FREE DVD

FEATURING UBUNTU 18.10 "Cosmic Cuttlefish"

CUSTOMIZE THE BOOT MENU

LINUX PRO MAGAZINE
FEBRUARY 2019

CUSTOMIZE THE BOOT MENU

Clean up your startup for clarity and fewer mistakes

PCIe SSDs
Make the storage fit your hardware

IoT Tricks
Store data in memory with Redis

Exploring Ubuntu 18.10

Pulse Sensor
Measure your heartbeat with a Raspberry Pi

EncryptPad
Text editor with easy encryption

LINUXVOICE

- EasySSH
- Saving Analog Data
- maddog: RISC-V Architecture

FOSSPicks
- Sweet Home 3D 6
- Textosaurus
- Flare: Empyrean Campaign

Tutorials
- Shell Variables
- Natron Video Effects

Issue 219
Feb 2019
US$ 15.99
CAN$ 17.99

HETZNER ONLINE

HETZNER CLOUD NOW WITH BLOCK STORAGE

e.g. Hetzner Cloud Server CX11

- ✔ Intel® Xeon® Skylake
- ✔ 1 vCPU
- ✔ 2 GB RAM
- ✔ 20 GB NVMe SSD
- ✔ 20 TB traffic*
- ✔ Intuitive Cloud Console
- ✔ Free DDoS protection
- ✔ Located in Germany or Finland

monthly $ **2.83**

Hetzner Cloud with flexible SSD storage

Volumes offer highly available and reliable SSD block storage for your cloud servers. You can expand a Volume to 10 TB each and attach up 16 to a cloud server, so they're great for jobs with high storage requirements.

Our award-winning Hetzner Cloud combines high performance hardware with intuitive usability all at an incredibly low price.

Volumes monthly $ **4.55** per 100 GB

cloud.hetzner.com

* With 20 TB of included traffic, you'll have lots of bandwidth for your projects, regardless of which Hetzner Cloud package you choose. But if you need, you may add more for an extra $1.14 a month per TB.

VIGILANCE

Dear Reader,

It is getting harder to write this column. I used to have a ready supply of topics, with all the outrageous things that were happening to Linux: Microsoft's disinformation, SCO's lawsuit, patent licenses, the Novell/Microsoft pact – big things that threatened the very existence of the Linux community.

I was trying to think up a new topic today, and it occurred to me that there used to be way more in the news on an average day that could rile up a Linux guy. That's the good news, because Linux is in a safer place and is no longer faced with the threat of imminent destruction. Microsoft is playing nice (sort of); SCO has collapsed under the weight of its own imagination deficit. But are we really walking on easy street now? Surely some other threats must be out there? Are there still factors that are threatening the livelihood of the Linux community, and if so, what are they?

That sounded like a good topic for a column, so I resolved to create my own list of the current top threats. One note on this list: Because these threats are not quite as dire as they used to be, they are also a little more arbitrary. This is my list – some of you might see different threats, but either way, the main point is that challenges still exist:

- **Fragmentation** – the Linux desktop still isn't unified, and the recent controversy over the systemd init daemon is a reminder that the community does not always move in the same direction. The beauty of open source is that you can always fork the code, but if too many developers take the code in too many different directions, the project could lose the critical mass necessary to hold the mainstream, becoming a collection of smaller projects, like the BSDs, that will receive less attention from hardware vendors and, ultimately, users.
- **Irrelevance** – will the general-purpose computer OS still be a thing in 10 years? Already, people are doing more with their cell phones and tablets. Linux is still running inside Android, but there are so many other things going on inside a smartphone that you can't exactly just hack on it like you can on a Linux system. Linux would still be running on toasters and washing machines (and on servers – see the next item), but it could recede into the background and be more under the control of hardware and cloud companies, rather than driven by a vibrant, independent community.
- **Cloud computing** – According to Free Software Foundation president Richard Stallman, cloud servers "… wrest control from the users even more inexorably than proprietary software" [1]. Software running from within a proprietary portal maximizes the power of the vendor and minimizes the freedom of the user in ways that are antithetical to the spirit of Free Software. Also, the copyleft protection of the GPL, which forces the sharing of source code when changes are made to a program, is triggered when the software is *distributed*. As many have pointed out, cloud computing doesn't really distribute the software, so it falls in a gray area that is beyond the protection of Free Software licensing.
- **Re-emergence of a "friendlier" Microsoft** – Microsoft is no longer bent on destroying Linux; in fact, they say they "love" Linux. But a little too much love from Microsoft could be a scary thing too. Many in the Linux community distrust Redmond's motives and wonder if some kind of assimilation might be taking place. The GPL offers some natural defenses against a single company gaining control, but could Microsoft use its cash stores and market clout to take Linux in a direction that the greater community doesn't want to go, and what would happen if they did?
- **Succession issues** – Linux is still run by the same guy who created it 27 years ago. Linus Torvalds is still young and healthy, but he might not want to do this forever. Will Linux survive the handoff to a new generation of leaders?
- **Bad judges and politicians** – Linux and open source licensing have survived several tests in the courts over patent law, copyright law, and other intellectual property issues, but the questions are complex and lots of politicians, business leaders, and jurists still don't exactly get what's going on with open source. Actually, I sometimes wonder if the open source community totally gets it. (Many voices in the community have called for a loosening of copyright laws to increase the freedom to consume music and movies, but actually, a strong copyright is the foundation on which the GPL is built, so you have to be very careful.) These issues are too vast and intricate to sort out in a one-page intro column, but suffice it to say, a few bad decisions from judges or regulators could bring back questions that we all thought were settled.

That's my list – at least for now. I'm not saying all this stuff is actually going to happen, but, as with any challenge, the best way to keep these dark threats in abeyance is to be aware and not get too complacent. As the old saying goes "Eternal vigilance is the price of liberty" [2].

Joe Casad,
Editor in Chief

Info

[1] "Who Does That Server Really Serve?" by Richard Stallman: *https://www.gnu.org/philosophy/who-does-that-server-really-serve.en.html*

[2] No one is totally sure who said this first: *http://www.thisdayinquotes.com/2011/01/eternal-vigilance-is-price-of-liberty.html*

LINUX MAGAZINE

WHAT'S INSIDE

That drab default boot menu on your Linux computer isn't just boring; it also could be confusing users and obscuring important boot choices. This month we show you how to add clarity and visual appeal to the GRUB boot menu with GRUB themes and the Grub Customizer. And while we're on the subject of GRUB, check out the Command Line column this month for a closer look at GRUB 2 passwords and encryption.

Also in this month's issue:

- **Ubuntu 18.10 "Cosmic Cuttlefish"** – Ubuntu is back on Gnome and back to normal (page 24).
- **PCIe SSDs on old hardware** – if you're planning on upgrading the storage on an old computer to a PCIe SSD drive, you'd better study the options before you buy (page 34).

Also inside, check out MakerSpace for a story on how to build a heart monitor using a Raspberry Pi, and read on to LinuxVoice for a look at converting your old analog audio and video files to digital format.

SERVICE

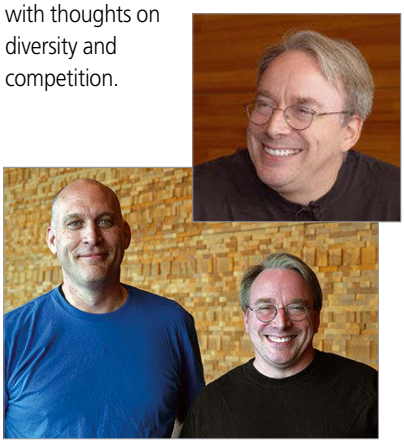
3 Comment
6 DVD
96 Featured Events
97 Call for Papers
98 Preview

NEWS

08 News
- Microsoft Gets an Open Source Web Browser
- Canonical Launches MicroK8s
- A New Raspberry Pi Board
- OpenStack Foundation Changes Name of the OpenStack Summit
- Red Hat Enterprise Linux 8 Beta
- System76 Announces a Line of US-Made PCs

12 Kernel News
- KUnit Testing Framework
- Removing the Ancient 00-INDEX Files
- Identifying Process Termination Without Polling

16 LINUS TORVALDS AND GREG K-H
Linux creator Linus Torvalds describes some recent challenges and opportunities for the Linux kernel – and stable branch maintainer Greg Kroah-Hartman joins in with thoughts on diversity and competition.

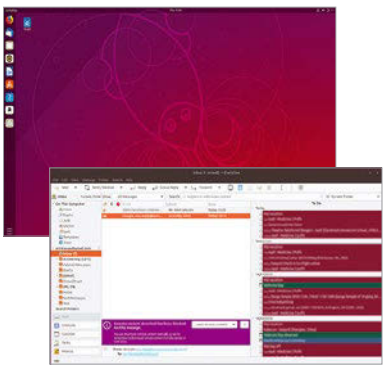


COVER STORIES

20 Grub Customizer
Is the simple black and white GRUB menu causing confusion and obscuring important choices? Why not customize with GRUB themes and the Grub Customizer?

REVIEWS

24 Ubuntu 18.10
Ubuntu Linux gets back to basics with the Ubuntu 18.10 release – an appealing and practical distro that isn't worried about conquering the world.



IN-DEPTH

30 EncryptPad
EncryptPad provides symmetric text encryption directly from the editor. You can also use EncryptPad to encrypt binary data.

34 PCIe SSD
A PCIe SSD can accelerate your system considerably, but you need to do your homework and choose the right product for your computer.

40 Programming Snapshot – goroutines
In the Go language, program parts that run simultaneously synchronize and communicate natively via channels. Mike Schilli whips up a parallel web fetcher to demonstrate the concept.

44 Command Line – GRUB 2
More than just a boot manager, GRUB 2 can help you add another line of protection to your security defenses.

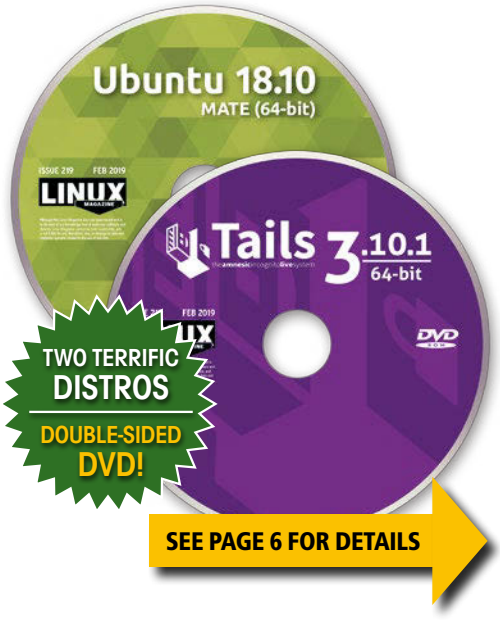
47 Charly's Column – moreutils
This month, sys admin columnist Charly dumps the moreutils toolbox on his workbench and takes combine and vidir for a spin.

MAKERSPACE

50 IoT with Redis Servers
Include binary data and image files in your IoT projects with Redis, an open source, in-memory data structure store.

56 DIY Pulse Sensor
A pulsemeter built with a Raspberry Pi, a digital-to-analog converter, and an optical sensor monitors your heart rate just as well as many far more expensive medical devices.

60 Open Hardware – Librem 5
Despite a few ups and downs in development, Purism moves ahead in its quest to produce a free and secure phone.



SEE PAGE 6 FOR DETAILS

LINUXVOICE

63 Welcome
This month in Linux Voice.

65 Doghouse – RISC-V
As open source software development for the RISC-V architecture moves ahead, maddog says stop complaining and start contributing to the project.

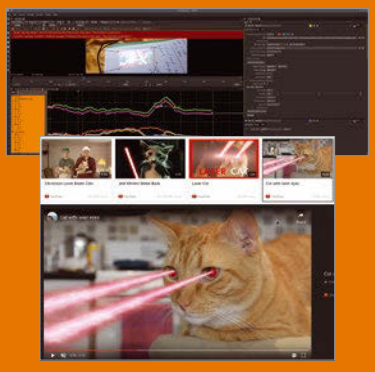
66 EasySSH
EasySSH lives up to its name and starts SSH connections at the click of a mouse.

70 File Conversion
The hardware to play old VHS tapes or audio cassettes is becoming more difficult to find. We'll show you how to convert those old analog treasures to digital format for future enjoyment.

78 FOSSPicks
Graham nearly made it through an entire month of FOSSPicks without an esoteric audio discovery. And then he found VeeSeeVSTRack.

84 Tutorials – Shell Scripting
You do not need to learn low-level programming languages to become a real Linux power user. Shell scripting is all you need.

90 Tutorials – Natron
Natron gives you the power to apply sophisticated effects to your videos.



On the DVD

Tails 3.10.1

Tails is a Live Linux system that puts the focus on privacy and anonymity. The Tails system comes with the Tor browser and all the tools you'll need to operate privately on Tor networks. Tails also comes with state-of-the-art cryptography tools and other tools you can use to surf invisibly and leave no trace.

Ubuntu MATE 18.10

MATE is a popular desktop environment based on the Gnome 2 desktop, which many users revere and prefer to later versions of Gnome. Ubuntu MATE 18.10 combines the latest MATE edition with Ubuntu 18.10.

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

Additional Resources

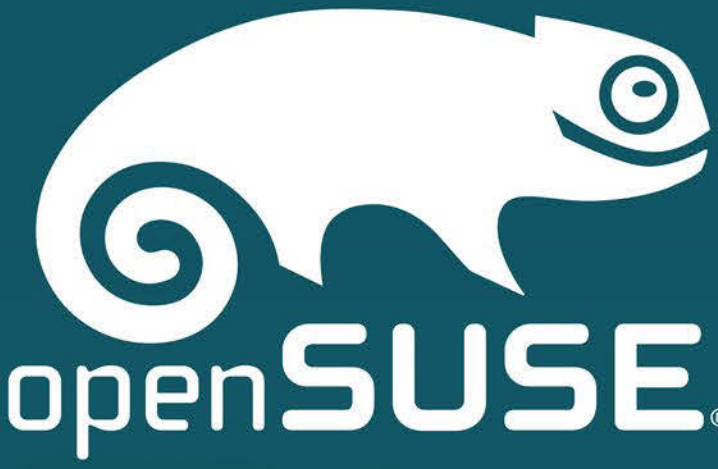
[1] Tails: *https://tails.boum.org/*

[2] Tails documentation: *https://tails.boum.org/doc/index.en.html*

[3] Ubuntu MATE: *https://ubuntu-mate.org/*

[4] Ubuntu MATE blog: *https://ubuntu-mate.org/blog/*

Nuremberg, Germany



Conference

Open Source Software

• Embedded Systems

• Cloud • Containers • More

May 24 - 26, 2019

events.opensuse.org

NEWS

Updates on tech...s

THIS MONTH'S NEWS

08
- Microsoft Gets an Open Source Web Browsers
- Canonical Launches MicroK8s

09
- A New Raspberry Pi Board
- OpenStack Foundation Changes Name of the OpenStack Summit
- Red Hat Enterprise Linux 8 Beta
- More Online

10
- System76 Announces a Line of US-Made PCs

Microsoft Gets an Open Source Web Browser

The "new" Microsoft under Satya Nadella is now going deeper with open source. The company is dropping its own technologies that power its Edge web browser, which replaced Internet Explorer. But instead of reinventing the wheel and creating their browser from scratch, Microsoft will use Google's open source Chromium browser as the base of its web browser.

Microsoft will cease to use the EdgeHTML rendering engine for its Chromium-based web browser and will use Google's Blink rendering engine.

"We will move to a Chromium-compatible web platform for Microsoft Edge on the desktop. Our intent is to align the Microsoft Edge web platform simultaneously (a) with web standards and (b) with other Chromium-based browsers," said Joe Belfiore, corporate vice president of Windows in a blog post (*https://blogs.windows.com/windowsexperience/2018/12/06/microsoft-edge-making-the-web-better-through-more-open-source-collaboration/#zeyDolmr4eDwoZzX.97*).

Microsoft is also planning to bring its Chromium-based web browser to competing platforms like macOS. "We also expect this work to enable us to bring Microsoft Edge to other platforms like macOS. Improving the web-platform experience for both end users and developers requires that the web platform and the browser be consistently available to as many devices as possible," said Belfiore.

Will it also come to Linux? Does this also mean that one day we may see Linux-powered Windows? Time will tell.

Canonical Launches MicroK8s

Canonical, the parent company of Ubuntu, has announced MicroK8s (*https://snapcraft.io/microk8s?_ga=2.230786783.1983860557.1544593799-633225961.1544593799*), a Snap package of Kubernetes that supports more than 42 flavors of Linux.

MicroK8s further simplifies Kubernetes deployment with its small disk and memory footprint. Users can deploy Kubernetes in a few seconds. It can run on the desktop, the server, an edge cloud, or an IoT device.

Snap is a self-contained app package solution created by Canonical that competes with Flatpak, which is backed by Red Hat and Fedora. Snap offers macOS and Windows-like packages with all dependencies bundled with it. A Kubernetes Snap package means any Linux distribution that supports Snap can benefit from MicroK8s

To deploy MicroK8s use the command `sudo snap install microk8s --classic`.

Canonical said in a press release that the benefits of providing MicroK8s as a Snap include automatic updates and well-defined security capabilities. Automatic updates ensure developers are always working from the latest upstream Kubernetes with binaries delivered directly from the source and configured in seconds.

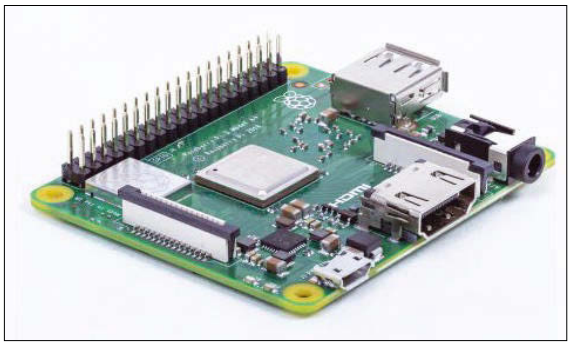
A New Raspberry Pi Board

Raspberry Pi Trading, the company behind the revolutionary Raspberry Pi platform, has announced a new board – Raspberry Pi 3 Model A+ (*https://www.raspberrypi.org/products/raspberry-pi-3-model-a-plus/*).

"You can now get the 1.4GHz clock speed, 5GHz wireless networking and improved thermals of Raspberry Pi 3B+ in a smaller form factor, and at the smaller price of $25. Meet the Raspberry Pi 3 Model A+," said Eben Upton, the founder of the Raspberry Pi project.

The board is powered by Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC at 1.4GHz; it comes with 512MB LPDDR2 SDRAM and 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN and Bluetooth 4.2/BLE.

Upton adds "By halving the RAM to 256MB, and removing the USB hub and Ethernet controller, we were able to hit a lower price point, and squeeze the product down to the size of a HAT" (*https://www.raspberrypi.org/blog/introducing-raspberry-pi-hats/*).



The new board has an extended 40-pin GPIO header, full-size HDMI, single USB 2.0 ports, CSI camera port for connecting a Raspberry Pi Camera Module, DSI display port for connecting a Raspberry Pi Touch Display, 4-pole stereo output, and composite video port, a microSD port for loading your operating system and storing data, and 5V/2.5A DC power input.

OpenStack Foundation Changes Name of the OpenStack Summit

The OpenStack Foundation has decided to change the name of the OpenStack Summit to Open Infrastructure Summit at the last OpenStack Summit in Berlin.

The name change was expected as the OpenStack Foundation was trying to position itself outside of the OpenStack world and as an organization to help manage their massive infrastructure.

