to save the content of all your email messages automatically as separate, plain text files?

Being able to copy automatically, into one folder and as separate files, all the email attachments and message bodies hidden in your email archives might save your day in situations like these. This tutorial explains how to do it with one relatively simple shell script and tools available from the standard repositories of most Linux distributions. Only basic knowledge of shell scripts is necessary. Additionally, patching the script to make it save just the attachment is also very easy.

MIME and Mailbox Formats

To process email messages, you need to know how files are attached to email and how email messages are archived inside digital mailboxes. To extract attachments from one email message, you need a MIME-aware processor that can split all the email's parts into separate files. Multipurpose Internet Mail Extensions (MIME) [1] is the open standard that describes how to:

- encode non-ASCII character sets in email bodies and headers,
- format message bodies with multiple parts (e.g., content in both HTML and plain text formats), and
- encode and attach to email any non-textual content (e.g., images to generic binary files).

To archive multiple messages as one mailbox, the simplest format is MBOX, which just concatenates all messages into one plain text file. Because MBOX is as simple as it is inefficient, other formats that store each message in a separate file inside a folder were developed [2]. One of the most common formats, and the one used in this tutorial, is called Maildir, which has the internal structure shown in Figure 1.

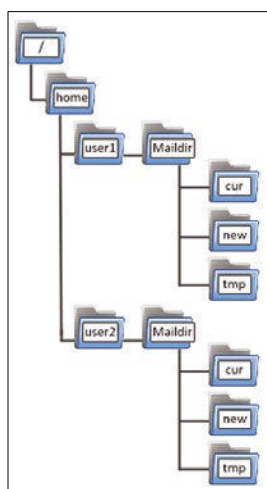


Figure 1: Maildir mailboxes have three subfolders: `new` for new email, `cur` for already read messages, and `tmp`, which is used by email software for processing.

Preliminary Work

Unless the mailbox you want to scan for attachments already is in Maildir format (or in any other format that puts every email into a separate file), you have to convert it. You can manually, but quickly, convert any mailbox to Maildir format in any Linux terminal using the Mutt email client:

1. Type:

```
mutt -R -m Maildir -f <YOUR MAILBOX>
```

2. Type `T` followed by the Enter key.

3. Type a semicolon (;) and then `s`.

The `-R` switch opens in read-only mode the mailbox defined with `-f`, and `-m` sets Maildir as the default mailbox format. By doing the second step inside Mutt, you tag *all* the messages in the mailbox. The third step makes Mutt ask you in which other mailbox all those messages should be saved, (i.e., copied, because the original mailbox is in read-only mode). Pass Mutt any mailbox name you want; when it has finished copying the messages, type `q` to exit.

If you have many mailboxes to convert, you can run Mutt from a script, as explained in an article online [3]. In any case, I recommend only working with copies of your mailbox, just to be on the safe side (not to mention that changing the access times of email files may confuse some email clients).

A Redundant, Overcautious Extractor

By default, the email extractor script that I introduce here extracts *everything* from email messages, not just attachments: message bodies, digital signatures, embedded images, and so on. With some very minor modifications, however, you can make it extract attachments only.

The code for this article is a shell script that contains three parts. For readability, I present them in three separate listings. The first part (Listing 1) defines all the necessary variables and folders, plus a shell function that generates unique folder names for each message.

I call the script “redundant” because its second part (Listing 2) does the actual job of saving all attachments and email bodies as separate files a total of four times, with four different tools. The final section (Listing 3) removes any empty or duplicate files produced by that process.

A Case for Redundancy

One reason to extract everything by default is just to show what is possible: If you really want or need it, you can automatically save the text content of every email you ever received as one single file.

The other reason to extract everything is the same reason why, by default, the script uses four different tools: the very nature and age of email. The good part of email being an open, decades-old standard is that we can still read very old messages. The bad part, especially when dealing with *old* email archives, is that, over the years, countless email applications of variable quality, plus careless email users, applied that standard with great creativity.

A generic old email archive may contain attachments in long-forgotten formats or named with weird characters with non-standard extensions, or none at all.