During a press conference in Berlin, Mark Collier, COO of the OpenStack Foundation said, "The OpenStack Foundation brand is not a big deal, while Open Infrastructure Summit is to bring people together who may not be involved with OpenStack."

This is not the first time a major open source project has changed it name. The changing market dynamics also led the Linux Foundation to change the name of LinuxCon to Open Source Summit.

What's happening is that open source is becoming very pervasive and technologies are evolving fast, broadening their reach and scope. At times, names tied to specific projects limit its scope.

Red Hat Enterprise Linux 8 Beta

Red Hat, soon to be owned by IBM, has announced the beta version of Red Hat Enterprise Linux 8 (*https://developers.redhat.com/blog/2018/11/15/red-hat-enterprise-linux-8-beta-is-here/*). As the IT landscape is changing and the workload is moving from traditional data centers to the cloud, leveraging emerging technologies like blockchain and machine learning, the expectation from the operating system that runs these workloads is also changing.

To keep up with the changing times, Red Hat Enterprise Linux 8 maintains a fine balance between past and future.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

TUIs, a Smoke-Jumping Admin's Best Friend
Jeff Layton
Sys admins are like smokejumpers who parachute into fires, fighting them until they are out, or at least under control. When you jump into the fire, you only have the tools you brought with you.

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

Kali Linux Is the Complete Toolbox for Penetration Testing • Holger Reibold
The Kali Linux distribution is a complete toolbox for penetration testing.

Automatic Build and Deploy with OpenShift and GitLab CI • Alexandre Nuttinck
OpenShift and GitLab CI/CD can build and deploy your apps automatically, so you can stay focused on writing code.

VTP for VLAN Management • Jan Ho
Cisco's VLAN Trunking Protocol for Virtual LAN management in medium to large computer networks can make a network administrator's life easier.

"Today, we're offering a vision of a Linux foundation to power the innovations that can extend and transform business IT well into the future: Meet Red Hat Enterprise Linux 8 Beta," Red Hat said in a press release.

One of the most notable highlights of this beta is the introduction of the Application Streams concept to deliver userspace packages more simply and with greater flexibility.

"Userspace components can now update more quickly than core operating system packages and without having to wait for the next major version of the operating system," said Red Hat.

What it means is users don't have to worry about "RPM hell" or conflict of packages. "Multiple versions of the same package, for example, an interpreted language or a database, can also be made available for installation via an Application Stream," explained Red Hat.

It allows users to consume an agile and user-customized version of Red Hat Enterprise Linux without impacting the underlying stability of the platform or specific deployments.

You can test the beta by downloading it from here: *https://access.redhat.com/products/red-hat-enterprise-linux/beta*.

System76 Announces a Line of US-Made PCs

System76, one of the few vendors that sells Linux PCs, is launching a series of computers that the company says is "made in the US." Although some of the elements within the system are imported, System76 says the Thelio desktop series (*https://blog.system76.com/post/179592732883/system76-on-us-manufacturing-and-open-hardware*) goes beyond mere assembly and that the system is actually manufactured on American soil.

"We've seen it argued that this isn't US manufacturing, because every part isn't made in the US. If we sourced every part externally, this would be called "assembled in the US." That's not what we're doing here. We're transforming raw materials into a final product," System76 said in a blog post.

There are three members of the Thelio family: The entry-level Thelio comes with Ryzen or Core CPUs, up to 32GB of RAM, and is priced at $1,099. Thelio Major is powered by Threadripper or Core-X CPUs, can pack up to 128GB of memory, and has a base price of $2,299. The biggest member of the family, Thelio Massive, is powered by dual Xeon CPUs, offers up to 768GB of ECC memory and up to 86TB of storage, and is priced at $2,899.

System76 has created its own Ubuntu-based operating system that runs on their hardware, Pop!_OS. By building their own operating system, System76 has optimized the performance.

System76 has designed their own chassis controller and hard drive backplane, called Thelio Io, which moves proprietary functionality from the mainboard to the open source Thelio Io "daughterboard."

"Moving chassis and thermal control to Thelio Io enables far more granular performance optimization. Motherboard data, fan speed, and GPU and OS data are used to coordinate optimal airflow," claimed System76.

Thelio Massive also includes an open source System76 designed SAS backplane for high performance 2.5" PCIe storage.

System76 has released its own work on the hardware and software parts into open source. Thelio hardware is certified by Open Source Hardware Association (OSHWA) and licensed under the GPLv3 and CC BY-SA.

Icinga Camp Berlin 2019

Join us in Berlin

REGISTER NOW!

icinga.com

ICINGA

Zack's Kernel News

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

By Zack Brown

KUnit Testing Framework

A sizable part of kernel development is taken up with testing, debugging, and speed evaluation. Even the revision control system includes the `git bisect` operation to quickly find which patch introduced a given bug. There are whole bug classes that, if not caught before the developer submits the patch, will get a stern rebuke from Linus Torvalds.

One type of test that has recently come to the Linux kernel is unit testing. Brendan Higgins introduced KUnit for consideration to the Linux Kernel Mailing List. The idea of unit tests is that code can be tested completely independently from the rest of the system. A given subroutine is tested solely for the action it performs. This is in contrast to other forms of testing, where code is so interconnected throughout the system that it can only be tested indirectly, by examining the behavior of a full running kernel. Unlike those more complex tests, unit tests are typically simple and repeatable, test a single thing, and give an immediate result. They are also fast and able to perform many tests in just a few seconds.

The response was a lot of enthusiasm for this work, especially from developers who had been rigging up their own unit tests on a case-by-case basis. Daniel Vetter was one of these, creating unit tests for the Direct Rendering Manager (DRM) driver. As he put it, "Having proper and standardized infrastructure for kernel unit tests sounds terrific. In other words: I want." And Dan Williams had been doing a similar thing for the `libnvdimm` driver. He said he planned to convert all of his unit tests to use KUnit.

Not everyone was immediately enthusiastic. Tim Bird wanted to make sure that any generalized unit testing system would behave properly under unusual conditions. For example, if a developer were running under one architecture, but testing code intended for another architecture, how would KUnit handle that? Would it compile the code for the target system or the running system? He also asked where unit tests should live. Did they belong in the same directory as the code they were testing, potentially cluttering them up? Or should unit tests all be collected together into a special directory only for unit tests? And Tim was also interested in the possibility that unit tests might be a useful entry point for new developers. If a unit test were easier to write than the code it tested, then perhaps new developers could start off writing unit tests, and then move "up" to writing actual driver and subsystem code.

Brendan addressed some of these concerns. Among other things, he confirmed that cross-compiling unit tests was not currently supported. If someone wanted to run unit tests covering a given architecture, he said, they would have to compile and run the tests on that architecture. The KUnit code would not handle that for them.

Brendan added that KUnit tests would live most naturally in the directories with the code they tested. This

Author

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

way it would be trivial to write driver and subsystem updates, along with all their unit tests; the same people who maintained the main code for a given project would also maintain the unit tests. He added that he thought this would actually reduce a maintainer's workload, because unit tests would result in cleaner, more maintainable code throughout a given project.

However, Brendan said he did not believe that unit tests would be easier to write than driver and subsystem code itself. He said, "the person who writes the test must understand what the code they are testing is supposed to do. To some extent that will probably require someone with some expertise to ensure that the test makes sense."

There was not much controversy over Brendan's KUnit patches. It seems as though most developers consider them an idea whose time has come; they'll soon take their place in the growing pantheon of testing and profiling code that supports the Linux kernel.

Removing the Ancient 00-INDEX Files

Sometimes kernel infrastructure can hang around for a long time after it's no longer needed. Documentation is especially susceptible to this, since out-of-date documentation doesn't actually break anything; it just makes it more difficult and annoying for humans to understand.

Originally, Linux used files named `00-INDEX` in a given directory to list that directory's files and give a brief description of their purpose. Nowadays, no one really relies on those files anymore, and kernel documentation has shifted quite far from those early days.

Henrik Austad recently decided to purge the `00-INDEX` files from the kernel. By now they are well-known to be very out-of-date. He wrote a script to test how out-of-date they were and found hundreds of instances. In light of that, he posted a patch to remove them from the source tree en masse.

Joe Perches liked this idea, pointing out that the kernel now used reStructuredText (`.rst`) files to document itself. He felt there was no need whatsoever to hold onto those old `00-INDEX` files.

Jonathan Corbet suggested sharing the patch around a little bit to see if there was any pushback, since there were still people who sent him patches to update them; there was actually useful information in them.

In fact, Henrik said he'd be happy to update all the `00-INDEX` files if that was the thing to do, but if there was no need, he said, he'd rather just ditch them.

Josh Triplett saw no need to keep any `00-INDEX` files. At most, he thought it might be beneficial to copy their useful information into the `.rst` files currently used for documentation, but even that didn't seem so important to him.

Paul Moore also supported removing the `00-INDEX` files. He saw no point in keeping them.

Overall, it does seem as though the `00-INDEX` files no longer serve a useful purpose. For a long time, the kernel had no native documentation, and the `00-INDEX` files provided an absolute minimum of information – just the names of each source file and a brief description. Since then, kernel documentation has blossomed, and there probably is no need for such primitive methods.

Identifying Process Termination Without Polling

Sometimes no one disputes the value of a new feature, but no one can agree on the right way to implement it. Often there are simply different sensibilities about what should be sacrificed in favor of what and which feature elements are the most important.

Recently Daniel Colascione posted some patches to help the Android OS identify when a given process had terminated. Of course, it wasn't only for Android; it was a general-purpose feature, but his motivation in writing the patch was to solve a particular problem for Android.

Typically processes only care about the exit status of their own children, and there are features already in the kernel for that. It's less usual for any process to care about the status of some random process elsewhere on the system. But as Daniel pointed out, Android ran a process called `lmkd`, which would kill processes that threatened to use too much RAM. For this, it needed to give the `kill` command and then check to see that the process had exited properly. Currently, `lmkd` would just keep checking over and over and over, polling on the process until the process ID disappeared. Daniel wanted to put a more efficient system in place. He wanted `lmkd` to block on a read request and simply wait until that request returned. When it did, that would mean the process had terminated. Much simpler and less resource-intensive.

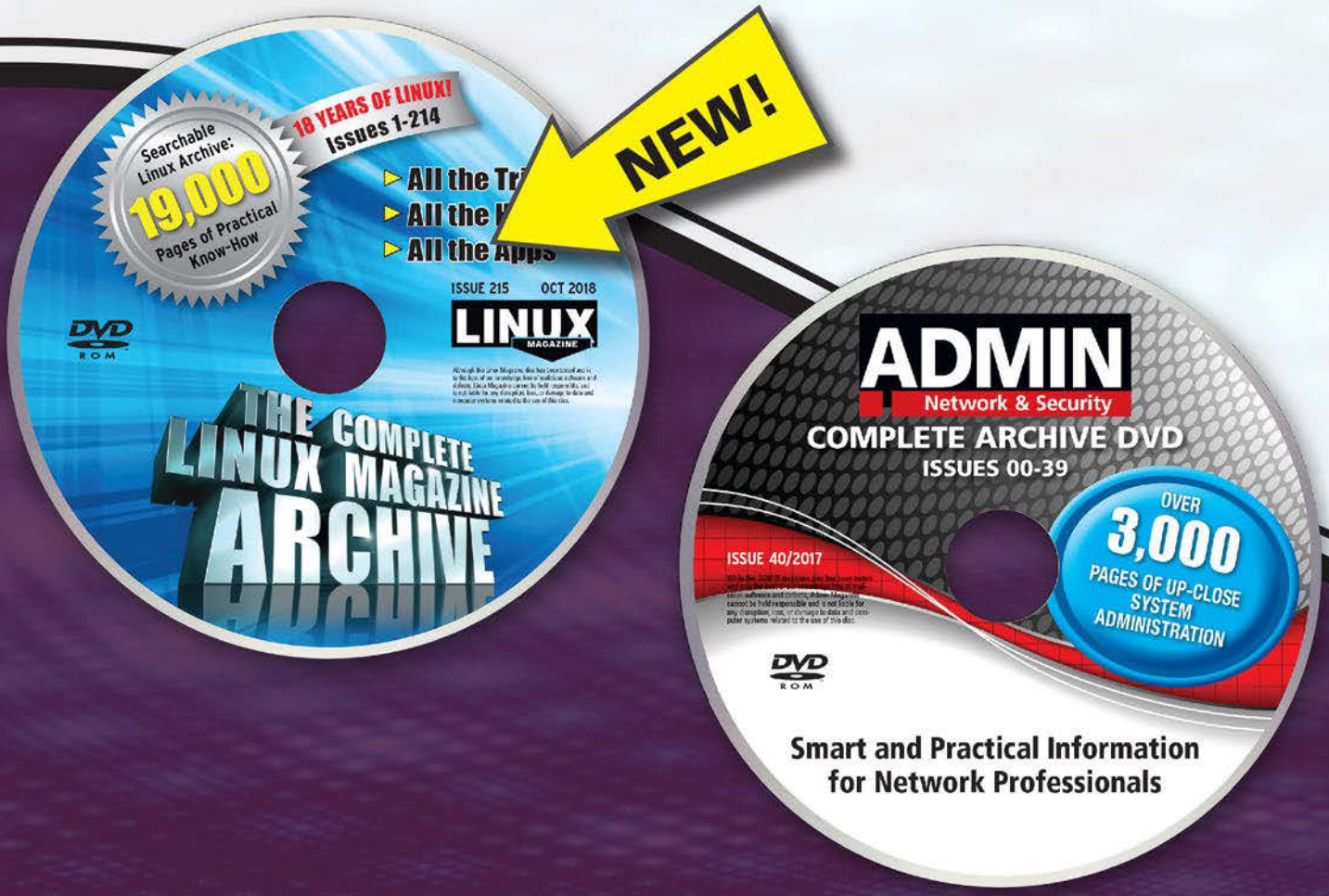
This was where controversy erupted.

Joel Fernandes felt that polling on the process ID was plenty good enough. The reason was that Daniel's idea involved creating new files under the `/proc` directory, which would increase the overall infrastructure of the kernel. Avoiding that seemed like an important thing to Joel.

Joel's alternative, however, involved tracking the target process with `ptrace()`, which Daniel felt would be odd, given that `ptrace()` was supposed to be more of a debugging tool and not really a general-purpose tool for a running system. Using it in this case, Daniel said, would even interfere with debuggers and core dumps.

In general, all the alternatives proposed instead of Daniel's patch were more complex and heavyweight than Daniel's code. But the developers advocating those alternatives liked them, because they didn't involve creating new files in the `/proc` directory. Daniel, meanwhile, felt very strongly that, "given that we *can*, cheaply, provide a clean and consistent API to userspace, why would we instead want to inflict some exotic and hard-to-use interface on userspace instead?"

There was not really any resolution to the controversy during the conversation. Sometimes, it's not clear what the most important issues really are in a given situation. We clearly don't want the `/proc` directory to grow without bound, but does Daniel's feature really represent that kind of bloat? We clearly would prefer to use existing mechanisms to solve problems rather than create new infrastructure for them, but do the alternatives to Daniel's code really represent the simplest solution? In all likelihood, the debate will continue, drawing in more prominent developers, until a clear technical requirement is identified, showing that one approach is truly better than the other. ■■■



Complete Your Open Source Library with Archive DVDs!

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

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

Order Your DVD Now!

shop.linuxnewmedia.com

Linus Torvalds and Greg Kroah-Hartman discuss the state of Linux

Kernel View

Linux creator Linus Torvalds describes some recent challenges and opportunities for the Linux kernel – and stable branch maintainer Greg Kroah-Hartman joins in with thoughts on diversity and competition. *By Swapnil Bhartiya*

In what's becoming an annual event, I sat down with Linus Torvalds at the Open Source Summit, Vancouver (Canada) to talk about the state of Linux. We chatted for over an hour and Greg Kroah-Hartman (maintainer of the stable branch of the Linux Kernel) also joined us. We talked about a wide range of topics, including the sustainability of Linux, threats to Linux (both from inside and outside), the state of security in the kernel world, the success of Linux on the desktop, his outbursts on the Linux kernel mailing list, privacy, diversity, and much more.

Following is an edited version of my interview with Linus, along with a special appearance from Greg Kroah-Hartman.

Swapnil Bhartiya: It's been almost 27 years since Linux came to life. How are things in the Linux world?

Linus Torvalds: It's been normal. Our model of doing development really hasn't changed for over 10 years. It's standardized, and we continue to make releases about every two to three months. One of the big changes this year has been dealing with all the hardware bugs (Spectre and Meltdown [1]). We had to spend a fair amount of efforts for workarounds, and that's been somewhat stressful, but I'm hoping that we're starting to see the tail end of that.

SB: Are hardware bugs more problematic than software bugs?

LT: Hardware bugs have been pretty problematic for the kernel community. It's not because the workarounds have to be in the kernel – it's also because of the secrecy around them, which means that our normal workflow doesn't work.

It's not just about bringing everybody in and talking about it on the mailing list; it's about not being able to use the open infrastructure that we have for testing and validation. Whenever we try to do development without all that infrastructure, things become very painful. So, it has really been a big issue for the kernel community.

SB: But there have been software bugs, too.

LT: Yes, but that's nothing new. We have always had software bugs [2]. What may have changed is that, over the last few years, people have gotten way more vocal about security issues. It's not necessarily because they are now more serious about them; it could be because there are more processes in place for them. But we have always had software bugs. Hardware bugs are also not new, but they are different for us in the way they affect our workflow.

SB: Linux played a critical role in establishing open source as a successful software development model. Do you see any risk that companies who have opened up their code might go back to a proprietary model to protect their investment?

LT: From a development angle, I think people do realize how powerful it is to have open code and let people be part of the development process. So the only situation where, I think, people might go back to the old-fashioned proprietary model is for niche products. These could be those cases where you actually have a hard time finding that general population that wants to help you with code. And I don't think those cases matter that much.

I also think that if you worry about just open source in general, you should worry more about the whole experience. When you do everything on the web, and as a service, you can use all those services with open source applications in a regular Chromium browser, but you don't see what's going [on] behind the curtain. I think that's what a

lot of people are worried about. You can always walk into a situation where the software you run on your machine may be open source, but what you use on that machine ends up running in a proprietary cloud system anyway.

SB: There is a genuine reason to worry. My desktop is free, but my data and applications (which run in the cloud) are not. I am curious, personally, what do you care about more – just that the code has to be open or also about the business practices as well?

LT: Personally, I only care about the code. When I say maybe there are people who worry about walled gardens and cloud providers who take ownership of your data, I am not one of those people. That's not what I actually care about. That's not what I do. What I do is code. What I care about is code. Even in some of those walled gardens and cloud services, they are running open source code. What these companies have is all the data on their users, which is obviously their bread and butter. Maybe it is a real issue for many people. I do understand that a lot of people care about open source because of the whole traditional "freedom" thing, but they will find it much more worrying when they know how much computing is happening inside of walled gardens.

SB: You come from Europe, and they have very strict laws to protect privacy, which we don't see here in the US. Don't you care about your privacy?

LT: I don't worry too much about my own privacy; I'm a very public person. I do all my work in public. I have very little that I care about.

At the same time, I think it's inevitable. There's going to be a lot of data. People are going to know a lot about you. People who worry about your DNA being available in various databases are right; it's going to be available in a lot of databases. Even if you are careful, the fact that you have family members who were not that careful means there will be a lot of data about you out there. I don't necessarily think that's a problem, but it could be problematic in certain circumstances.