Email bodies can be nested in the strangest ways: For example, think of email, in both text and HTML format with attachments, that has been forwarded and then replied to from an email digest. Even a message’s “plain text” may be encoded in any way possibly conceived over the last 50 years. Oh, and have you ever tried to count how many different combinations of language and format there are to write a “simple” date?

Even the way in which archives have been stored on disk is relevant. Think of a Maildir first compressed and saved to a floppy, later copied on a CD-ROM and then back onto a hard drive, and finally to the cloud – every time on a different filesystem. In general, each of these migrations might alter the name, access, or creation time of an email file.

This is why the script does the “same” job four times. I have found that, no matter how you set it, each tool returns a slightly different set of files. However, for the reasons above, I cannot predict which of them might return the best file (whatever that means) for all the mailboxes that you may encounter. So you get four tools, and you decide whether to keep them all or not. I will discuss this issue later further in the tutorial.

Regardless of which tool you use and how, it is almost certain that you will end up with lots of duplicate or useless files. Even one single email thread can produce duplicates because of emojis, HTML email with the same image in the signature of each message, or an attachment that was sent to many people with you CC’d each time. There is no way to avoid this, which is the reason for last part of the script.

Because of space constraints, I cannot describe in detail all the options used for the several commands presents in the code: To know how they work, please check each command’s man pages.

The script takes two arguments. The first, saved in the `MAILBOX` variable, is the Maildir folder that the script should scan for attachments. The second, `TARGET`, is the folder where it should save the files it finds.

`TMPDIR` is, as its name suggests, just a temporary work folder, and `CNT` is a counter used to generate unique names for its subfolders. The purpose of the associative array `CHECKSUMS` defined in line 7 is explained in Listing 3.

Lines 9 to 18 define a shell function called `emaildirname` that first defines three variables: the time the file containing the current email message was created (`MSGTS`) and the time that email was received (`MSGDATE`) and its subject (`MSGSUBJ`).

Keep in mind the many different date formats, which can also be ambiguous if they don’t mention a time zone. File creation times are often not preserved when files are copied from one medium to another. Consequently, more often than not, *neither* of those timestamps will be correct. They are just the least worst guess possible to make in a simple shell script.

In practice, line 10 saves in `MSGTS` the timestamp of the current email file, expressed in seconds in Unix time (i.e., seconds that have elapsed since January 1, 1970). That number is then converted

Listing 1: Preparation

```
01 #! /bin/bash
02
03 MAILBOX="$1"
04 TARGET="$2"
05 TMPDIR="$HOME/extractor-tmp"
06 CNT=0
07 declare -A CHECKSUMS
08
09 emaildirname() {
10   MSGTS=`stat -c %Y $EMAIL`
11   ORIGTS=`date -d @$MSGTS '+%Y%m%d%H%M.%S'`
12   FILENAMETS=`date -d @$MSGTS '+%Y%m%d%H%M%S'`
13   MSGDATE=`grep '^Date: ' $EMAIL | cut -c7- | head -1`
14   MSGSUBJ=`grep '^Subject: ' $EMAIL | cut -c10-210 | head -1`
15   DIR=`echo $MSGDATE-$MSGSUBJ | sed -r 's/[^a-zA-Z0-9]+/-/g' |`
16     `sed -r 's/-+$//'`
17   DIR="$FILENAMETS-$DIR-$CNT"
18   let "CNT=CNT+1"
19 }
20 rm -rf $TMPDIR $TARGET
21 mkdir -p $TMPDIR $TARGET/tmp
22
23 echo "EXTR mbox      : $MAILBOX"
24 printf "EXTR contains  : %s messages ( %s Kbytes )\n"
25     `find $MAILBOX -type f | wc -l` `du -sk $MAILBOX | cut -f1`
26 echo "EXTR target   : $TARGET"
```

with the `date` command in two different formats in `ORIGTS` and `FILENAMETS` (lines 11-12), for reasons that will become clear later.