I think at some point, the US will wake up to realize the problem and make strict data privacy rules. There will be some kind of protection, but let's face it, data is cheap and it's not going away. Whoever thinks that it will go away is deluding themselves.

SB: While we are talking about privacy and looking at the ruckus your emails create on LKML, do you wish there was a private channel where top maintainers could discuss things candidly without prying eyes of bloggers who are looking for sensational stories?

LT: Honestly, no. We do have private mailing lists for security issues. We also email each other or a group of people to discuss things. Usually, these private discussions are not about code; they are about things like maintainership issues.

But whenever such private conversations happen, I find them to be problematic, because then you don't have any archive that you can look back at. When you bring in new people, they miss the whole context. So, I am much happier having all the discussions in the open. It doesn't always work that way; it does mean that then when people get heated, everything they say becomes public, but I think that's healthier than closed room discussions.

SB: Has there ever been any moment where you felt burnt out and said I will go and do something else?

LT: I've never been burned out in that sense. Things do get stressful sometimes. The last merge window was fairly painful. There were two weeks of not just the regular work of a merge window, which is my busiest time anyway, but there were all other security related issues going on.

When I finally closed the window I said, "okay, I'm going to take a breather." But it's not like burnout. It's more like, "Okay, it's good. I have a calm week coming up. I have some travel. I will just step back for a couple of days and then I'm fine again."

SB: But you don't travel much.

LT: No. It was just this time. I don't travel very much for conferences. But my point is even normally I can just say, "okay, I'll do something else for three days," and that's fine. It's not where I feel this project is not worth it. It's more like, wow, okay, now I just need to have an extended weekend to take a break from all that stress.

Honestly, it's not that common. Most of the time, I am not stressed. We have a very efficient workflow. Even when I'm busy, it's seldom really stressful.

SB: Okay, and we all want you to be around forever. That said, what have you done to ensure sustainability of Linux when we all are gone, given that you have set high standards when it comes to code quality?

LT: We have a very big and a great community. When it comes to open source, the kernel is not just unusual: It's a complete outlier. As far as I know, there are no other open source projects that literally have a thousand people participating in every release. We have hundreds of maintainers and tens of pretty high-level maintainers. We have both depth and breadth in our development community. We have a very efficient workflow.

(Note: The interview was conducted before Linus took a break in the aftermath of the events surrounding the new Linux code of conduct. Greg Kroah-Hartman stepped in and everything moved forward normally, which implies that the

Linux kernel does indeed have a very efficient workflow that would ensure the longevity of the project. Coincidentally, that was about the time when Greg Kroah-Hartman joined us).

SB: We often hear that the kernel community has a mono culture; that it is not as diverse as the rest of our world? Does that worry you?

Linus looks at Greg.

Greg Kroah-Hartman: There's two things here. One nice thing about Linux is that we've been doing this for so long that we all enjoy the experience and we don't go away. People may change companies and jobs, but they continue to work on the kernel. So we have the same crowd of mostly white males who got involved with the project at the early stage. At the same time, we are getting a lot of new people.

It's a big community, and we don't know everyone. I don't know the person whose patches I took. I met someone here who said that "you took my patch when I was 12 years old." In most cases, we don't know every committer; we don't know their age or their background. In itself, that's a good thing too. Anyone can submit a patch, regardless of who they are. But, we agree that diversity is important. It's really important to have people who bring different perspectives, different approaches to solving a problem. We are aware of the problem, and many people are working on it. We know we have a long way to go.

We participate in the Outreachy program [3], two times a year. Outreachy is really good because most of the people that come through Outreachy easily get jobs. A lot of kernel developers at ARM came through Outreachy. We participate in the Google Summer of Code, and most participants are from Asia or India. So, we are trying to do our best.

SB: But you can do only so much; you are not a company where you go and hire people.

LT: True. It's very easy to talk about diversity, but at the same time, it's really hard to find people. There are cultural and historical reasons. It's social. It's about who has access to computers. I think girls are discouraged from getting into the whole math and hard technical stuff, so there's a very small percentage coming in.

GKH: It's also up to companies. They are the ones that assign people to work on the kernel. I was at IBM, and they have a large number of women at IBM who work on the kernel. They are participating, but sometimes we just don't realize it.

SB: Is there anything that still worries you and keeps you awake at night?

LT: The technology doesn't worry me. I do worry about the community and maintainership issues. There are certain people that I worry about burning out. These people are very central to the code. I am concerned about David Miller [4], who looks after the networking code. Sometimes I get the feeling that he may not be burning out, but he does get frustrated. I am also concerned about Greg as the last merge window was very stressful for all of us.

So there are areas in the kernel where I worry about maintainership, but on the whole, we've never actually had a big problem. We never really had huge maintainership issues, even though it's actually what worries me most.

The real problems we've had have been when we needed to change the flow of patches or when we changed from BitKeeper to Git. That was really very painful.

GKH: I have talked to a bunch of maintainers here about how to do that job better. There are maintainers like Dan Williams [5] who are working on a document to help other maintainers.

I talked to someone who was a co-maintainer, and he says that's the best thing ever. Now he can go away for a week and lean on the other person. Working together really helps.

SB: I was talking to Jonathan Corbet, the editor of LWN.net, and he said that what he worries about sometimes are the new projects like Zephyr and Fuchsia [6], which come from companies that don't understand the importance of the GPL license.

LT: Companies do what companies do. If you don't like the GPL and choose another license, if you need to create a competitor to the kernel, go ahead do it. We can compete. We like competition.

SB: It's good to hear that you want competition, because Linux, in a positive way, has a monopoly: Everyone is using it (laughter).

GKH: It's not a monopoly, because it's free. Competition is good. Just look at the GCC and Clang communities [7]. They are both happy. It sped up their development process. They are now bouncing ideas at each other, and everyone benefits.

Zephyr is a different case, as it tried to solve a problem where there wasn't any good solution. But then if you look at what Microsoft is doing with Linux and Azure Sphere OS, they have slimmed down Linux even tinier for some ARM system. We are really getting close to that edge. Maybe we do need to run Linux on these tiny devices.

LT: I don't follow those projects, but I have an opinion that most projects fail. It's true not just of projects; it's also true of branching, too. I don't get too upset when somebody makes a branch and says, I will rewrite something. I am

fine with that. Go ahead and prove us wrong. The problem with "tinyfication" efforts – and there have been multiple – is that they usually end up doing a hatchet job. They just cut off pieces in ways that are not back mergeable. They go off for a few years, and they cut off all the pieces they don't care about. Then we are left with a situation where we would like to be more modular, and we'd like to be better at supporting a smaller model, but the way they did it makes it hard to merge it back. They chopped things without taking into account other cases.

GKH: But there have also been cases where the branched project was so successful that communities worked together to make it mergeable, but it takes a lot of time and efforts.

SB: Let's switch the gear from serious topics to a lighter one. Linux has conquered the world, but the desktop still remains an unrequited dream. I am curious if the Linux desktop really matters anymore, now that most of our workload has moved to the cloud?

LT: I think it still matters. Not everybody uses the cloud; there are a lot of people who still use the desktop. I still wish we were better at having a standardized desktop that goes across all the distributions. It's not a kernel issue, but it's more of a personal annoyance to see how the fragmentation of the different vendors have held the desktop back a bit. I do see some progress with things like Flatpack.

SB: True, but there is a new kind of fragmentation now – there is AppImage, Flatpack, and Snap (package tools). There is no single platform that developers can target?

LT: Yes (shakes his head in disappointment). And maybe that's what Chromebook ends up doing – turning into a de facto standard for desktop applications. Once Chromebooks start running Debian packages, we will see. I am still optimistic after 25 years. It's going to be another few years, but the Linux desktop is not there yet.

Actually, Chromebooks are getting pretty good. The Crouton tool has been around for a while now, and it lets you run Linux distributions on a Chromebook. Now they have official support for running Debian apps. I have not actually used it yet, because I only use Chromebook for calendaring.

SB: Why you don't use Chromebook for your work?

LT: I would actually not mind having a Chromebook, but right now my main problem is even when you can run native Linux on Chromebooks with Crouton and official support for Debian, you can't do the kernel testing. That's what I care about. Chromebooks have reached the point where I could see myself using a Chromebook in a few years, but it is not there yet.

But, in general it seems that Chromebooks and Android are the paths towards Linux on desktop. ■■■

Info

[1] Meltdown: *https://meltdownattack.com/*

[2] "Linus Torvalds: Security Will Never Be Perfect": *https://www.cio.com/article/2973995/linux/linus-torvalds-security-is-never-going-to-be-perfect.html*

[3] Outreachy: *https://www.outreachy.org/*

[4] David S. Miller: *https://en.wikipedia.org/wiki/David_S._Miller*

[5] Dan Williams: *https://github.com/djbw*

[6] Jonathan Corbet interview: *https://youtu.be/VOxHKyglvLA?t=1018*

[7] Clang vs. other open source compilers: *https://clang.llvm.org/comparison.html*

Design your own boot menu with Grub Customizer

Paint by Numbers

Is the simple black and white GRUB menu causing confusion and obscuring important choices? Why not customize with GRUB themes and the Grub Customizer? *By Anzela Minosi*

Many Linux distributions keep the boot menu so simple that it hardly differs from the BIOS messages. If you settle for the default boot menu, you might be missing the chance to provide some visual enhancement to the user experience. The need for visual clarity is often more than cosmetic. The boot menu sometimes contains real choices, including alternative boot options and operating system versions, and some attention to design can make those choices much easier to see and understand.

With just a few settings, you can spice up a drab and confusing boot menu. But be aware that incorrect GRUB settings can quickly cause the boot process to fail. A tool with a graphical user interface can help ensure that editing the GRUB files is a safe experience. On the other hand, editing the files manually is a time-honored practice if you know what you are doing, and it might be your only option: Some Linux distributions don't provide a graphical tool.

First Step: Backup

But before you get started, you will want to back up the boot partition or boot directory: A faulty GRUB configuration can quickly lead to entries disappearing from the boot menu or a system unable to boot. Use the commands in Listing 1 to create the necessary backup copies.

GRUB Themes

On Gnome-Look.org, you will find ready-made GRUB themes with matching font types and sizes, background images, and colors [1]. Download your choice of theme and unpack it; then place it in the GRUB theme directory. For some distributions, the theme directory is located in `/boot/grub2/themes/`; others prefer the path `/boot/grub/themes/`.

Listing 1: Backup!

```
# dd if=Partition of=/Path/filename.img status=progress; sync
# rsync -vuar /source_directory/ /target_directory/
```

Listing 2: Testing the Theme

```
&lt;font color="#ffff00"&gt;-=https://github.com/hartwork/
   grub2-theme-preview/archive/master.zip=- proudly presents
&lt;font color="#ffff00"&gt;-===- proudly presents
$ cd /directory_name
$ sudo make install
$ grub2-theme-preview /path/grub-themes/theme_name
  --grub2-mkrescue command_sequence
```

Listing 3: Specifying a Grub Theme

```
GRUB_THEME="/path/Grub-Themes/theme_name/theme.txt"
```

Listing 4: Updating the Grub Configuration

```
# grub-mkconfig -o /path/grub/grub.cfg
```

If a theme directory does not exist, create it with the `mkdir` command.

To make sure that the theme works, first test it with the `grub2-theme-preview` tool [2]. Many distributions do not have this tool in their repositories, so you have to download and compile it yourself. Then emulate booting with the last command from Listing 2 (Figure 1).

You only need the `--grub2-mkrescue` option if the command is different in the distribution you are using. Enter the appropriate command sequence after the `--grub2-mkrescue` option. If you don't know the command, open a terminal and type `grub` lowercase. Then press the Tab key and search the Bash suggestions for the appropriate command.

After testing the theme, open the `/etc/default/grub` configuration file as root and enter the path to the theme (Listing 3). Check in advance whether a file named `theme.txt` exists in the theme directory. The `theme.txt` file contains the configuration for the theme.

In order for the changes to take effect the next time the computer is restarted, you need to update the configuration file `grub.cfg`. Call the command in Listing 4 as root.

GRUB searches the boot directory for kernel images and writes the names of the filenames' image files to the GRUB configuration file. GRUB accesses this list during the boot process to generate a list of the operating system choices.

Grub Customizer

Some distributions, including Fedora and Ubuntu, have Grub Customizer [3] in their repositories. Grub Customizer lets you include or change themes in a graphical user interface, adjusting settings that alter the appearance of the boot menu and making further changes to the configuration.

To customize the appearance of the boot menu, first launch Grub Customizer. On Fedora, look in the menu under *System* | *System Administration* | *Grub Customizer*. You can also enter the `grub-customizer` command from a root console.

In the *List configuration* tab, you can change the boot order of the operating systems found by GRUB by right-clicking an entry and selecting the up or down arrows to move the entry to a new location.

Under *General Settings* you can define the timeout (i.e., the time in seconds that GRUB waits until it continues booting the default entry without user interaction). The standard entry itself can also be modified (Figure 2). If necessary, uncheck *show menu* to completely hide the boot menu, so that users no longer have the option to select an operating system when starting GRUB.

Finally, you can include a theme in the *Appearance settings* tab by clicking on the plus symbol top right and selecting the path to the theme's archive. You can then edit the theme by selecting

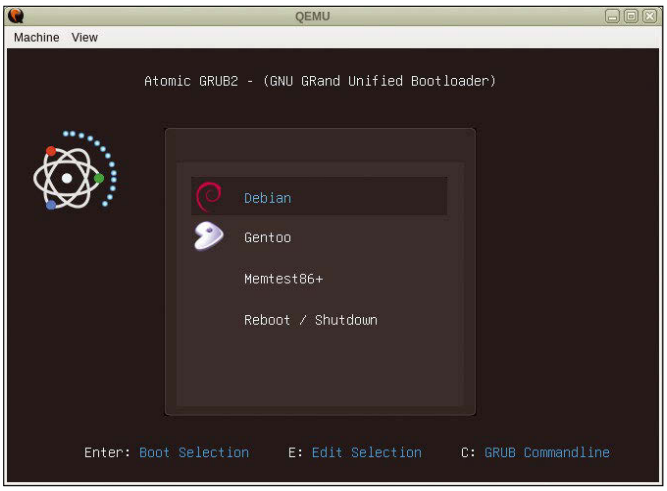


Figure 1: The `grub2-theme-preview` tool checks the downloaded theme for errors by emulating GRUB startup and displaying GRUB error messages.

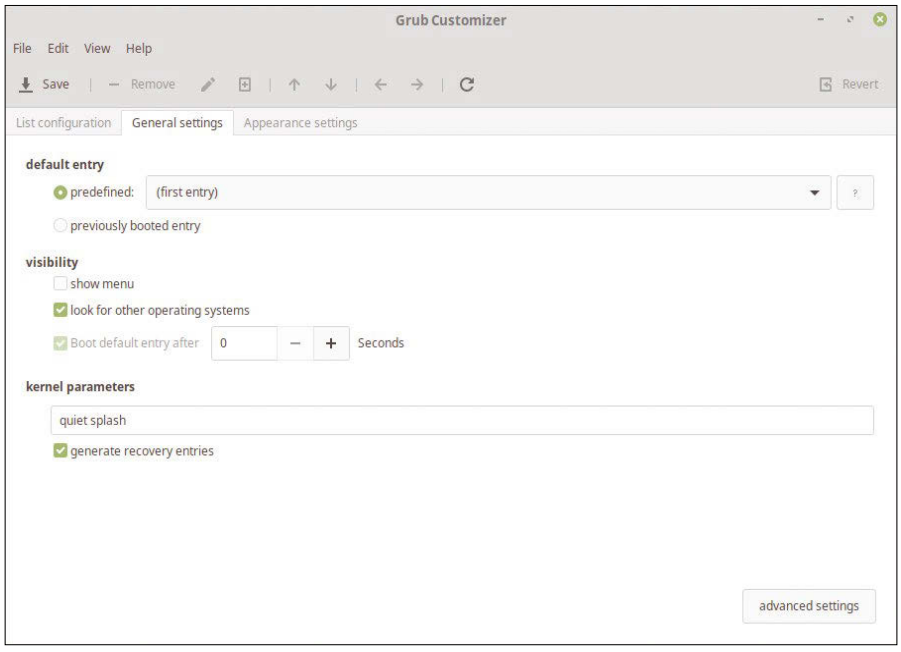


Figure 2: In the *General Settings* tab of Grub Customizer, you can define the default operating system and change other settings related to the boot process.

`theme.txt` in the file list on the left (Figure 3). When the changes are complete, click *Save*, which internally triggers the `grub-mkconfig` command.

Troubleshooting

When you add a new theme using Grub Customizer, the software sometimes fails to comment out previous settings in the configuration menu. This can cause the original settings to collide with the settings defined through the theme. To disable the settings you don't need, open the GRUB configuration file as root using the `nano /etc/default/grub` command, and comment out the unnecessary lines, as shown in Listing 5.

Alternatively, you can edit this file directly in Grub Customizer by clicking on *advanced settings* bottom right in the *Appearance settings* tab. In the dialog that follows, make your changes by setting or removing the check marks (Figure 4). You can also add your own GRUB parameters. A detailed explanation of the options is available online [4].

On Fedora, the operating system may not accept new themes – in this case, only manual adjustments in the *Appearance settings* tab will help (Figure 5). If GRUB does not load the defined background image at startup, the problem might be that the image has a color depth greater than 8-bit. If necessary, check the color depth using the `file` command:

```
file image_filename.png
```

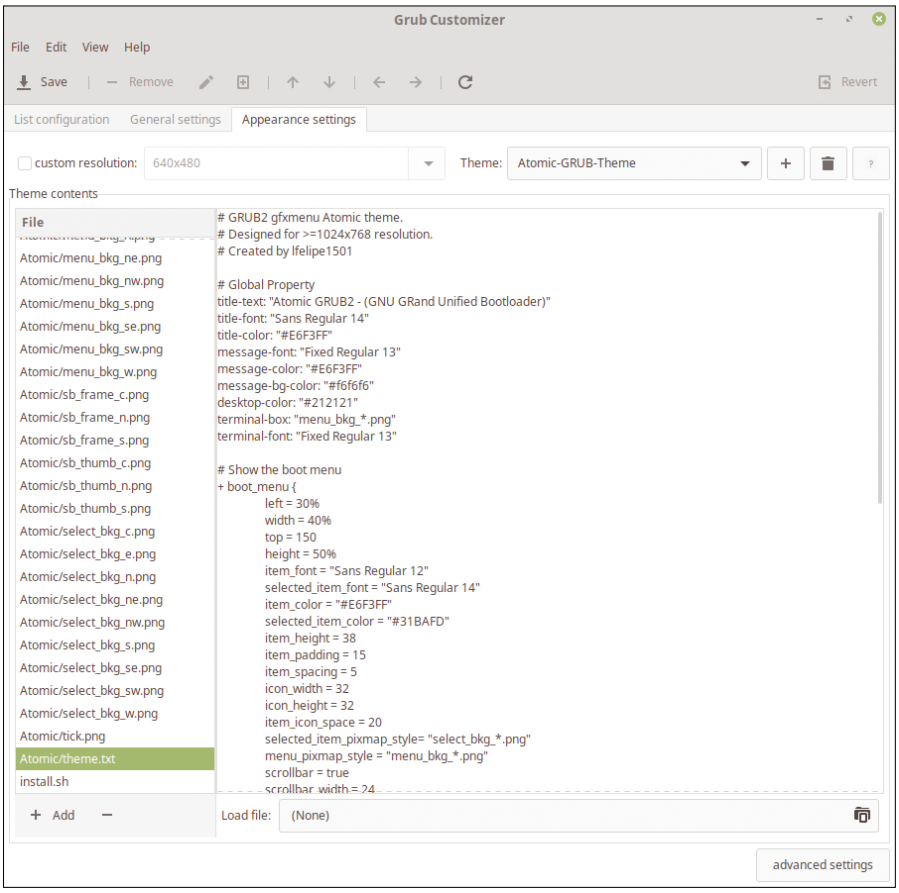


Figure 3: GRUB expects a `theme.txt` file with the configuration for each included theme.

Listing 5: Comment Out Unnecessary Settings

```
# export GRUB_COLOR_NORMAL="..."
# export GRUB_COLOR_HIGHLIGHT="..."
# GRUB_BACKGROUND="..."
# GRUB_FONT="..."
```

Conclusions

Customizing the GRUB boot menu can add clarity to the user experience, especially for computers with complex boot configurations. Grub Customizer saves you a huge amount of manual work – if your distribution supports it. But even if you have to adjust the settings manually, the result is often well worth the effort. ■■■

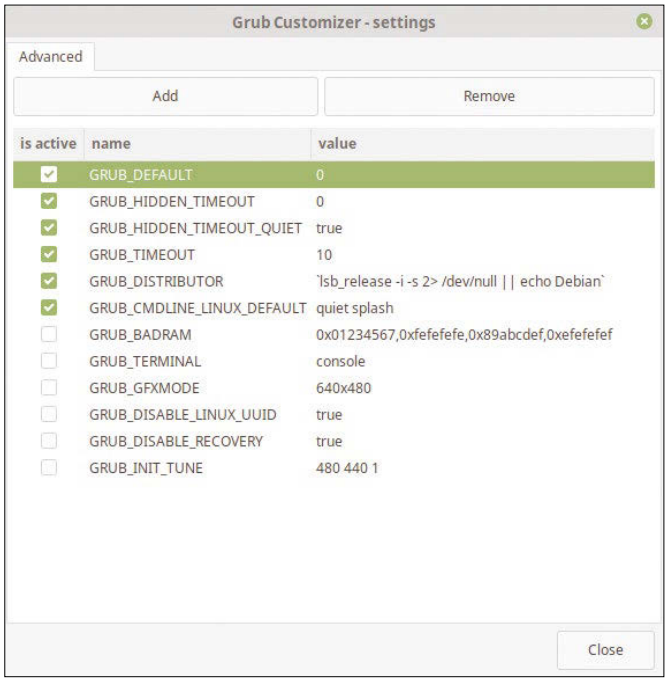


Figure 4: You can enable or disable all GRUB parameters in Grub Customizer's advanced settings.

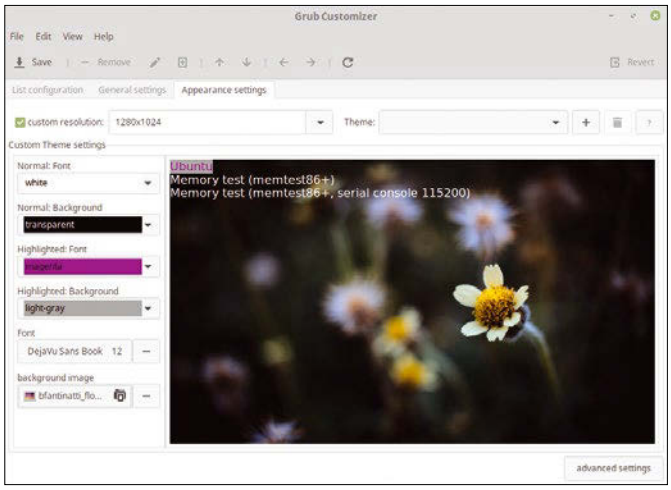


Figure 5: In addition to supporting themes, Grub Customizer also lets you customize the boot menu.

Info

[1] GRUB themes: *https://www.gnome-look.org/browse/cat/109/ord/latest*

[2] Theme preview: *https://github.com/hartwork/grub2-theme-preview*

[3] Grub Customizer: *https://grub-customizer.en.uptodown.com/ubuntu*

[4] GRUB Manual: *https://www.gnu.org/software/grub/manual/grub/grub.html*

Exploring Ubuntu 18.10 "Cosmic Cuttlefish"

Back on the Block

Ubuntu Linux gets back to basics with the Ubuntu 18.10 release – an appealing and practical distro that isn't worried about conquering the world. *By Swapnil Bhartiya*

Ubuntu is back. The same Ubuntu that I loved back in 2011 before Unity and Gnome 3 happened. Both were great projects, but they broke my workflow, so I moved to openSUSE and Arch Linux with the Plasma desktop.

Much water has flowed under the bridge since then. Canonical's dream of taking over Microsoft (Windows), Google (Android), and Apple (iOS) didn't materialize, and they decided to reduce their focus on the consumer space.

What was supposed to be bad news for Canonical turned out to be good news for open source communities, because Canonical shut down its in-house projects and returned those projects upstream. The controversial Unity desktop went away, and Gnome resumed the throne of being the default desktop environment and shell for the world's most popular Linux distribution.

Canonical's shift of focus towards enterprise doesn't mean they don't care about desktop anymore. Even if Canonical has shut down consumer-centric projects, Ubuntu desktop remains critical to the company. Ubuntu is a dominant player in the public cloud, OpenStack private cloud, and on Internet of Things (IoT) devices. Developers and sys admins of all those enterprise customers who are running Ubuntu in the cloud, IoT, and Tesla cars, need a desktop to work from, and Ubuntu client workstations are an important part of Canonical's contracts with high-volume enterprise customers.

This mindshare leads to better support for Ubuntu by both hardware and software vendors. It's simple economics: Vendors target the platform that has most users, and this means average Linux users will have better luck finding the apps they need on Ubuntu than on other distributions.

"The desktop remains hugely important for numerous reasons for Canonical, and we still see a strong appetite and usage from a variety of audiences, including home users, enterprises and developers," said Will Cooke, engineering director for the desktop at Canonical. "In the PC market, we continue our positive relationships with the major vendors, including Dell and Lenovo, who ship workstations with Ubuntu preinstalled."

I am basing this story on a Dell XPS 13 Developer Edition that came with Ubuntu preinstalled. What more evidence do you need?

Ubuntu: Just Works

Dell XPS 13 (2018) came with Ubuntu 18.04. Instead of attempting an update, I wanted to see if the stock Ubuntu would work on it. So, I downloaded Ubuntu 18.10 and did a fresh install. Everything worked as expected.

When it comes to Linux, Ubuntu is one of those distros that offers a "just works" experience. I tried 18.10 on three different machines, and every one worked as expected. The only exception was full support for a touch screen. There is a fix, but since there aren't any apps on Linux that are optimized for touch screen, I didn't bother fixing it.

One thing I did notice was that Ubuntu 18.10 booted much faster than any other distro, thanks to new compression technology. It's very responsive, and I have had virtually no problem with it so far on conventional hardware systems.

A Tale of Two Releases

Ubuntu has two types of releases: Ubuntu interim releases and Ubuntu Long Term Support (LTS) releases.

Lead Image Photo by Kris D'souza on Unsplash

There is a release of Ubuntu every six months: in October and in April. All the even-numbered releases (10.04, 12.04, 14.01, 16.04, 18.04) made in April are LTS releases. The LTS version is supported for five years, plus Extended Security Maintenance for another three years.

LTS releases focus on stability, and interim releases introduce new features that are tested to get them ready for LTS, which also means those who like to see what is new in Ubuntu should stay with regular releases.

Ubuntu has made it extremely easy to upgrade from one release to the next without breaking anything. Although I stick to LTS on my server, I always run the latest regular version on my laptops. Jumping from one point release to the next is very easy.

Open the Software Updater and click on the *Settings* button (Figure 1).

Then go to the tab that says, *Updates*. In the drop-down menu labeled *Notify me of a new Ubuntu version*, select *For any new version* (Figure 2). Click *Close*, which will take you back to the Update Manager. If any updates are available, you'll have a chance to install those updates.

If the Update Manager doesn't show any updates, you can force an upgrade for your system. Close the Update Manager, and then run the following command in the Terminal:

```
update-manager -d
```

Just follow the instructions to upgrade your Ubuntu to the latest version.

A First Look at Ubuntu 18.10

Despite their so-called withdrawal from the consumer space, Canonical has done more work to make Ubuntu an appealing platform for desktop users.

Ubuntu 18.10 came with a fine-tuned and customized (not forked) Gnome using extensions, to make it more appealing to those who have a good aesthetic sense (Figure 3). A desktop user will notice a refreshing look and feel. It looks modern and elegant. Ubuntu 18.10 is using a theme created by the Yaru community. I like it.

I don't care about the wallpaper; the first new Ubuntu feature that bloggers talk about is always the new wallpaper, but if you have time to notice the wallpaper on your PC, you are using it for all the wrong reasons. However, I do care about icons, window decorations, and fonts – for the same reason I care about an aesthetically pleasing environment in real life. Because I loved the new theme and Ubuntu fonts, I didn't need to stop for further customization (something I have to do with virtually every Linux system out there, except for Zorin OS and elementary OS).

Figure 1: Click on the *Settings* button in the Software Updater to make changes to which updates are installed on your system.

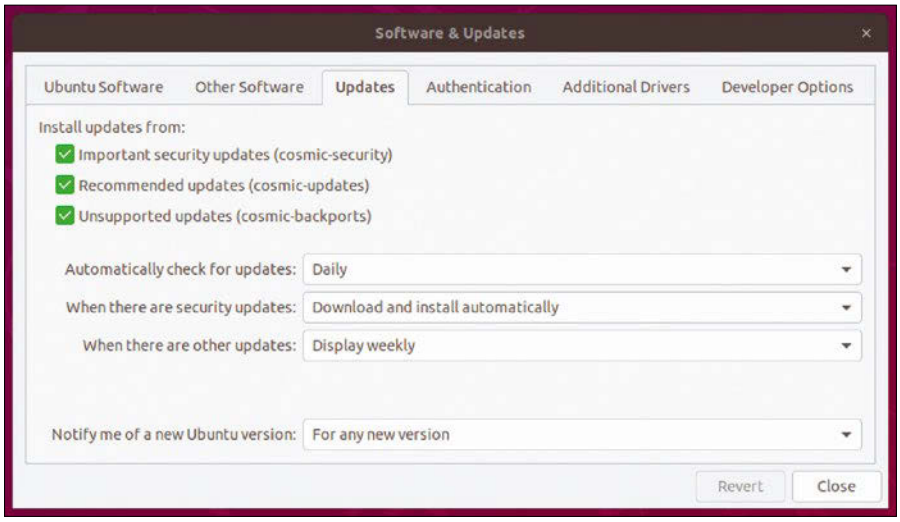


Figure 2: Choose what kind of upgrade you want for your Ubuntu.

It's All About Apps

Ubuntu 18.10 reflects the work Canonical is doing to lower the bar for app developers to attract mainstream developers towards Ubuntu. Canonical has been working on their app package – and on a delivery mechanism called Snap. Snap enables developers to offer pre-built

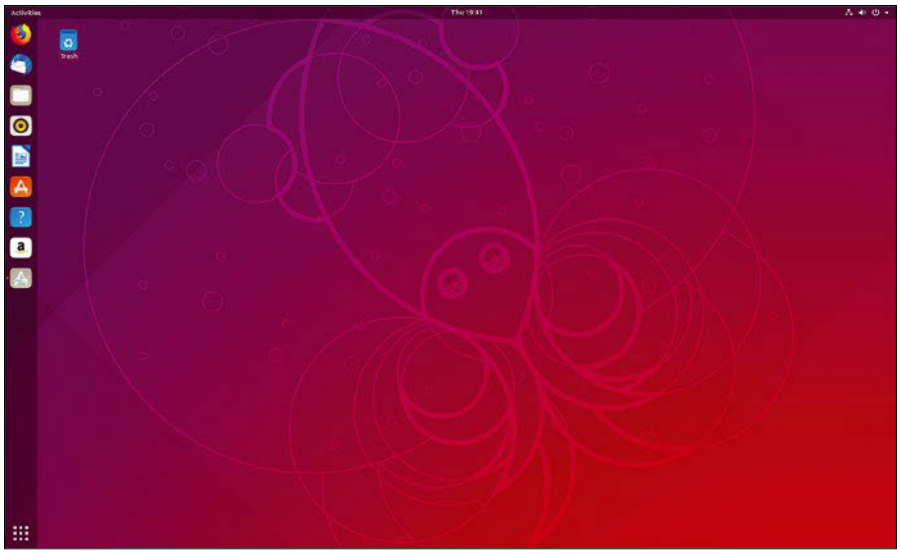


Figure 3: A pleasant default desktop.

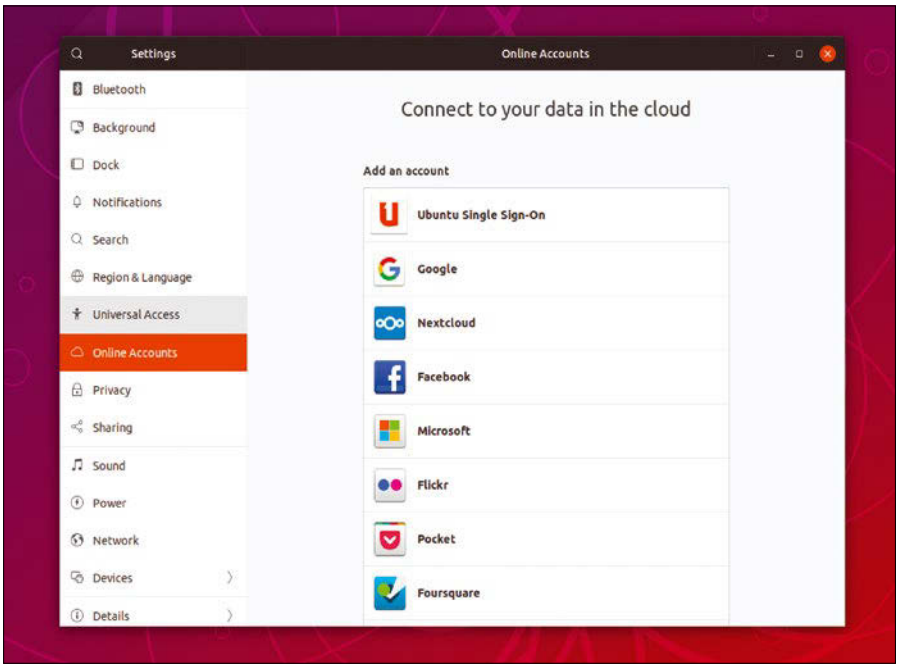


Figure 4: Configure the online accounts that you want to integrate with Gnome.

packages of their applications, removing the dependency on system-wide frameworks and libraries that turned distros into gatekeepers.

With Snaps, developers can package their apps, and users can download and install them easily. Since there is no dependency on the system, developers can use the latest libraries to offer the latest features.

"We have deepened the Snap integration [in Ubuntu 18.10] within the desktop experience," explained Cooke.

But you don't have to go out of your way to install these applications.

Ubuntu 18:10 comes preinstalled with a decent set of applications that allow a user to get to work as soon as you boot into it.

Ubuntu 18.10 comes with LibreOffice (productivity suite), Firefox (web browser), Rhythmbox (music player), ToDo (task management), Thunderbird (email client), videos, and a text editor. For many end users, this collection is all you need.

The only change I made was configuring email clients. Although Thunderbird is fantastic, I prefer the tight integration that Evolution offers for email, contact, and calendar. It's very easy to configure it. Open Settings and go to the *Online Accounts* tab (Figure 4).

I use G Suite, so I chose Google. Provide it with account details, and it will ask your permission to give Gnome access to the account. Voila, Gnome can now manage your account (see Figure 5).

Evolution doesn't come preinstalled, but you can easily install it using either the command-line interface (CLI) or from Ubuntu Software Center.

To use the CLI, simply open the Terminal app and run these commands:

```
sudo apt update
```

the command will refresh the repository information so that it downloads the latest packages).

```
sudo apt install evolution
```

This command will install the Evolution email client.

If you prefer a GUI, open Ubuntu Software Center and search for Evolution, and then install it with one click.

Because I already configured an online account, as soon as Evolution was installed, my email account, calendar, and contact book began to populate (Figure 6).

Ubuntu is a great distro for gamers. With its large install base, the Ubuntu project tends to get good graphics processing unit (GPU) support for new GPUs. Ubuntu is also used for deep learning and machine learning workloads, so GPU vendors like Nvidia work with Canonical to support both physical GPUs and virtual GPUs (vGPU).

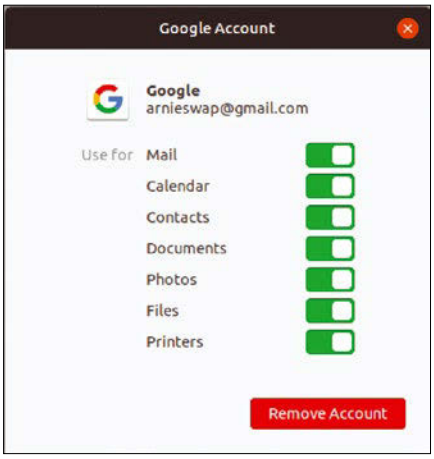


Figure 5: The Google Account dialog shows which services will be accessible by Gnome.

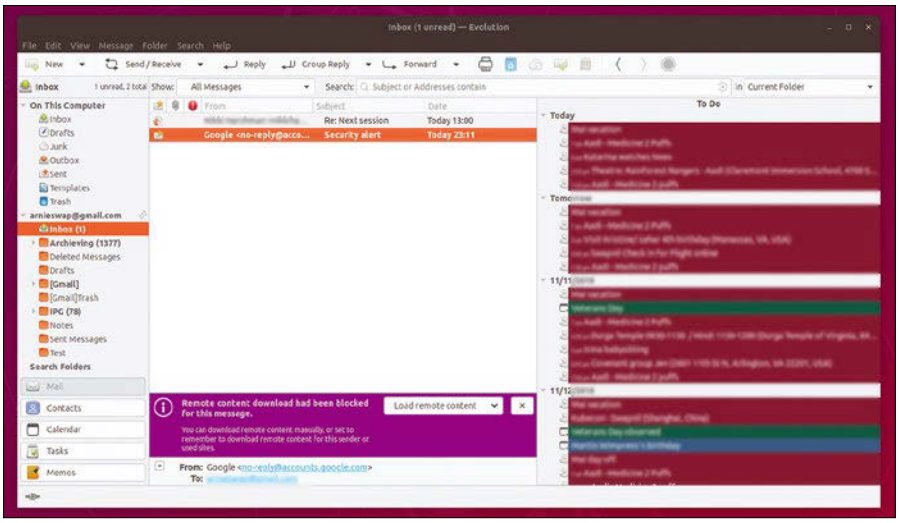


Figure 6: Once configured, Evolution offers tight integration with email, contacts, and calendar.

If you are a gamer, installing the Steam client will bring not only Linux games, but also many Windows games that Valve Software (the maker of Steam) is bringing to Linux via Wine.

While I'm on the subject of Wine, you can always use Wine and PlayOnLinux to install many Windows apps in Ubuntu.

Know Your Ubuntu

Once you install your Ubuntu, the first thing you must do is update your system. You can do it either with the CLI or a GUI-based Ubuntu Software management tool. If you are using the CLI, then run the following command:

```
sudo apt update && ➥
sudo apt upgrade
```

As you can see, this is actually two separate commands: The first command refreshes the repository info, and the second command installs any available updates. You can also run the following command:

```
sudo apt dist-upgrade
```

which will upgrade the packages to the latest version and also remove older packages.