`MSGDATE`, in line 13, stores the content of the first `Date:` header in the current email (there may be many of them, if the email is a reply to a reply to a reply). `MSGSUBJ`, instead, stores the first 200 characters (minus the initial `Subject:` substring) of the subject email header.

The `emaildirname` function's main job is to generate a unique folder name (`DIR`) that includes, in a more or less readable way, both the current received date and subject of the email. To do this, line 15 concatenates the two corresponding variables, replacing with `sed` any non-alphanumeric characters with dashes. Next, to guarantee that `DIR` has a unique value, the `$CNT` variable is appended right before incrementing it. Without this trick, two distinct copies of the same email would be processed in the same folder.

In this script, any line containing the strings `EXTR` or `DUPX` simply print to the terminal some information about what is happening. Here, lines 23 to 25 print the name of the input mailbox, how many messages it contains, how big it is, and where the attachments will be stored.

The four Linux command-line tools – all available as binary packages in the most popular distri-

butions – with the ability to split the email parts into separate files are: `mu`, `uudeview`, `munpack`, and `ripmime`. Listing 2 runs them all, one at a time, on all of the given `$MAILBOX` messages by means of the `for` loop starting in line 29. A separate folder for each tool is created inside `$TMPDIR` (line 31), and the `$CNT` variable is reset.

Then, for each single email message contained in the given `$MAILBOX` (line 35), four things happen in sequence. First, the `emaildirname` function updates the values of the variables previously described. Second (lines 38-39), the script creates a unique folder to store all the current email parts. Third, the `case` statement (line 40) passes to the active tool the current email, making it dump all its separate parts inside that same folder (lines 42, 46, 50, and 54). A numeric index `$TOOLNUM` is also associated with each tool (more on this later).

Sometimes, the email decoder tool will find nothing to save inside the `$TMPDIR/$TOOL/$DIR` folder. In that case, the loop started in line 35 will just move on to the next email.

If `$NUMFILES` is greater than zero (lines 59-60), `$TMPDIR/$TOOL/$DIR` will contain potentially tons of files (e.g., plain text body, HTML body, many icons, attachments, etc.) with the weirdest possible names and often without any extensions. There-

Listing 2: Extraction

```

26-28 <these lines only contained comments...>
29 for TOOL in mu uudeview munpack ripmime
30 do
31     mkdir -p $TMPDIR/$TOOL/
32     printf "EXTR\nEXTR %-10s: %-9s start\n"
           $TOOL `date +%H:%M:%S`
33     CNT=0
34
35     for EMAIL in `find $MAILBOX/ -type f`
36     do
37         emaildirname
38         mkdir $TMPDIR/$TOOL/$DIR
39         cd $TMPDIR/$TOOL/$DIR
40         case $TOOL in
41
42             ripmime)
43                 TOOLNUM=3
44                 ripmime -i $EMAIL --paranoid ;;
45
46             munpack)
47                 TOOLNUM=2
48                 munpack -t -q $EMAIL > /dev/null ;;
49
50             uudeview)
51                 TOOLNUM=1
52                 uudeview +a -m -n -q -i $EMAIL ;;
53
54             mu)
55                 TOOLNUM=0
56                 mu extract -a $EMAIL ;;
57         esac
58
59         NUMFILES=`find . -type f | wc -l`
60         if [[ "$NUMFILES" -gt "0" ]]
61         then
62             find . -type f | cut -c3- > /tmp/file_list.txt
63             while IFS= read -r file
64             do
65                 NEWNAME=`echo ${file%.*}`
66                 NEWNAME=`echo $NEWNAME | sed -r 's/[^a-zA-Z0-9]+/-/g' |
                           sed -r 's/-+//`
67                 EXT=`echo $file | awk -F . '{print $NF}`
68
69                 if [ "$EXT" == "$NEWNAME" ]
70                 then
71                     EXT='probablyemailbody.txt'
72                 fi
73                 mv -- "$file" $FILENAMETS-$TOOLNUM-$CNT-$NEWNAME.$EXT
74                 touch -t $ORIGTS $FILENAMETS-$TOOLNUM-$CNT-$NEWNAME.$EXT
75                 done < /tmp/file_list.txt
76             fi
77
78         done
79         printf "EXTR %-10s: %-9s end\n" $TOOL `date +%H:%M:%S`
80         printf "EXTR %-10s: %-9s files extracted (%7s empty)\n"
           $TOOL `find $TMPDIR/$TOOL/ -type f | wc -l`
           `find $TMPDIR/$TOOL/ -type f -empty | wc -l`
81
82         find $TMPDIR/$TOOL/ -type f -exec mv -i {} $TARGET/tmp \;
83
84     done