Ubuntu comes with three tools for software management. The Ubuntu Software app [1] helps to install applications. The tool is also capable of installing system-wide updates. If there are any updates available, you will see them in Ubuntu Software app. You can also use the app to remove installed applications.

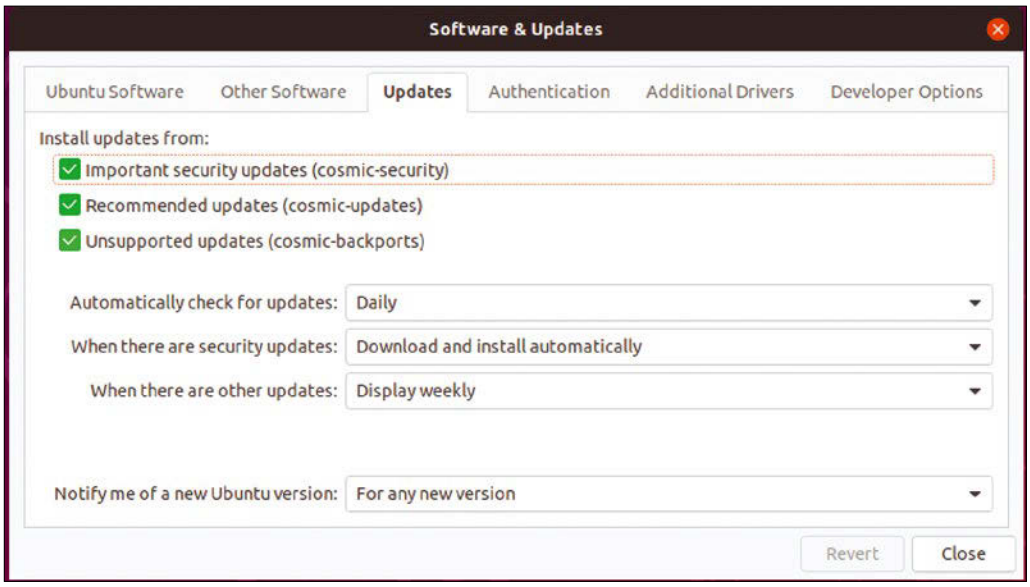


Figure 7: Get total control over how updates are managed on Ubuntu.

Software Updater is a tool whose sole purpose is to check available updates and install them.

However, if you want to fine-tune what kind of updates you want to install, you need the Software & Updates tool (Figure 7), which you can also open by clicking on *Settings* in the Software Updater tool.

The Software & Updates tool allows you to enable repositories, so you can download and install proprietary drivers and packages restricted due to copyright and other legal issues (Figure 8).

The *Other Software* tab gives you access to applications packaged by Canonical partners.

The *Updates* tab allows you to fine tune how frequently you want Ubuntu to check for updates and what kind of updates to install. I would recommend that you do updates as frequently as possible.

If you have any specific proprietary hardware, such as a GPU, the *Additional Drivers* tab gives you the option to use proprietary drivers for that hardware.

I wish all of these tools were part of Ubuntu Software, so that users could manage everything from one place.

Fine-Tuning Gnome

The default layout of Ubuntu 18.10 is lovely, but you can fine tune it to your liking if you feel like making changes. Install the Gnome Tweak Tool, which will allow you to change theme, icons, fonts, move windows titlebar buttons, and more.

You can also install third party extensions to add extra functionality to Ubuntu. Unfortunately, there is still no built-in tool to manage extensions. You need to open the Firefox web browser and visit *https://extensions.gnome.org/*.

Enable the browser extension from the pop-up dialog box. Now you can easily install third-party extensions and manage them from the Tweak Tool.

Apps You Need

Ubuntu comes with all the apps that you need to get started. However, you might have your favorite applications. I have my own set of apps that I install to be more productive.

Suppose you want to install VLC, a must-have app that will play virtually every video format. Instead of installing the version of VLC that is available in the Ubuntu repositories, suppose you want to set up the Snap version available directly from the VLC project, which means you get the latest version as packaged by the VLC developers.

Using Snap is as easy as using `apt` and `apt-get`. To search for the Snap version of the app, run the following command:

```
snap find vlc
```

This command will show all the available versions of the app. Now install it by running this command:

```
sudo snap install vlc
```

Firefox is a great browser, but I use Google Docs for my stories, and Google Docs shortcuts work better with Chrome. You can install Chrome by downloading the `.deb` package file and installing it. Or you can choose the command-line path and add the Google repository to your system, and then install it from the Terminal app.

First, you need to download the key:

```
wget -q -O - ➥
https://dl-ssl.google.com/➥
linux/linux_signing_key.pub | ➥
sudo apt-key add -
```

Now add the repository:

```
sudo sh -c 'echo ➥
"deb http://dl.google.com/➥
linux/chrome/deb/ ➥
stable main" >> ➥
/etc/apt/sources.list.d/➥
google-chrome.list'
```

Now refresh the repository, and install Chrome:

```
sudo apt-get update
sudo apt-get install ➥
google-chrome-stable
```

The good news with Chrome is that it comes with the Digital Rights Management (DRM) extension enabled, so you will be able to watch movies and TV shows from web-streaming services like Netflix.

If there are other apps that you need, you can very easily install them using either Snap, `apt`, or simply finding a `.deb` binary from the official site. However, I would recommend going with the Snap route.

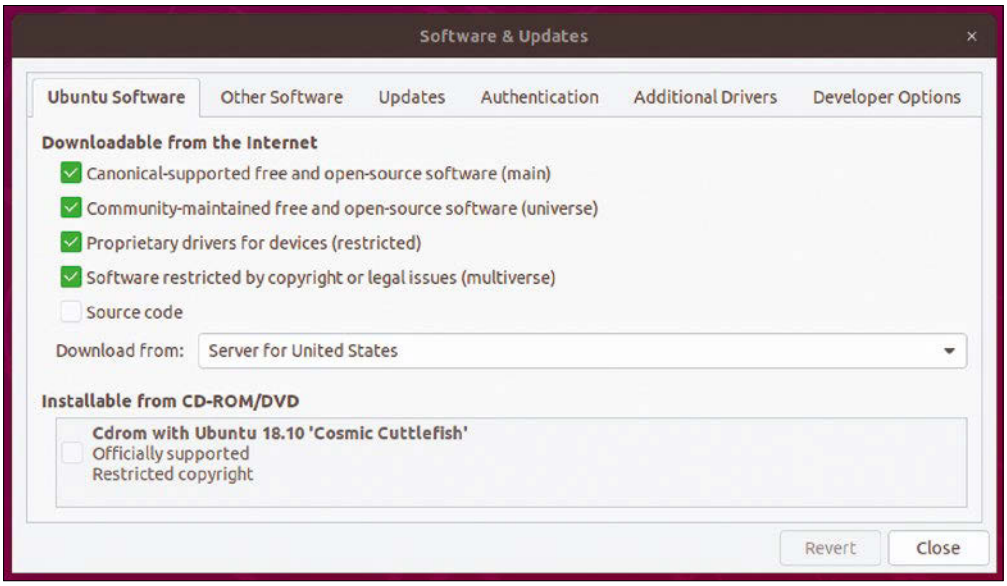


Figure 8: If needed, you can install restricted drivers and packages.

Final Thoughts

The Ubuntu project is back to being what a good distribution should be, and you can once again benefit from the fruits of upstream projects like Gnome. ■■■

Info

[1] Software Center is an app to manage Software add/remove: *http://manpages.ubuntu.com/manpages/xenial/en/man1/software-center.1.html*

EncryptPad is a handy text editor with encryption

Secure Writer

EncryptPad provides symmetric text encryption directly from the editor. You can also use EncryptPad to encrypt binary data. *By Ferdinand Thommes*

EncryptPad [1] is a text editor with an encryption function. If you have a file with sensitive information, such as passwords and account names, you can use EncryptPad to edit and maintain the file, all the while ensuring that the file remains encrypted. EncryptPad also provides a data integrity feature through the SHA-1 hashing algorithm, so if you receive a file from someone else, you can ensure that it hasn't been altered in transmission. EncryptPad even protects you from a brute force attack by letting you safeguard the encrypted information with both a key and a passphrase. Store the key separately; if the USB stick with the encrypted information falls into the wrong hands, the thief won't have access the to data just by guessing the password.

In contrast to other common encryption tools, EncryptPad uses symmetric (rather than asymmetric) encryption. You use the same key to encrypt and decrypt the file. If you are sending the encrypted file to someone else, you need to provide the passphrase and/or key file to the other user separately.

EncryptPad, which has been on GitHub for about three years [2], is still at the beta stage; the current version at this time of writing is 0.4.0.4. The software is available for Linux, Mac OS X, and Windows. EncryptPad relies on OpenPGP [3] [4], and it uses the AES 256 symmetric encryption standard [5]. Some of the functionality of EncryptPad is similar to the `gnupg.vim` extension for the Vim editor.

You can access EncryptPad through your desktop interface or use the `encryptcli` command to run the application at the command line.

AppImage

EncryptPad has not yet made it into the archives of many Linux distributions, although you will find it in the Ubuntu PPA, the Arch Linux AUR, and FreeBSD. The developer offers the application for Linux as an AppImage [6] with a size of 29MB. Note that you have to adjust the permissions for the file before the first start. To set the permissions, use the `chmod +x filename` command in a terminal window.

To launch EncryptPad, double-click on the file; later on, you will be able to start from the application menu in some distributions. At first glance, the main window looks like a regular text editor (Figure 1).

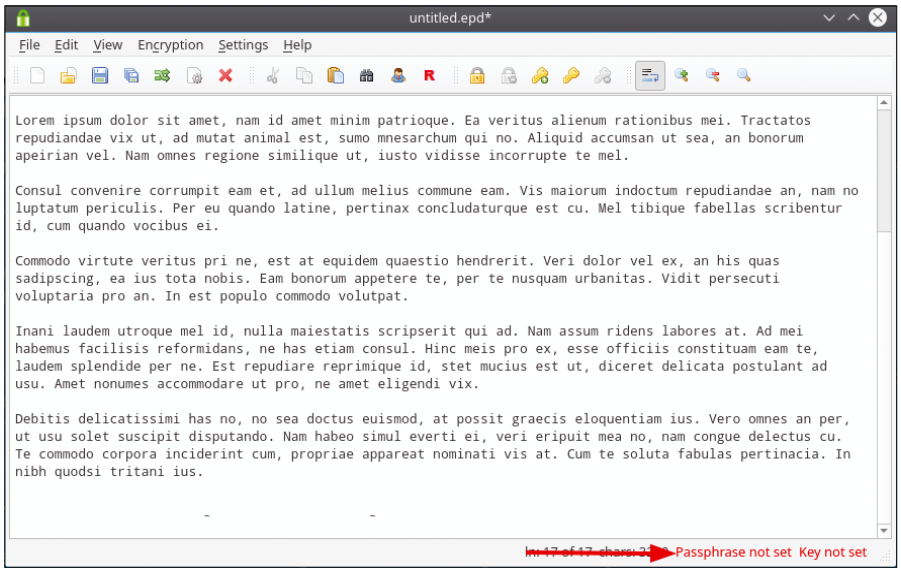


Figure 1: EncryptPad looks similar to classic text editors. Note the arrow at the bottom pointing to the encryption status.

But if you mouse over the icons or browse the menu, you'll find the functions for creating passphrases and key files, as well as a read-only mode that ensures that you don't unintentionally change important documents.

The Settings menu mainly contain parameters for creating passphrases and key files. If necessary, you can adjust these settings directly when you execute an action. However, you might want to configure the font and the number of files displayed in the Open dialog to suit your requirements.

If you plan to use keys, you will also need to specify the path to the cURL binary file. You can determine the path by typing `which curl` in a terminal window; it may be necessary to install the program via the package manager. cURL lets you download keys directly from a remote server in EncryptPad.

Two Formats

EncryptPad supports two file formats: GPG and EPD. The GPG file type is for the OpenPGP format and is compatible with other OpenPGP tools. You can use it when opening a file even if EncryptPad is not available. The format does not support double protection with a key file and passphrase.

When using GPG, it is not possible to store the path of the key file in the encrypted file itself, so every time a file encrypted with a key file is opened, you are prompted to choose which key file the editor should use.

EPD is the native EncryptPad file format. Other OpenPGP software can open an EPD file as it is only protected with a passphrase, because then it is effectively a GPG file.

If you use a key or a combination of key and passphrase to protect the data, the program packs the GPG file into a WAD container. WAD [7], which stands for "Where's All the Data," is a simple format for combining multiple binary files. You can open WAD containers with SLADE [8].

The simplest case to protect text with EncryptPad is via the *Save As* function. In the case of unprotected text, you will see the information *Passphrase not set* and *Key not set* in red at the bottom right of the window.

Select *GnuPG (*.gpg)* in the Save dialog under *Files of Type*, and add the rest of the information, such as a filename and a desired storage location. Before the application saves the text, it prompts you to enter a password. After saving, the *Passphrase not set* message in red changes to a *Password protected* message in black.

The result is an OpenPGP file that no longer relies on EncryptPad for decryption but can be opened with standard GPG tools.

Using a Key

To protect text with a key, open or create the text file. Then click on the icon with the key and the plus sign. If you already

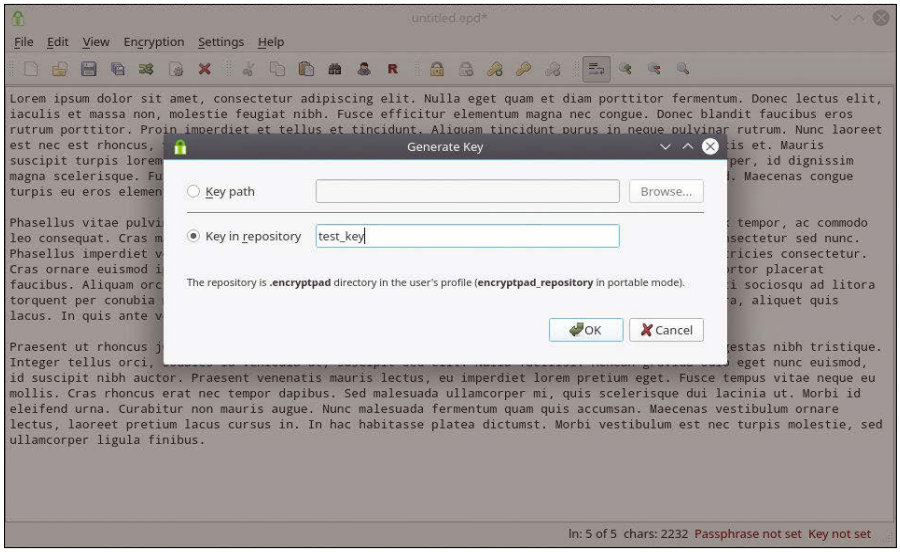


Figure 2: The Generate Key dialog box lets you specify a path for saving the key, or you can name the key and store it in the key repository.

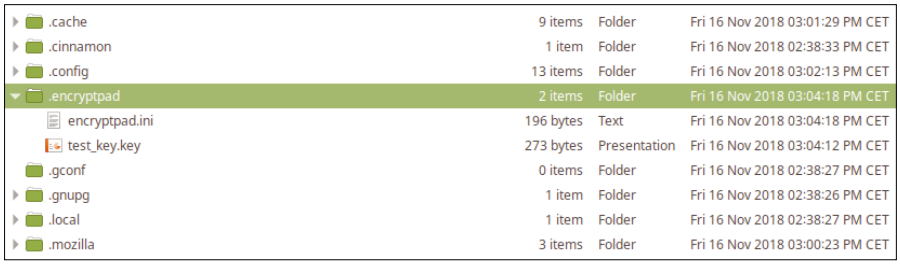


Figure 3: EncryptPad hides the created keys in the user's home directory.

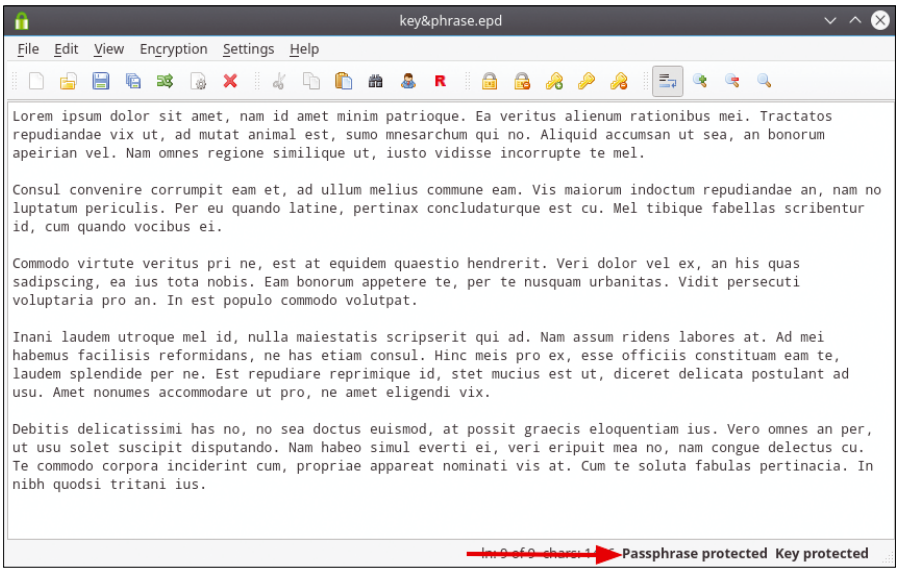


Figure 4: Double protection with a passphrase and a key: One password protects the key; a second protects the resulting file on the hard disk.

have a key that you want to use, enter the path in the upper field of the input screen. To create a new key, enter a name in the Generate Key dialog box (Figure 2).

When you create a key file, EncryptPad creates a random byte sequence, prompts the user for a passphrase, encrypts the resulting sequence with it, and stores the results in a file. The application saves the key file with the `.key` extension in the home directory of the user below the hidden `~/.encryptpad` folder (Figure 3).

Now press *OK* and enter a passphrase to protect the key. The software will ask you if you want to use the key for the open file. After you've said yes, nothing seems to happen. However, if you take a look at the `~/.encryptpad` folder below your home directory, you will see that the key is already there. Further indicators are the *Key not set* message changing to *Key protected*. In addition, the icon with a plus sign now has a minus sign that lets you remove key protection for this file.

To open the currently encrypted file later on, first enter the location of the key in the dialog. The software automatically detects this correctly as long as you do not move the key file. In a second step, enter the passphrase that you have assigned.

The whole thing can be nested one level further by protecting the key file with an additional password. Proceed as in the previous example, but do not select `.gpg` as the file format; instead, go for `.epd`. The software will ask you for an additional passphrase. The second passphrase protects the file on the hard disk. When the process is complete, you should see *Passphrase protected* and *Keyword protected* at the bottom (Figure 4).

If you want to close the file and open it again later, you will be prompted for the passphrase within the file and then for the key. For simplicity's sake, you will want to check *Persistent key location in the encrypted file*, because this option will eliminate the need to query the key in the future. All you need to do is enter the two passwords. If this option is set, EncryptPad shows you the option at the bottom in the *Keyword protected* section (Figure 5).

Binary Files

EncryptPad also lets you encrypt binary files. You create a new key or decide to use an existing one. Say *no* when you are asked whether the software should use the key with this file.