```

fore, the script saves all those file names into a list (line 62) then reads them all with the `while` loop of line 63, and renames all the corresponding files.

This fourth step is necessary to make both the rest of the script and, generally speaking, your future management of those files easier. Trust me on this.

The renaming that happens in lines 65 to 67 is probably the blackest piece of magic of the whole thing but luckily is much easier to explain:

- Line 65 is standard Bash syntax to save into `$NEWNAME` the name of the current file, *minus* its extension, if present.
- Line 66 just repeats the same trick from line 15 (Listing 1): All the non-alphanumeric characters become dashes.
- Line 67 is, like line 65, standard Bash voodoo to fetch the extension (if present) of a file name.

If a file has no extension, both `$EXT` and `$NEWNAME` will have the same value. In my experience, at least 95 percent of the time, this happens with files that are the body of an email, in plain or HTML format.

Therefore, I just give those files the telling `probably_emailbody.txt` extension (lines 69 to 72).

Finally, line 73 gives the file a new name composed by `$FILENAME`s and calculated by the `emaildirname` function, plus the current `$TOOLNUM`, `$CNT`, `$NEWNAME`, and, of course, its extension.

The `-t` option of the `touch` command in line 74 sets the freshly renamed file's creation timestamp to the creation timestamp contained in `$ORIGTS`. That value is the same as `$FILENAME`s, but because the `-t` option wants a different format, I put it into a separate variable for clarity.

The final part of the main loop (lines 79 to 82) just does some reporting and moves all the files that were extracted and given unique names to the `$TARGET/tmp` folder.

At this point in the script, all the files extracted by all the tools are inside the folder `$TARGET/tmp`. Depending on the content of your mailboxes and on how you tweaked the script, you may also get a lot of empty files: Line 86 (Listing 3) finds and deletes them.

Now is the time to tackle a bigger problem: Regardless of how the code was tweaked, that folder will contain many duplicate files (Figure 2). Removing them is the task of the loop in lines 92 to 104.

Line 92 finds and sorts by name all the files inside `$TARGET/tmp` to look at them one by one. Line 94 stores the checksum of the current file in the `$CK` variable. If the `CHECKSUMS` associative array declared in line 7 (Listing 1) already contains a key equal to `$CK` (line 92), it means the loop has already found a first copy of that file, and so it can safely remove the other copy found in this iteration (to be precise, it leaves it in a folder that will be moved elsewhere in line 106, but it has the same result). Therefore, line 97 prints what will happen, and line 98 increments the counter of duplicate files.

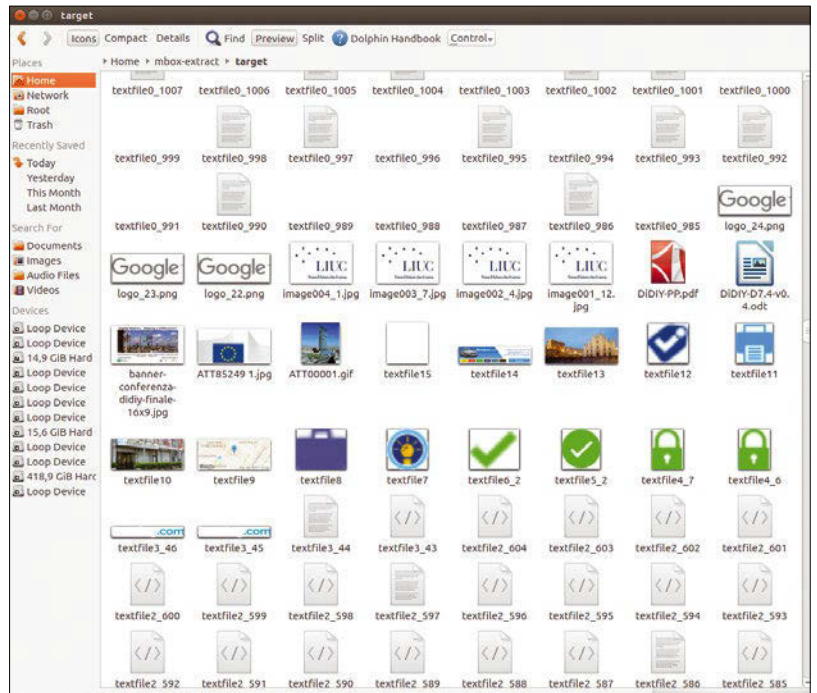


Figure 2: Running the script with the default options will produce hundreds of files with meaningless names and lots of duplicates.

If there is no key equal to `$CK` inside `CHECKSUMS`, then `$F` is the first copy of that particular file. Therefore, it is moved to the parent directory, and another key-value element for `$CK` is added to the `CHECKSUMS` array in line 102.

By now, you can also appreciate all the effort put into giving each file a unique name that starts with its (assumed!!!) creation time: Be-

Listing 3: Cleanup

```

85
86 find $TARGET/tmp -type f -empty -exec rm {} \;
87
88 printf "EXTR\nEXTR cleaning : %-9s start\n" `date +%H:%M:%S`
89
90 CNT=0
91
92 for F in `find $TARGET/tmp -type f | sort`
93 do
94   CK=`md5sum $F | sed 's/ .*$/ ' `
95   if [ "${CHECKSUMS[$CK]}" == "found" ]
96   then
97     echo "DUPX: removing $F"
98     let "CNT=CNT+1"
99   else
100    echo "DUPX: keeping $F"
101    mv $F $TARGET/
102    CHECKSUMS[$CK]='found'
103   fi
104 done
105
106 mv $TARGET/tmp $TARGET-tmp
107
108 printf "EXTR total : %-9s files found, after removing
% s duplicates\n" `find $TARGET -type f | wc -l` $CNT
109 printf "EXTR cleaning : %-9s end\n" `date +%H:%M:%S`
110 >

```

cause line 92 sorts files by name, the first copy of any file that will be processed (i.e., the only one that will be kept) also is the one with the *oldest* timestamp. For the same reason, should you want to keep the newest copy instead, you should just add the `-r` (reverse) sort switch, in line 92.

The last four lines of code generate some final statistics, after moving the `$TARGET/tmp` folder out-

side of `$TARGET`, so you can still look at all those files if something went wrong.

Scanning a Real Mailbox

To give you an idea of the performance of both the complete script and each of the individual tools it uses, I ran the script three times on one of the largest mailboxes in my email archive. In the first run, I used all four tools, with exactly the options shown in Listings 1 through 3, to save every component of each message. Listing 4 shows the complete report generated by the script (compare it with the source code to understand exactly how each piece of information was generated):

The report shows that my email archive for June 2017 contains 1,685 messages and takes about 150MB of disk space. It also shows that the time spent by each tool, in addition to the number of files it found, varies greatly.

The `mu` program/tool saved 348 files in less than two minutes, whereas `ripmime` saved more than 4,300 files in six minutes – including more than 1,000 empty files (not its fault, as explained below). Eventually, more than 1,700 duplicates were removed, leaving more than 5,300 files, between attachments, email bodies, digital signatures, and whatnot in the `attachments-2017.06` folder.

The second run, I made `munpack` and `ripmime` ignore almost everything but the attachments by:

- Not passing the `-t` switch to `munpack`. This means “ignore the plain text MIME parts of multipart messages.”
- Adding the `--no-nameless` switch to `ripmime` to make it skip nameless attachments.

Running the script on the same mailbox in “attachment-only mode” produced the results shown in Listing 5.