In the next step, click on the fifth icon from the left with the crossed green arrows. It is labeled *File Encryption*. The selection is already set to *Encrypt*; below that you select either EPD or GPG as the file format. You should only use EPD if you want to use a passphrase and a key. If you will only be using one of these options, change the format to GPG to retain compatibility with OpenPGP.

Now select the file to encrypt as the *Input File*. The editor automatically suggests the path and name of the output file; normally, you will not need to change anything. Enter the storage location for the key or set a password (Figure 6).

If you use both a key and a password, it is possible to store the key persistently in the file. You will be protecting the file with a passphrase. But if you choose GPG as the format, this option has no effect. Click on *Start*. The encryption process can take some time, depending on the size of the file you wish to encrypt.

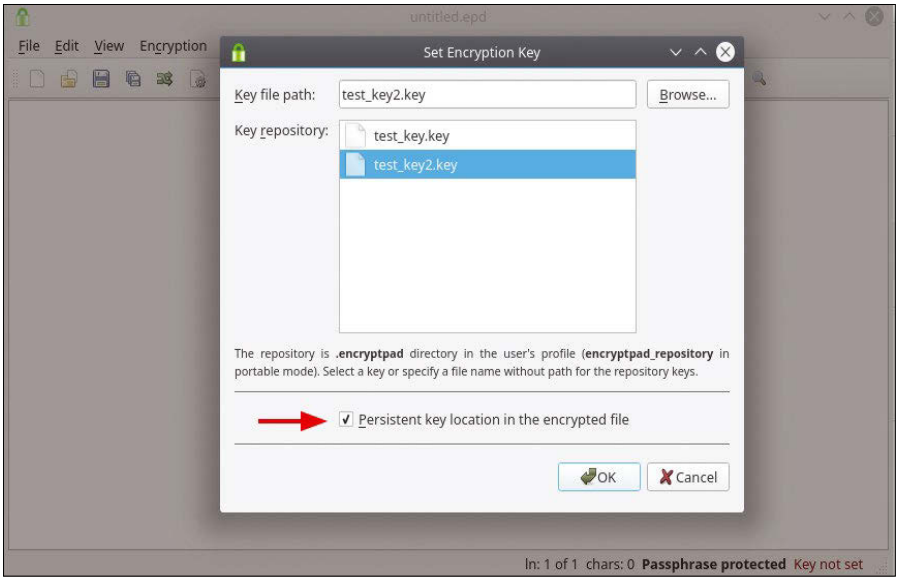


Figure 5: The check mark saves you some typing later on: EncryptPad stores the path to the key in the file itself and automatically adopts it when opening the file.

Conclusions

EncryptPad is a simple text editor that provides symmetric encryption as well as data integrity through SHA-1. You can also use EncryptPad to encrypt binary files.

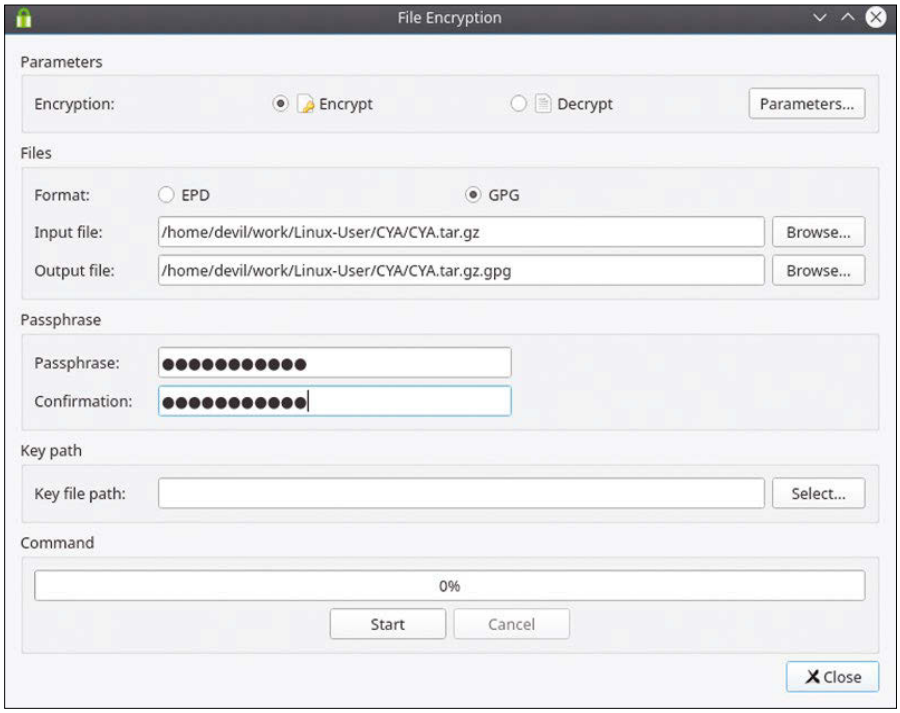


Figure 6: EncryptPad opens a new dialog for encrypting binary files. Both GPG and the native EPD are available as formats.

The developer points out that the software is still beta and is therefore not yet suitable for use in critical areas. The documentation [9] on the website is extensive and covers tutorials, technical backgrounds, and known weaknesses.

Info

[1] EncryptPad: *https://evpo.net/encryptpad/*

[2] EncryptPad on GitHub: *https://github.com/evpo/EncryptPad*

[3] OpenPGP: *https://en.wikipedia.org/wiki/Pretty_Good_Privacy#OpenPGP*

[4] RFC 4880: *https://tools.ietf.org/html/rfc4880*

[5] AES: *https://en.wikipedia.org/wiki/Advanced_Encryption_Standard*

[6] EncryptPad download: *https://evpo.net/encryptpad/downloads.aspx*

[7] WAD: *https://en.wikipedia.org/wiki/Doom_WAD*

[8] SLADE: *http://slade.mancubus.net/index.php?page=wiki&wikipage=Installation*

[9] EncryptPad documentation: *https://evpo.net/encryptpad/default.aspx*

Workshop: Fitting a PCIe SSD with your existing hardware

AFTERBURNER

A PCIe SSD can accelerate your system considerably, but you need to do your homework and choose the right product for your computer. *By Erik Bärwaldt*

PCIe is a downward-compatible interface technology with standardized slots, but the chaos with interface cards has not gone away. If you want to use fast mass storage to upgrade older systems without UEFI BIOS and NVMe support, you will certainly benefit from the advantages of the Peripheral Component Interconnect Express (PCIe) bus. However, before you buy expensive components, it makes sense to investigate the specifications; otherwise, you may end up spending several hundred dollars on equipment you can't use.

Understanding PCIe

PCIe [1] is a standard for extensions in personal computers that has been in use for around 15 years, and by now, it has completely replaced its predecessors PCI, PCI-X, and AGP.

The PCIe bus thoroughly eliminates many of the problems of its predecessors. Unlike the PCI bus and its variations, PCIe works with serial connections. Point-to-point connections are created between the components, which leads to significantly higher data transfer rates with significantly less overhead, because slow devices on the bus cannot slow down the faster devices (Table 1).

With the legacy bus topology used by the PCI standard, all the connected components have to share the available bandwidth; with the PCIe specification, a switch integrated in the chipset enables the individual connections. The devices can use several lanes for data transmission; unlike the PCI bus, lanes do not have a specified and fixed clock signal. Depending on the specification, up to almost 2000MB/s can theoretically be transferred per lane. The PCIe bus operates in duplex mode, which enables simultaneous data transmission in both directions.

Since the PCIe bus is hot-pluggable, the system is also suitable for high-performance mass storage devices that can be swapped on-the-fly if defects occur. Especially in high-performance workstations and servers, the PCIe standard replaced the PCI standard at a very early stage.

However, even today, you occasionally find a mixture of one or two conventional PCI slots and (usually) four to six PCIe slots in such computer systems. Mixed operation of the different standards is also

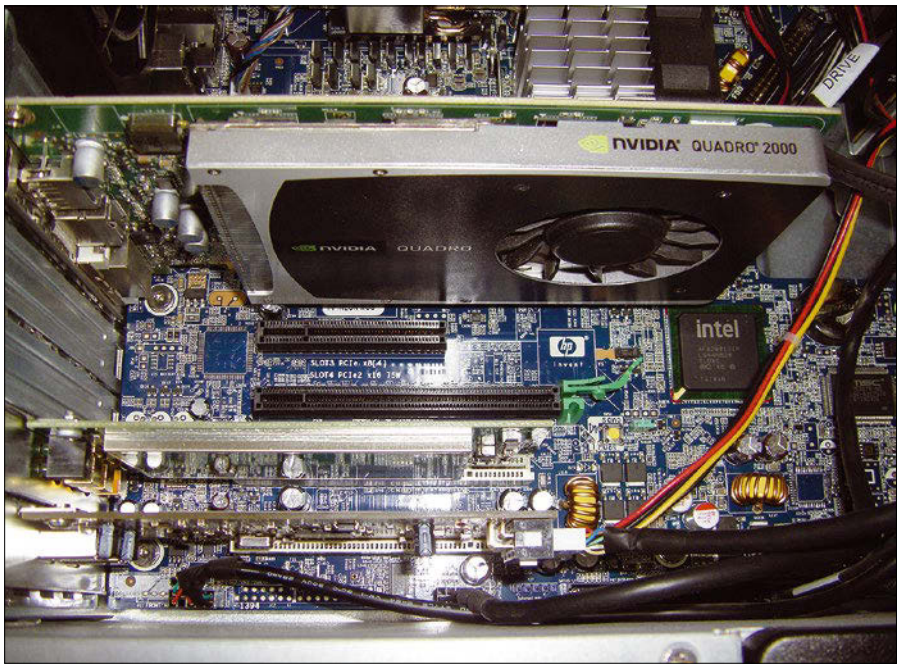


Figure 1: Different PCIe slots in an HP Z600 workstation.

Table 1: Maximum PCIe Specification Transfer Rates

Version	PCIe x1	PCIe x4	PCIe x8	PCIe x16
PCIe 1.x	250MB/s	1GB/s	2GB/s	4GB/s
PCIe 2.x	500MB/s	2GB/s	4GB/s	8GB/s
PCIe 3.x	0.98GB/s	3.93GB/s	7.87GB/s	15.75GB/s
PCIe 4.x	1.96GB/s	7.87GB/s	15.75GB/s	31.50GB/s

possible. Thanks to upward and downward compatibility, you can operate plugin cards of different specifications without the cards influencing each other in terms of performance.

Lanes

Data transfer within a PCIe infrastructure takes place along one or more lanes. The maximum transfer speed has increased with each new specification, and the connectors have changed accordingly. In addition to PCIe x1 connectors with only one lane, there are also standardized x4, x8, and x16 slots.

Short slots are sufficient for relatively slow plugin cards (such as USB 3.0 interfaces), but graphic cards, PCIe SSDs, and high-performance network cards in particular depend on many lanes and thus long slots. In the case of graphics cards, the problem of power supply also arises: Modern graphics adapters consume 300W or more of power.

Because a PCIe slot typically does not deliver more than 75W, such graphics cards require external power connections. The standard provides for standardized connectors for this purpose. In high-performance workstations, for simultaneous use with multiple graphics cards and multi-monitor operation, you'll need multiple PCIe x16 slots, along with a correspondingly dimensioned power supply (Figure 1).

Disadvantages of Modern SSDs

Especially with high-performance components such as PCIe SSDs, the use of the devices must always be compared with the technical capabilities of the host computer before you decide to purchase. Although the slots match mechanically, even across different specifications, and could therefore accommodate components of all standards, not every PCIe SSD is suitable for every computer.

Older computers with the PCIe 1.x and 2.x standards do not support all modern NVMe SSDs, but mostly only older ones that follow the AHCI standard. Since such computer systems still rely on a legacy BIOS, the SSDs must also provide a Boot ROM that allows the computer system to boot from the SSD despite having an older BIOS. Without the Boot ROM, an SSD in a computer with a legacy BIOS is not recognized as a bootable device.

The Boot ROMs of modern PCIe SSDs, however, usually only have routines for devices with a UEFI BIOS, so such mass storage devices do not work with older computer hardware despite the manufacturer claiming Linux compatibility. However, problems can also arise with seemingly suitable specifications. For example, certain older high-performance SSDs with a PCIe 2.x interface must be integrated with proprietary driver modules on Linux for them to be recognized as fast mass storage devices (Figure 2).

Kernel modules for these SSDs are usually available directly from the manufacturer, but only for a few distributions and often only for certain kernel versions. Although the support and thus also the driver development for such very expensive high-end components is generally much better (and also designed for longer periods of time) than for consumer products, the manufacturer's support expires after a few years at the latest.

Figure 2: A PCIe SSD that can only be addressed with its own kernel driver on Linux.

In these cases, there is also the drawback that such SSDs promise an enormous speed boost in servers and workstations but are not bootable due to the lack of operating system integration. In such cases, another mass storage device is needed to boot the computer.

A further drawback when using high-performance SSDs lies in the high power draw and what can be a massive heat generation associated with it. Very fast PCIe SSDs therefore come with active cooling out of the box, as do high-performance graphics cards. Since these SSDs generally require longer slots with a higher number of lanes, it is important to ensure during installation that there is sufficient clearance for air to circulate between the components.

High-performance workstations, often equipped with eight to twelve internal fans, provide good air circulation out the box. For smaller systems, you need to make sure that there is enough space between an actively cooled graphics card and an SSD with its own fan, otherwise heat build-up may occur.

Since the individual memory cells of the SSDs usually also passively dissipate heat through a large heat sink, it may not be possible to use two adjacent PCIe slots simultaneously for such high-energy components.

Alternatives

Older workstations and servers usually have several second-generation PCIe slots of different lengths. Since these machines, which often still use a conventional BIOS without an NVMe module, cannot cope with current NVMe SSDs, you can turn to AHCI SSDs. They support the AHCI protocol and work with older systems.

SSDs that follow the AHCI specification are available in all common form factors and as PCIe plugin cards. Conventional 1.8 or 2.5-inch form factor SSDs communicate with the PC via the SATA interface, achieving a maximum throughput of 6Gbps. Thus, with a conventional SSD, a maximum data throughput of around 500 to 550MB/s is possible.

PCIe SSDs with AHCI support are often connected in a RAID array – if the capacity allows it – and thus reach higher speeds. If you still have a computer system that only supports the SATA II standard, you will already feel a significant increase in speed with a small PCIe SSD that follows the SATA III standard, without having to buy a controller card.

But even if you have identified a PCIe SSD with AHCI support, the devil lurks in the details: Some of these models – for example, by Fusion-io, which now belongs to the SanDisk Group – do not have their own firmware and ROM-based bootloader (OPROM) on board. Although this allows older computer systems to detect these drives, Linux cannot boot from them.

Even for data storage only, such SSDs have restrictions: In order to integrate them on Linux, you need to integrate a specific kernel module for controlling the controller into the system. This is usually provided by the manufacturer. However, most manufacturers are not interested in supporting more than just a few distributions. In addition, support often expires after a few years.

Other manufacturers of PCIe SSDs, such as OCZ or Mushkin, integrate the firmware directly on the board of the plugin card. The software automatically self-initializes when the computer boots. The BIOS then loads it so that the drive is immediately available as a mass storage device.

BIOS – A Sticking Point

Many older BIOS variants, which do not yet support the UEFI specification [2], often cite USB memory sticks and memory cards as bootable devices, but do not allow booting from PCIe hardware such as PCIe SSDs. They often display a message during initialization indicating that booting is only possible from hard disks and CD-ROM drives.

In such cases, it is a good idea to check for BIOS updates for the appropriate motherboard. Especially with newer models, manufacturers sometimes retrofit the capabilities to boot PCIe SSDs.

OCZ RevoDrive 350

OCZ Technology Group, now part of the Japanese Toshiba Corporation, has been making a name for itself in the mass storage market for years. In addition to conventional SSDs, OCZ also manufactures PCIe SSDs, such as the RevoDrive 350 [3]. This very fast drive is suitable for computers with conventional PCIe slots from the second generation. However, you will only find end-of-line stock on offer.

The drive uses Toshiba NAND MLC memory produced in the 19nm manufacturing process. Depending on the model, the capacity is between 120GB and 480GB. The AHCI-compatible SSD contains a maximum of four SandForce SF2282 controllers, which can be configured as a RAID 0 network. The RevoDrive 350 also has its own firmware and is therefore compatible with Linux.

After inserting the SSD board, the drive is initialized when booting the computer and is available as mass storage under current distributions. In the larger variants, with 240GB and 480GB capacity, it is noticeable that the usual partitioning tools do not display one, but two or four drives with 120GB capacity each.

This strange construct results from the architecture of the hardware: OCZ uses one SandForce controller per 120GB capacity to control the NAND memory devices. These controllers, in turn, are connected to their own RAID controller located on the SSD, which the system addresses via a virtual software layer.

This ensures that the two or four controllers each work in a RAID network and thus achieve almost twice or even four times the transfer speeds of the SATA III standard. If a corresponding driver is missing for the proprietary RAID controller, then only the individual drives can be addressed.

OCZ/Toshiba provide kernel modules on their support pages to control the RAID controller, but these modules are only suitable for older editions of Fedora, Ubuntu, and Linux Mint. Thus, the drive, like most PCIe SSDs with AHCI support, is not usable for booting current Linux derivatives, nor as a single fast mass storage device. The RevoDrive therefore only works like a conventional SATA drive and only achieves the typical SATA III data transfer rates in read and write tests.

M.2 or NGFF

Another alternative for using SSDs in older systems via the PCIe interface is to use of adapter cards. Modern SSDs with the M.2 form factor [4] are used. This relatively new technology is also

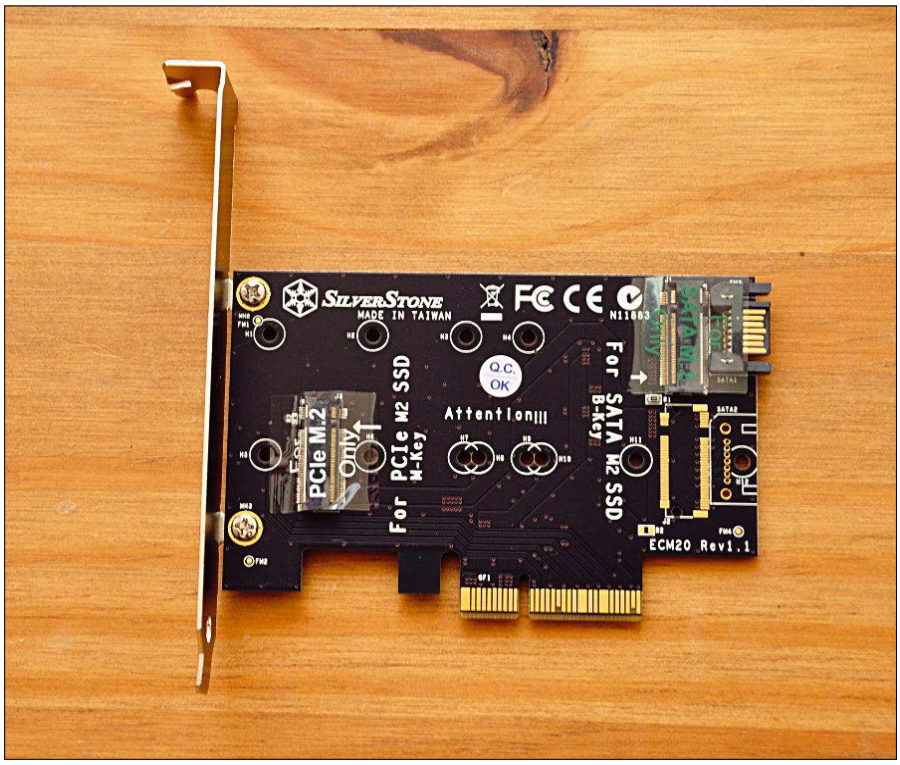


Figure 3: The SilverStone adapter card offers two different M.2 slots.

available with support for both AHCI and NVMe and was originally only intended to replace the mSATA interface in notebooks.