As expected, both `munpack` and `ripmime` were much more efficient, and overall, the script produced many fewer files to deal with manually. However, a direct inspection of this run’s 846 results showed that about three quarters of them were email bodies, not attachments, almost all extracted by `uudeview` or `munpack`. Therefore, in the third run, I removed those two tools from line 29 (Listing 2), leaving the script to use only `mu` and `ripmime`. Listing 6 shows what remained after deduplication.

This third run did produce, almost exclusively, actual attachments (plus a few logos embedded in HTML email). All the numbers prove, I hope, why I present a script that can extract every single part of an email message in four different ways. Email messages can be so different from each other that, especially in older archives, one tool might find what the others miss.

The reports show that *if* the content of your email archive is similar to mine and you only care about attachments, you can very likely get away with enabling only `mu` and `ripmime` with the `--no-nameless` option.

Listing 4: First Run – All Four Tools

```
EXTR mbox      : /home/marco/mbox-extract/original/2017.06
EXTR contains  : 1685 messages ( 150224 KBytes )
EXTR target    : /home/marco/mbox-extract/mu-2017.06
EXTR
EXTR mu        : 08:57:25   start
EXTR mu        : 08:59:10   end
EXTR mu        : 348       files extracted (    0 empty)
EXTR
EXTR uudeview  : 08:59:11   start
EXTR uudeview  : 09:01:06   end
EXTR uudeview  : 648       files extracted (    0 empty)
EXTR
EXTR munpack   : 09:01:09   start
EXTR munpack   : 09:03:50   end
EXTR munpack   : 2780      files extracted (    1 empty)
EXTR
EXTR ripmime   : 09:03:58   start
EXTR ripmime   : 09:07:18   end
EXTR ripmime   : 4334      files extracted (  1017 empty)
EXTR
EXTR cleaning  : 09:07:32   start
EXTR total     : 5369      files found, after removing 1723 duplicates
EXTR cleaning  : 09:08:42   end
```

Listing 5: Second Run – Attachment-Only Mode

```
EXTR mu        : 10:52:23   start
EXTR mu        : 10:54:07   end
EXTR mu        : 348       files extracted (    0 empty)
EXTR
EXTR uudeview  : 10:54:08   start
EXTR uudeview  : 10:55:50   end
EXTR uudeview  : 648       files extracted (    0 empty)
EXTR
EXTR munpack   : 10:55:52   start
EXTR munpack   : 10:57:34   end
EXTR munpack   : 655       files extracted (    0 empty)
EXTR
EXTR ripmime   : 10:57:36   start
EXTR ripmime   : 10:59:25   end
EXTR ripmime   : 325       files extracted (    2 empty)
EXTR
EXTR cleaning  : 10:59:26   start
EXTR total     : 846       files found, after removing 1128 duplicates
EXTR cleaning  : 10:59:43   end
```

Listing 6: Third Run – mu and ripmime

```
EXTR cleaning  : 11:35:08   start
EXTR total     : 245       files found, after removing 426 duplicates
EXTR cleaning  : 11:35:16   end
```

At the same time, if you also need to save the bodies of your messages in separate files, if you have old archives created with non-standard clients, or if you want to be as sure as technically possible that you did not miss anything, you can get there using all four tools sequentially and waiting just a few more minutes. Again, in my opinion this is a decision that you have to make. I do suggest, however, to make at least one trial run with all four tools.

Other Scenarios

Regardless of how you configure it, this script is a lifesaver whenever you have to recover one or many attachments quickly from large email archives. Of course, in many cases it would not be enough or it would be overkill. For lack of space, I cannot explore all those cases in detail, but the following sections have some pointers on how to handle the most common possibilities.

Searching Many Mailboxes

If you need to look into more than one mailbox, there are two ways to do it. The easiest way is to copy all of mailboxes manually into a new mailbox with any email client and run the script only on the new folder. The other method is to call the script inside a loop from another script, which could look more or less like this:

```
for MAILBOX in mailbox1 mailbox2...
do
    attach-extractor.sh $MAILBOX $MAILBOX-target
done
```

This is also the way to go when, for whatever reason, you need to know the mailbox from which an attachment came.

Searching Just a Few Messages

If you are sure that the attachment(s) you are looking for are somewhere inside a specific thread, or from a specific person, processing a whole mailbox to extract them would be a waste of time. As in the previous case, you can copy only

those specific messages into a new mailbox and launch the script on it. Although still overkill, this method works with any email client.

A more efficient solution, instead, is to tag all those messages and tell your email client to find and save all the attachments they contain, all by themselves. A StackExchange post shows you how to write a Mutt macro that does this [4].

Extracting Attachments On Arrival

Sometimes it is useful to have all email attachments extracted and saved to a dedicated folder automatically, the moment each email arrives. This is possible by configuring the `procmail` tool (explained in detail online [5]).

Conclusions

The script presented here does not spare you the unavoidable task of manually separating the wheat from the chaff. If you run the script on a large mailbox, the result will be many files with either cryptic or very similar names (Figure 3). In the latter case, finding out which file is the true final (or initial) version requires manual examination.

Also, be prepared to fix permissions and ownership of files manually. By default, email folder and files permissions on Linux are set to `600`, which means “only readable by the owner.” Depending on how you configure the script, many of the files it extracts will have the same permissions, which may or may not be what you want.

Final thought: Some weird combination of character encodings and recursively embedded messages surely exists out there that would make this extraction script fail and requiring tweaking or other manual work. Unfortunately, there is nothing to be done about this scenario. However, considering that some files from just 15 or 20 years ago are already unreadable, you should be happy that you can still process all email messages ever created without particular problems. This all goes to prove that the best “innovation” is based on simple and really open standards. ■■■

The Author

Marco Fioretti (<http://mfioretti.com>) is a freelance author, trainer, and researcher based in Rome, Italy. He has been working with free/open source software since 1995 and on open digital standards since 2005. Marco also is a Board Member of the Free Knowledge Institute (<http://freeknowledge.eu>).



Figure 3: Running the script with the most restrictive options: 99.99 percent are real attachments, without any duplicates.

Info

- [1] MIME: <https://www.hunnsysoft.com/mime/mime-guide.html>
- [2] Email format overview: <https://wiki2.dovecot.org/MailboxFormat>
- [3] Using Mutt as a mailbox converter: <https://foolab.org/node/1737>
- [4] Save tagged attachments with Mutt: <https://unix.stackexchange.com/questions/37218/how-to-really-easily-save-all-tagged-attachments-in-mutt>
- [5] procmail: <https://unix.stackexchange.com/questions/421433/procmail-save-attachment-with-received-date-in-filename>



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.

Open Source Summit Europe

Date: October 22-24, 2018

Location: Edinburgh, United Kingdom

Website: <https://events.linuxfoundation.org/events/open-source-summit-europe-2018/>

Open Source Summit Europe (OSSEU) is the leading conference for developers, architects, and other technologists – as well as open source community and industry leaders – to collaborate, share information, learn about the latest technologies and gain a competitive advantage by using innovative open solutions.

SC18

Date: November 11-16, 2018

Location: Dallas, Texas

Website: <https://sc18.supercomputing.org/>

SC18 is the international conference and exhibition for high performance computing, networking, storage, and analysis. Discover one of the broadest programs of incredible learning opportunities of all HPC conferences, and explore the latest from the world's leading HPC exhibitors, research organization, and universities.

LIBRECON powered by CEBIT

Date: November 21-22, 2018

Location: Bilbao, Spain

Website: <https://www.librecon.io/?lang=en>

LIBRECON powered by CEBIT is the reference event for open source technologies in the south of Europe. The event brings together open technology professionals in fields such as Internet of Things (IoT), Intelligent Machines, Future Mobility, Digital Transformation, Cybersecurity, and more.