However, new stationary computer systems are increasingly being equipped with M.2 interfaces that are directly connected to the PCIe bus. However, these interfaces, also called Next Generation Form Factor (NGFF) interfaces, are again available in a variety of designs, so that it is essential to obtain precise information before purchasing such an SSD.

M.2 SSDs are available in different lengths and widths. Typically, they are 42, 60, or 80mm, so not every SSD will fit in every slot. In addition, different recesses on the contact strips (often known as keys) sometimes prevent the use of certain models. Usually, M.2 SSDs fit in sockets that support either Key M or Key B.

The sockets differ not only in terms of the cut-outs for the plugin cards, but also in the number of PCIe lanes supported: Cards that have Key M and Key B cut-outs use two PCIe lanes, while those that only have Key M cut-outs use four PCIe lanes.

As a rule of thumb: SATA SSDs in the M.2 form factor have either a single Key B or a Key M plus a Key B cut-out. PCIe SSDs with AHCI or NVMe support, on the other hand, have only one Key M cut-out.

Card Game

If your PC does not yet have an integrated NGFF slot, then adapter cards for PCIe slots, which provide one or two and sometimes also four M.2 sockets, are available as an alternative. There are even cards that have an additional mSATA connection.

When selecting the adapter cards, you have to look carefully. The slots for the M.2 SSDs are usually the same length and width, regardless of the interface. However, especially for those adapter cards that accept two M.2 models, one is usually intended for SSDs with Key B and the other for Key M.

The connection for Key B is only designed for SATA SSDs and is looped through to the controller already present in the computer via a standard SATA cable. True PCIe SSDs in NGFF form factor, on the other hand, integrate into the system via the Key M slot. It is therefore advisable to read the technical specifications carefully when buying new adapter cards and also to look at the printing on the adapter card if you are unclear about the slot assignment.

Most manufacturers now print explicit warnings on their adapter cards to avoid confusion. In addition, before purchasing a new M.2 SSD, you should read its technical description carefully: In many cases, it is not immediately clear whether it is an M.2 NVMe SSD, an M.2 AHCI SSD, or an M.2 SATA SSD.

Since the PCIe SSDs have Key M cut-outs, it is easy to get confused. The Kingston M.2 Type UV500 SSD is a SATA SSD, for example, while the HyperX Predator model from the same manufacturer is available as an older PCIe AHCI variant and – under the same type designation – as a newer PCIe NVMe version. Externally, the three M.2 SSDs differ only marginally.

Lite-On CX1

For our test, we chose a PCIe-AHCI-SSD with a Lite-On CX1 [5] as an example. The mass storage module is available in variants between 128 and 512GB; under the hood, the SSD uses NAND memory from Toshiba. The technical data of the SSD clearly identifies it as a PCIe SSD with maximum read and write rates of 730 and 830MB/s, respectively, which requires a version 2 PCIe slot with four lanes.

The Lite-On SSD is facing off against an adapter by Taiwan-based manufacturer SilverStone [6], which allows the simultaneous use of two M.2 SSDs in dual-slot operation: One of the slots accommodates

Figure 4: The Lite-On SSD complies with the PCIe AHCI standard.

Figure 5: The Wings X2 adapter by Austrian manufacturer Angelbird accepts two conventional SATA SSDs.

a SATA SSD, and the second is designed for a PCIe mass storage device. The slots are clearly marked, so there should be no confusion (Figure 3).

The SATA slot is also connected to the appropriate controller on the computer's motherboard via a matching cable. This design ensures that an M.2 SATA SSD installed on the adapter card is capable of booting the operating system installed there in any case (Figure 4).

The Lite-On SSD fits into a PCIe slot with a Key M assignment. We then installed the adapter in a free PCIe x4 slot. The operating system – in our lab, Linux Mint 19 "Tara" – detected the PCIe SSD without additional manual work and addressed the entire drive with its correct capacity of 512GB.

The performance test revealed that, as expected, the SSD is not one of the fastest: With a read rate of 678MB/s and a write rate of 526MB/s, the small M.2 drive is only slightly above the maximum possible transfer rates of the SATA III standard. However, the Lite-On SSD proves to be a true sprinter compared to the RevoDrive under Linux. The access times of the RevoDrive are also significantly higher than those of the Lite-On CD.

SATA SSDs

A round-about way to use the PCIe interface for fast mass storage is to use adapter cards for traditional SATA SSDs. These cards offer space on a PCIe board for two standard SATA SSDs. A RAID controller mounted on the board makes it possible to operate the disks in a RAID 0 network and thus achieve respectable speeds that are almost twice the speed of the SATA III standard.

When purchasing the equipment, you will want to check whether the ROM of the plugin card contains a bootloader [7] and whether the system can boot from the drives installed on the card (Figure 5).

Conclusion

PCIe technology opens up entirely new options, especially in the area of high-performance components such as network adapters, graphics cards, and mass storage. The path to a fast system based on PCIe SSDs, however, is paved with stumbling blocks due to dependencies on external influences such as BIOS variants and boot options, which can lead to costly mis-investments if you make ill-considered purchases. ■■■

Info

[1] PCIe: *https://pcisig.com/specifications/pciexpress/*

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

[3] RevoDrive 350: *https://ssd.toshiba-memory.com/en-amer/*

[4] M.2: *https://en.wikipedia.org/wiki/M.2*

[5] Lite-On CX1: *http://www.liteonssd.com/m/Products/product.php?alias=CX1-SERIES*

[6] SilverStone ECM20: *https://www.silverstonetek.com/product.php?pid=575*

[7] Angelbird Wings X2: *https://www.angelbird.com/prod/wings-x2-hba-blue-1316/?category=1*

Available Now

* 2018 EDITION *

Linux Magazine Archive DVD

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

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

ORDER NOW!

shop.linuxnewmedia.com

Performance gains with goroutines

Simultaneous Runners

In the Go language, program parts that run simultaneously synchronize and communicate natively via channels. Mike Schilli whips up a parallel web fetcher to demonstrate the concept. *By Mike Schilli*

I often wonder why some developers seem committed to designing new programming languages. Of course, the young guns today are all hungry for slight improvements in the syntax, while hipsters enthuse over smart ideas for compact code. But the effort of building an ecosystem and setting up a community is immense!

Alas, since processors stopped running faster every year some time ago and only simulate more speed with cores running in parallel, one thing is very important: Your choice of language has to be able to coordinate parallel program parts easily. When I visited the WhatsApp team at Facebook in Menlo Park after work a few months ago, I learned what the secret of the small team's success was when they used a handful of machines to text millions of users. They used the old-fashioned Erlang language, which has parallelism as a native feature.

It's the same with Go. The smart people at Google have not only built process management and threading into the programming language, but also have added new primitives such as goroutines and channels, thus not only making concurrency available, but an integral part of the language.

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.

All Inclusive

To prepare my experiment, Listing 1 first builds a little helper, a library for easy web access [1]. Users later simply call `httpsimple.Get()` and receive a success or error code, as well as the text of the retrieved web page. The `Get()` function is given an initial capital so that external clients can use it from the package later, as required by Go. As the declaration in line 10 shows, it accepts a URL of the `string` type as an argument and returns two values: the result string with the content of the web page and an error value, which is set to `nil` in case of successful access.

Go aficionados can simply create web clients from the `net/http` core package with `http.Get()`, but for parallel access, the client should also be able to pull the ripcord in case of hanging web pages or sluggish data traffic. According to re-

Listing 1: httpsimple.go

```
01 package httpsimple
02 import(
03   "fmt"
04   "net/http"
05   "io/ioutil"
06   "time"
07   "errors"
08 )
09
10 func Get(url string) (string, error) {
11   tr := &http.Transport{
12     IdleConnTimeout: 30 * time.Second,
13   }
14   client := &http.Client{Transport: tr}
15   resp, err := client.Get(url)
16
17   if err != nil {
18     fmt.Printf("%s\n", err)
19     return "", err
20   }
21
22   if resp.StatusCode != 200 {
23     return "", errors.New(fmt.Sprintf(
24       "Status %v", resp.StatusCode))
25   }
26
27   defer resp.Body.Close()
28   body, err := ioutil.ReadAll(resp.Body)
29   if err != nil {
30     fmt.Printf("I/O Error: %s\n", err)
31     return "", err
32   }
33
34   return string(body), nil
35 }
```

ports [2], the default client is not suitable for this, therefore lines 11-13 in Listing 1 define a transport that sets the timeout to 30 seconds. And it's great that the client can also speak HTTPS, as if it were the most natural thing in the world!

The rest of Listing 1 is used for error handling, checking the status code (which should be `200`), and requesting and reading the web page text arriving via the socket. The function returns the empty string as the result and an error code in the event of a premature termination. In lines 19 and 31, it only passes on the error values provided by the core libraries `net/http` and `io/ioutil`, while in lines 22-25, it even compiles a new error type if it receives a status message other than `200` from the web server.

If all goes well, line 34 returns the page text converted to a string and the error value `nil` to the caller. For a client to find the helper later on, I have to copy the Go code from Listing 1 into a new directory `~/go/src/httpsimple` and then compile it there using `go install` to make it available as a library for other Go code.

One by One

Listing 2 now calls the web servers of some large US companies, one after the other, and fetches their homepages with the new `httpsimple` library [3]. To do this, it defines an array of strings with their URLs in lines 9-13 and iterates over this in a `for` loop from line 15. Instead of outputting the whole mess of incoming web data, it uses `len()` to determine the data length and outputs it for illustrative purposes. Figure 1 shows the call to the compiled binary (created via `go build http-serial.go`), wrapped with the command-line timer `time`, revealing that the whole action takes a little over two seconds.

How could this data retrieval be accelerated? The web client is by no means fully loaded but waits patiently until the web server finally serves up the data; this wait must feel pretty much like an eternity to a fast CPU. It would be more effective if the web client were to send the requests to all four web servers at once and then collect the incoming data as it trickles in. This could be done either with several parallel running processes, with lightweight threads, or with an event loop, as in Node.js, for example.

Own Soup

In addition to the above, Go offers goroutines as a concurrency primitive. Their lifetime is planned and executed by the Go runtime. They are even more lightweight than threads, since several goroutines share one thread. The `go` keyword – followed by a function call – starts off a parallel goroutine in the background, executing the function, but also jumps to the next line to continue executing the main program. Nice! However, if you execute the example program below, with only a few calls to goroutines that output one letter at a time,

```
go fmt.Println("a")
go fmt.Println("b")
go fmt.Println("c")
```

you will be surprised that nothing appears at all on standard output while the program runs and then ends abruptly! The reason for this is that although Go starts the three routines in parallel, it closes the main program so quickly that none of the spawned program flows reaches its `Println()` command.

An interesting race condition occurs when I add a `Sleep` statement from the `time` package, delaying the program end by a few microseconds in line 10 of Listing 3. The output of the program then varies between nothing, one, two, or three letters, depending on how far the program gets in the given time, but this is obviously not deterministic (Figure 2).

Waiting for Stragglers

Extending the length of the `Sleep` command in the main program and hoping for good luck that all the underlings have finished their work in the meantime is obviously not a good solution. If the computer is busy with costly operations in other processes in the meantime, it is possible that the duration will extend to a few seconds, and the race is on again.

For every goroutine to have a guaranteed outcome, the main program and the routines have to communicate. At the end of the program flow, the main program has to wait until each routine has successfully completed before it can shut down the main process. In the form of the `sync` package, Go offers

Listing 2: http-serial.go

```
01 package main
02
03 import(
04   "fmt"
05   "httpsimple"
06 )
07
08 func main() {
09   urls := []string{
10     "https://google.com",
11     "https://facebook.com",
12     "https://yahoo.com",
13     "https://apple.com"}
14
15   for _, url := range urls {
16     body, err := httpsimple.Get(url)
17     if err == nil {
18       fmt.Printf("%s: %d bytes\n",
19         url, len(body))
20     }
21   }
22 }
```

```
$ time ./http-serial
https://google.com: 11439 bytes
https://facebook.com: 585409 bytes
https://yahoo.com: 496435 bytes
https://apple.com: 65926 bytes

real    0m2.105s
user    0m0.118s
sys     0m0.049s
$ _
```

Figure 1: With requests fired one after the other, the Go client retrieves all four URLs from the network in a good two seconds.

```
$ go build racecond.go
$ ./racecond
a
b
$ ./racecond
a
$ ./racecond
a
c
b
$
```

Figure 2: Three different results for three consecutive calls due to unsynchronized goroutines.

Listing 3: racecond.go

```
01 package main
02 import "fmt"
03 import "time"
04
05 func main() {
06   go fmt.Println("a")
07   go fmt.Println("b")
08   go fmt.Println("c")
09     // unreliable!
10   time.Sleep( 50 * time.Microsecond )
11 }
```

Listing 4: gochannel.go

```
01 package main
02 import "fmt"
03
04 func main() {
05   done := make(chan string)
06
07   go func() { done <- "a" }()
08   go func() { done <- "b" }()
09   go func() { done <- "c" }()
10
11   defer close(done)
12
13   for i := 0; i <= 2; i++ {
14     msg := <-done
15     fmt.Println(msg)
16   }
17 }
```

some tools based on semaphores that do this job reliably.

The most elegant method, preferred by Go programmers, however, uses channels. These communication lines, reminiscent of Unix pipes, transport information from one part of the program to another. In addition, they block the program flow in a routine if nothing can be fed into the channel or read out from it temporarily and are thus ideal for synchronization, because individual program parts can wait for each other.

Channels, Synchronize!

Listing 4 fires off three different goroutines again, but doesn't output anything directly in them. Instead, the goroutines feed their output, which contains data of the `string` type, into a channel named `done` defined in line 5. The inverted arrow `<-` pointing from the data to be written (e.g., `"a"`) to the channel sends the data into the channel (Figure 3).

Depending on the free capacity in the channel, this is done either immediately, or the Go runtime blocks the execution of the respective goroutine until the channel can receive the data. It is very important that while one goroutine might block at any given time, other goroutines in the system continue to run unhindered and thus do not cause a hiccup in the program, but instead empower a high-performance system.

The output from Listing 4 is also nondeterministic, whether you will see *abc* or *cba* or *bac* is uncertain, because the single goroutines in this simple implementation don't coordinate their work with each other, and the main program only waits until all routines are finished – the order thus is rather random. What is guaranteed, though, is that the output will always contain three letters, which is an improvement over the previous race condition.

After firing off the goroutines, the main program uses the `defer` statement in line 11 to stipulate that the `done` channel will be closed after the program terminates; it then enters a `for` loop, which uses the read operator `<-` on the left (!) side of the channel variable to fetch the next value that exists in the channel in line 14 and assigns it to the `msg` variable.

Synchronization with the previously spawned goroutines also occurs during this read operation. When the main program reaches the `for` loop and the read operation for the first time, it is highly unlikely that any of the goroutines have had an opportunity to execute their write statements thus far. But this doesn't matter; if the channel is still empty, the program blocks in line 14 until data becomes available and the Go runtime lets one of the goroutines loose to perform its task. As soon as the first chunk of data has trickled in, the blocked read statement in the main program also notices that things are happening, obtains the available data, and the `for` loop goes into the next round.

Precise Count

Now it becomes clear why, in this simple implementation, the `for` loop in line 13 needs to know exactly how many data packages were fed into the channel by the goroutines in order to retrieve precisely that number. If the main program were simply to continue to ask the channel for data, the Go runtime would block the fourth read process for an infinite period of time, because no further data would be written to the channel from this point onward. A hanging main program would be the result.

Faster Through Parallelism

With this tool, the web client in Listing 5 now sends the requests to the different Internet pages at the same time and saves time instead of awaiting each request's return and then moving on to the next. It also uses the `httpsimple` package shown in Listing 1 to retrieve the data from the web. The `fetchall()` function as of line 32 starts a separate goroutine for each request; this means that four goroutines are working on retrieving and processing the data, and another one is collecting the results, all at the same time!

The channel through which the worker bees send their results to the main program is defined in line 35, setting the type of the data fed into the channel as the `Result` structure defined in line 8. After the `Get()` function of the `httpsimple` package has returned the text data from the retrieved web page, and the result has been stored in the `body` variable, line 40 inserts it along with any error codes and the URL into the data structure and then writes it into the channel, using the write opera-

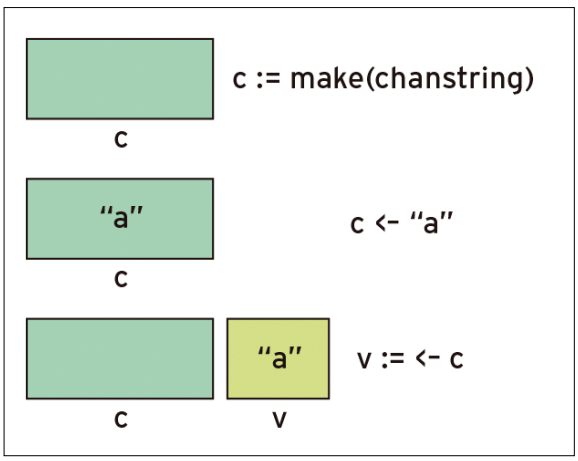


Figure 3: The Go syntax for write and read channel access takes some getting used to.

Listing 5: http-parallel.go

```
01 package main
02
03 import(
04   "fmt"
05   "httpsimple"
06 )
07
08 type Result struct {
09   Error error
10   Body string
11   Url string
12 }
13
14 func main() {
15   urls := []string{
16     "https://google.com",
17     "https://facebook.com",
18     "https://yahoo.com",
19     "https://apple.com"}
20
21   results := fetchall(urls)
22
23   for i := 0; i<len(urls); i++ {
24     result := <-results
25     if result.Error == nil {
26       fmt.Printf("%s: %d bytes\n",
27         result.Url, len(result.Body))
28     }
29   }
30 }
31
32 func fetchall(
33   urls []string) (<-chan Result) {
34
35   results := make(chan Result)
36
37   for _, url := range urls {
38     go func(url string) {
39       body, err := httpsimple.Get(url)
40       results <- Result{
41         Error: err, Body: body, Url: url}
42     }(url)
43   }
44
45   return results
46 }
```

tor `<-` on the right side of the `results` channel variable.

Beware Pitfalls!

When firing off goroutines in `for` loops, there is one typical newcomer mistake that you will want to avoid [4]. The `go func(){}()` call to an anonymously defined function as a goroutine acts as a closure (i.e., any locally defined variables in the main program are available in the goroutines, even if the variables lose their validity on leaving the current code block.)

But since the `url` loop variable changes its value in each new pass of the loop, and most likely none of the goroutines will start running before the loop ends, programmers will find themselves faced with the strange phenomenon that each of the goroutines is given the same value for `url`, usually the last element of the array in the loop. To prevent this from happening and to make sure that each goroutine gets its own `url` value, the loop body in Listing 5 adds the `url` parameter to the argument list of the anonymous function in line 38, while line 42 passes it into the function as an argument.

The main program iterates as of line 23 over a fixed number of channel entries. Thankfully, the number is defined by the length of the `urls` array in line 15. The channel read operator can then simply block in line 24 until the next result is available in the channel, since the parallel goroutines will store exactly the specified number of results in the channel.

Figure 4 shows that parallel data collection indeed saves a good deal of time; the program completes the process about three times as fast. It is undoubtedly more efficient to keep the computer busy with other tasks while waiting for web data than to sit around, twiddling its tiny thumbs.

I highly recommend the book by Katherine Cox-Buday on the subject of concurrency with Go [5]. It meticulously walks the reader through good and bad design with Go channels, and it not only shows the common design patterns, but also looks behind the scenes and explains why a certain approach will produce faster and less error-prone programs.

```
$ time ./http-parallel
https://apple.com: 65926 bytes
https://google.com: 11425 bytes
https://facebook.com: 580876 bytes
https://yahoo.com: 483393 bytes

real    0m0.730s
user    0m0.112s
sys     0m0.016s
$ _
```

Figure 4: If goroutines fire off requests simultaneously, the program is three times faster.

The speed increase does not come as a free gift with parallelization. If you don't pay meticulous attention, you might end up scratching your head and wondering why you have race conditions, deadlocks, or other mysterious panic attacks of the program on production systems under load. Consequently, careful design is important. ■■■

Info

[1] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/219/*

[2] "Don't use Go's default HTTP client (in production)" by Nathan Smith: *https://medium.com/@nate510/don-t-use-go-s-default-http-client-4804cb19f779*

[3] "Tower of Babylon" by Michael Schilli, *Linux Magazine*, issue 201, August, 2017, pp. 60-62: *http://www.linux-magazine.com/Issues/2017/201/Programming-Snapshot-Multilingual-Programming/(language)/eng-US*

[4] "Closure mistake with for loops": *https://github.com/golang/go/wiki/CommonMistakes*

[5] Cox-Buday, Katherine. *Concurrency in Go*. O'Reilly, 2017

GRUB 2 Passwords and Encryption

IN-DEPTH SECURITY

More than just a boot manager, GRUB 2 can help you add another line of protection to your security defenses. *By Bruce Byfield*

A boot manager is almost as much of the Linux tradition as compiling a custom kernel. Traditionally, a boot manager has been used for choosing a kernel to start and for running multiple operating systems on a single computer. However, at a time when everybody is becoming security conscious, few are aware that GRUB 2, the most popular boot manager, is also capable of using passwords and encryption to provide another level of security [1]. Admittedly, GRUB 2 security is not enough by itself, but it is still worth adding to your in-depth defenses.

GRUB 2 has existed for well over a decade and is rapidly replacing GRUB Legacy, the original version of the boot manager, especially in major distributions. As a result, its basic operation and traditional uses are reasonably well-known. However, before I dive into setting up passwords and encryption, a quick overview is useful, both as a reminder and as an introduction for those who might be still using GRUB Legacy or another boot manager, like the now discontinued LILO.

GRUB 2 has configuration files in several places. The first is the `/boot/grub/` directory, which contains `grub.cfg`, the main configuration file. However, unlike GRUB Legacy, the main configuration file is not edited directly. Neither are the config files for each menu item that are stored in `/boot`. Instead, GRUB 2 is updated automatically when a kernel is added or deleted from the system or when the user runs the command `update-grub`, which creates the menu list of available kernels and operating systems. Resources such as the background image are also generally stored in `/boot/grub/`, although they can be stored in another path.

Setting GRUB Display Options and Behavior

The first GRUB 2 configuration file that is directly edited is `/etc/default/grub` [2]. This file sets display and performance settings (Table 1). Typically, these options consist of a human-readable value, each of which is edited by either uncommenting the option or changing the value. You might, for example, change the value `GRUB_TIMEOUT` from its default of five seconds on a Debian system to 20 seconds if you had a long list of different kernels that a user needs to read through before choosing one. The file is heavily commented, but full instructions for editing `/etc/default/grub` can be had by running the command:

```
info -f grub -n 'Simple configuration'
```

No man file is available. GNU projects like GRUB often prefer to use `info` instead. The third source of configuration information are the files in the `/etc/grub.d/` directory. Each file in `/grub.d` is an executable file, whose name indicates the order in which it is run at bootup. For example, Table 2 shows the GRUB 2 configuration files commonly found in Linux. Most of these files are created automatically as you install Linux and only require editing if you want additional refinements, such as passwords or encryption. An especially important section

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*

Table 1: Selected Entries in /etc/default/grub

GRUB_DEFAULT	Sets the default menu item to boot.
GRUB_TIMEOUT	Sets the time before GRUB 2 boots; default if no choice is made or no key is pressed to show menu.
GRUB_HIDDEN_TIMEOUT	Shows how long before GRUB 2 boots when no menu displays.
GRUB_HIDDEN_TIMEOUT_QUIET	Suppresses countdown when no menu displays.
GRUB_DISTRIBUTOR	Shows the variant of GRUB 2 used.
GRUB_BADRAM	Prevents GRUB 2 from using designated bad RAM.
GRUB_TERMINAL	Disables graphical display.
GRUB_GFXMODE	Sets resolution for GRUB 2.
GRUB_INIT_TUNE	Beeps when boot begins.
GRUB_BACKGROUND	Show the path to the splash screen to use with GRUB 2.

Table 2: Common Files in /etc/grub.d

Third-party apps and custom apps may vary with the distribution.	
00_*	Linux headers
00_header	Sets environmental variables, such as system file locations and video settings
05_debian_theme	Sets the theme for the menu display and the splash screen behind it
10_*	Boot entry headings for distribution
10_linux	Identifies the Linux kernel
20_*	Third-party apps
20_memtest86+	Displays option for /boot/memtest86+.bin if it is present
20_linux_xen	Must use for interaction with Xen virtualization
30_uefi-firmware	Sets variables needed to run with UEFI
30_os-prober	Searches for Linux and Windows operating systems if os-prober is installed
40_custom*	User-generated scripts
40_custom	Provides a template for adding other custom menu entries
41_custom	Custom menu entries

comprises the 40_custom files, which are designed for your own entries. These custom files are useful for restoring a system from a recovery disk, although that is a subject outside the scope of this article.

Each time you finish editing /etc/default/grub or any file in /etc/grub.d, or you make changes in both locations, they only take effect after you run update-grub as root. Running this command rebuilds /boot/grub/grub.cfg, so be sure your changes are valid and typo-free before running update-grub. In fact, backing up grub.cfg will reduce your recovery time if the worst happens.

Other scripts for modifying GRUB 2 also exist (e.g., grub-mkfont) that do much of the work for you. However, I emphasize customizing using a text editor, because that is what setting passwords and using encryption requires, and it gives users a chance to learn the application in depth.

Setting Up Password Support

GRUB 2 supports passwords for the entire menu, the type of operating system, and individual menu items. By itself, the password support does not provide comprehensive security, since by default, all passwords are stored in plain text and can be bypassed by booting from a security disk.

To set up passwords, you must have os-prober installed on your system. Three files need to be edited as root: /etc/grub.d/00_header*, /etc/grub.d/10_linux, and /etc/grub.d/30_os-prober. Back up all three anywhere outside /etc/grub.d, so you can easily recover from any problems. Do not place the backups in /grub.d, or GRUB 2 may overwrite them. If you have the expertise and need a reference, you can find a sample file online [3].

The setup for passwords requires four steps:

1. Add a root user and password. This root user can access all menu items.

Technically, the information can be added to any of these three files, but usually it is placed in /etc/grub.d/00_header. Scroll all the way down to the bottom of the file and add lines with the following structure:

```
cat <<EOF
set superusers="USER"
password USER PASSWORD
export superusers
EOF
```

2. Add other users. You will probably give all other users on the system a password for each menu item, but first you need to make GRUB aware of each user. Use the structure password USER PASSWORD, adding one user per line below the password line for the root user.
3. Once the users are defined, decide which menu items to password protect. Any user will be able to select unprotected menu items, and the root user can select any items, entering a password to select protected ones. Other users must be specifically permitted to open protected menu items. You can set up menu items for using passwords by opening /etc/grub.d/10_linux and finding the line:

```
printf "menuentry '${title}' ➥
  ${CLASS} {\n" "${os}" "${version}"
```

Add --users '' after ${CLASS}, so that the line reads:

```
printf "menuentry '${title}' ➥
        ${CLASS} --users '' ➥
        {\n" "${os}" "${version}"
```

Note that --users is followed by two single quotation marks, not a double one.

4. If necessary, create an /etc/grub.d/30_os-prober file, using online examples. Then, to add password protection to all entries, run:

```
sed 's/--class os /--class os --users ➥
  /' -i /etc/grub.d/30_os-prober
```

Alternatively, you can set passwords for a certain type of operating system by adding --users before the last curly bracket on the line. For instance, for Linux, the edited line should read:

```
menuentry "${LLABEL} (on ${DEVICE})" ➥
  --class gnu-linux --class gnu ➥
  --class os --users {
```

While for Windows, the edited line would be:

```
menuentry "${LONGNAME} ➥
  (on ${DEVICE})" --class windows ➥
  --class os {
```

Should you want to password protect a particular partition that has an operating system on it, find in `/etc/grub/330_osprober` the lines:

```
cat << EOF
menuentry "${LONGNAME} ➥
  (on ${DEVICE})" --class windows ➥
  --class os {
EOF
```

Edit them to read as shown in Listing 1. Replace `DEVICE` in line 1 with the name of the partition (e.g., `/dev/sd5`).

Save each of the edited scripts and run `grub-update` to enable the password protection. At the login screen, clicking a menu item results in a pop-up box for entering the user name and password.

Encrypting Passwords

Encryption greatly enhances the effectiveness of GRUB 2 passwords. However, somewhat arbitrarily, GRUB 2 encryption depends on a utility called `grub-mkpasswd-pbkdf2` as much as on the manual editing of a file. `grub-mkpasswd-pbkdf2` is included with GRUB 2 when it is installed, but when you try it for the first time, you should probably keep at least one menu item unprotected and unencrypted, at least until you are certain that you have the setup right (Figure 1).

`grub-mkpasswd-pbkdf2` is easy to use. Rather than editing manually, set up passwords and then run the command as root and generate the encryption hash by entering a user's password twice. By default, the result is a hash of several hundred characters, but you can increase the length of the hash – and the resulting strength of encryption – by increasing the number of iterations with the `c=NUMBER` option and the amount of salt (random data) with the option `-s=NUMBER`. You can also use `-l` to increase the length of the hash.

Create the password and then copy and paste it into `/etc/grub.d/00_header` so that each password line has the format:

```
password_pbkdf2 USER ENCRYPTED-PASSWORD
```

The password will be stored in encrypted form, but users will type in the unencrypted form. Although a boot disk will still be able to boot into the system, the result will strengthen GRUB 2 passwords in general. However, until `grub-mkpasswd-pbkdf` has been tested more, use it cautiously.

Listing 1: Edited /etc/grub/330_osprober File

```
01 if [ ${DEVICE} = "/dev/sdXY" ]; then
02 cat << EOF
03 menuentry "${LONGNAME} (on ${DEVICE})" --users "" {
04 EOF
05
06 else
07 cat << EOF
08 menuentry "${LONGNAME} (on ${DEVICE})"
09 EOF
10 fi
```

Defense in Depth

Since GRUB 2 passwords can be so easily circumvented, you might wonder if they are worth setting up, especially when one mistake can lock you out of your system and require awkward recovery time. It is true that depending only on GRUB 2's own security would provide weak protection. However, a basic principle of security is defense in depth.

Simply put, defense in depth means setting up as many security measures as possible. If one measure fails to stop an intruder, another one will. Moreover, if a security measure requires a time-consuming workaround, then an intruder has to be strongly motivated to persist, especially if there is a chance that other measures also have to be circumvented. In other words, some defenses are simply not worth the time to penetrate.

I would put GRUB 2's passwords and encryption into this second category. Their value lies less in absolute security than in their nuisance value for intruders and their role as only one of a number of defenses. If you doubt that, make a deliberate mistake in your GRUB 2 configuration and try to recover from it. Even if you know exactly what to do, you may still resent the loss of time. At that moment, you will understand why even relatively weak security can still be part of your defenses.

However, if you want truly strong encryption, encrypt the `/boot` partition during installation; then, set up GRUB 2 following the concise instructions on the Debian wiki [4]. The instructions assume a higher degree of expertise than is required to edit GRUB 2 files, which is why I have not detailed them here. ■■■

Info

[1] GRUB 2: *https://www.gnu.org/software/grub/*

[2] /etc/default/grub fields: *https://help.ubuntu.com/community/Grub2/Setup#User_Settings:_.2Fetc.2Fdefault.2Fgrub*

[3] Sample /etc/grub.d files: *https://www.apt-browse.org/browse/ubuntu/trusty/main/i386/grub-common/2.02~beta2-9/file/etc/grub.d/*

[4] GRUB 2 and encrypted boot: *https://wiki.debian.org/Grub2#Configure_encrypted_.2Fboot*

```
root@nanday:~# grub-mkpasswd-pbkdf2
Enter password:
Reenter password:
PBKDF2 hash of your password is grub.pbkdf2.sha512.10000.C5B2CE33EEC0A4C83C738CE578E96
21072A0CC6C183DDFD48E98898489DBEA7760870CBD88DC226B014ABA87B06D12D363B35A98DBA8498B5EF
29366AAC74D33.0AA6A1DD8B2E72EA41D095CD758012B53990547EAB160215FB35AE80B3F55BC1A56A0861
7C71A7EB4F9078D3D2956F923E7D854691D41EBDF7A3F37C8291679E
```

Figure 1: `grub-mkpasswd-pbkdf2` is a simple tool for creating an encrypted hash for a GRUB 2 password.

The sys admin's daily grind: The moreutils collection

Hot Stuff!

This month, sys admin columnist Charly dumps the *moreutils* toolbox on his workbench and takes *combine* and *vidir* for a spin. *By Charly Kühnast*

In the March 2013 issue, I wrote about `ifdata`, which gives you information about networks in a way that is great for scripting. I had taken the tool from the *moreutils* package at the time; this toolbox comes with almost every flavor of Linux, and, if not, you can download it [1]. Further investigation of the package reveals even more laser-sharp tools. For instance, `combine` is really practical for comparing stuff. You need to pass in the names of two text files and a logical operator: `and`, `not`, `or`, or `xor` (exclusive or).

As an example, I created two text files with IP addresses and networks. Some of the addresses and networks are included in both files, while others are not. Now I let `combine` compare the files, first with the `and` operator:

```
combine iplist-a.text and iplist-b.text
```

In Figure 1, you can see the content of the two files, and then the `combine` command line. The output I got was all the lines that occur in both files.

When I executed the command with the `or` operator, I could see everything that occurred in one or both the files. Important: If a line appears in both files, it also appears twice. This is often undesirable, but can be suppressed with:

```
combine iplist-a.text ➥
  or iplist-b.text ➥
  | sort | uniq
```

One thing is probably already clear to most readers: The same results are possible with `cat`, but many roads lead to Rome. Using the `not` operator, I can output all the lines that occur in the first file, but not in the second. The `xor` operator tells `combine` to return the lines that are only found in one file, but not in both.

```
charly@gw:~$ cat iplist-a.text
10.50.1.0/24
172.16.30.1/32
192.168.60.0/24
10.16.3.0/24
192.168.1.254/32
charly@gw:~$ cat iplist-b.text
172.16.1.0/24
192.168.60.0/24
10.16.3.0/24
10.90.0.0/16
192.168.128.0/24
charly@gw:~$
charly@gw:~$ combine iplist-a.text and iplist-b.text
192.168.60.0/24
10.16.3.0/24
charly@gw:~$ combine iplist-a.text or iplist-b.text
10.50.1.0/24
172.16.30.1/32
192.168.60.0/24
10.16.3.0/24
192.168.1.254/32
172.16.1.0/24
192.168.60.0/24
10.16.3.0/24
10.90.0.0/16
192.168.128.0/24
charly@gw:~$ combine iplist-a.text not iplist-b.text
10.50.1.0/24
172.16.30.1/32
192.168.1.254/32
charly@gw:~$ combine iplist-a.text xor iplist-b.text
10.50.1.0/24
172.16.30.1/32
192.168.1.254/32
172.16.1.0/24
10.90.0.0/16
192.168.128.0/24
charly@gw:~$
```

Figure 1: The contents of two sample files at the top. `combine` compares them in different ways.

Sly List

Another workhorse from the *moreutils* package goes by the name of `vidir`. I don't use it often, but if I do, it saves a huge amount of typing. (Praise be to anything that contributes to my laziness.) `vidir` in particular makes it easier to rename files. Normally, I do this with:

```
mv <File_1> <File_2>
```

```
1       ./iplist-1.text
2       ./iplist-2.text
~
~
                2,17-23     All
```

Figure 2: `vidir` fetches a file list and presents it in the editor.

The Perl `rename` tool does this more conveniently and can process multiple files at the same time. But `vidir` has a special trick up its sleeve. When I run it in the current directory, it opens Vi (or whatever `$EDITOR` defines) and displays a list of the files present in the directory. Now I can edit the file names to suit my needs. In Figure 2, I renamed the `iplist` files from the earlier example. When I leave the editor and run `ls`, hey presto, I find the files have been renamed.

This excursion has by no means exhausted the *moreutils* box – in fact, it comprises 17 tools. Some only do tiny jobs (e.g., `isutf8` checks whether a file is valid UTF-8); others are more extensive – some great DIY stuff included! ■■■

Info

[1] moreutils: *https://joeyh.name/code/moreutils/*

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.


LINUX MAGAZINE
INTERNET OF THINGS
Tools for home automation
Bluetooth Tricks
Track movements within your house
Linux on Old Hardware
Tablets on Linux
Configure graphics settings with xsetwacom
Raspberry Pi 3 B+
Does the world need a new Rasp Pi?
Python Gambling
Simulating games of chance
Logism
Design and draw digital circuits
PIRATEBOX
Handy anonymous file server for parties and meetups
MYCROFT
An open source personal assistant
MYCROFT
Can an open source personal assistant compete with Alexa?
Scanners and Linux
Peppermint OS
Peppy distro built for the cloud
Meltdown & Spectre
Kernel developers take swift action to protect Linux
Knight's Tour
Classic chess puzzle gets a Python twist
Node-RED
Control a Rasp Pi using text messages
LINUXVOICE
FOSSPicks
UBUNTU 17.10
GNOME IS BACK IN THE GARDEN
INSIDE A COMPILER
INSIDE A COMPILER
What really happens to that code
C++ CODERS
What's New in LibreOffice 5.4
Brown Dog Gadgets
More fun with science
Neural Network
Can a program play the Monty Hall game?
Catch a Fox!
Monitor wildlife traps with a microcontroller
Guetzli
JPEG-quality images with a tiny file size
GUI Firewall Tools
Airtight security in an easy view
Tools for creating and customizing maps
MAPPING TOOLS
Create and edit digital maps with QGIS and QMapShack
4 Open Source Microblogging Tools
Kubuntu Up Close
Explore the new 18.04 release
MakerSpace
9 Handpicked Pi projects
Caddy
HTTPS server out of the box
ODROID-C2
This single-board system is twice as fast as the competition
Ogg Vorbis
Free format for audio files
A Python Script
That solves the Chinese ring puzzle
4 COLLABORATIVE OFFICE SUITES:
Keep your team in sync
LIBREBOOT and the liberation of firmware
REE BOOT!
boot and the liberation of firmware
Bash Tricks
Manipulating binary data
FEDORA 27
We test drive the newest Fedora
TERMINAL TUNING
RMINAL TUNING
Bash shell
New in Fedora 27
Do Managers
task-management tool
Eelo
This new phone puts your privacy first
BROWSER SHOOTOUT
We compare Firefox, Chrome