Events

LinuxDay Vorarlberg	October 13, 2018	Dornbirn, Austria	https://www.linuxday.at/
Open Source Automation Day	October 16, 2018	Munich, Germany	https://osad-munich.org/
All Things Open	October 21-23, 2018	Raleigh, North Carolina	https://allthingsopen.org/
Open Source Summit Europe	October 22-24, 2018	Edinburgh, UK	https://events.linuxfoundation.org/events/open-source-summit-europe-2018/
EclipseCon Europe 2018	October 22-25, 2018	Ludwigsburg, Germany	https://www.eclipsecon.org/europe2018
LISA18	October 29-31, 2018	Nashville, Tennessee	https://www.usenix.org/conference/lisa18
Open Rhein/Ruhr	November 3-4, 2018	Oberhausen, Germany	http://openrheinruhr.de/
Web Summit 2018	November 5-8, 2018	Lissabon, Portugal	https://websummit.com/
Open Source Monitoring Conference 2018	November 5-8, 2018	Nuremberg, Germany	https://osmc.de/
OSCamp on Puppet	November 8, 2018	Nuremberg, Germany	https://opensourcecamp.de/
Linux Presentation Day 2018.2	November 10, 2018	Europe-wide in many cities	http://www.linux-presentation-day.de/
SC18	November 11-16, 2018	Dallas, Texas	https://sc18.supercomputing.org/
Linux Kernel Summit	November 12-15, 2018	Vancouver, Canada	https://events.linuxfoundation.org/events/linux-kernel-summit-2018/
Open Stack Summit	November 13-15, 2018	Berlin, Germany	https://www.openstack.org/summit/berlin-2018/
LIBRECON powered by CEBIT	November 21-22, 2018	Bilbao, Spain	https://www.librecon.io/

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

Erik Bärwaldt	26, 70
Swapnil Bhartiya	8, 14
Paul Brown	85
Zack Brown	10
Bruce Byfield	40, 64
Bradley Campbell	16
Joe Casad	3
Mark Crutch	67
Marco Fioretti	90
Jon "maddog" Hall	69
Roman Jordan	34
Charly Kühnast	51
Christoph Langner	80
Vincent Mealing	67
Pete Metcalfe	52
Graham Morrison	74
Mike Schilli	46
Scott Sumner	56
Alexander Tolstoy	22
Stefan Wintermeyer	44

Contact Info

Editor in Chief

Joe Casad, jcasad@linux-magazine.com

Copy Editor

Amy Pettle

News Editor

Swapnil Bhartiya

Editor Emerita Nomadica

Rita L Sooby

Localization & Translation

Ian Travis

Layout

Dena Friesen, Lori White

Cover Design

Dena Friesen

Cover Image

© floralset, 123RF.com

Advertising

Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 3090 5128

Marketing Communications

Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
2721 W 6th St, Ste D
Lawrence, KS 66049 USA

Publisher

Brian Osborn

Customer Service / Subscription

For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America

www.linux-magazine.com – Worldwide

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the disc provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2018 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media USA, LLC, unless otherwise stated in writing.

Linux is a trademark of Linus Torvalds.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by hofmann infocom GmbH on recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 2721 W 6th St, Ste D, Lawrence, KS, 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 2721 W 6th St, Ste D, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany.



UK / Europe
USA / Canada
Australia

Nov 03
Nov 30
Dec 03

Issue 217 / December 2018

Innovative Distros

Hundreds of Linux distros exist in the world, but Linux forums (and Linux magazines) often just focus on just a few of the most popular. Next month, we'll shine the light on some unsung distributions that are worthy of a little more attention.

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

Photo by Alex Iby on Unsplash



Register Today

Register before October 15 and save.

sc18.supercomputing.org



Discover a broad program of HPC-focused technical presentations, papers, workshops, informative tutorials, timely research posters, and more.



Explore exhibits showcasing the latest HPC technologies from the world's leading vendors, universities, and research organizations.



Enjoy several days in Dallas, one of the country's top tech cities, and home to great amenities, food, and nightlife. Bring the family!

November 11–16, 2018

The International Conference for High Performance Computing, Networking, Storage, and Analysis



SC18

Dallas, TX | **hpc inspires.**

Sponsored by: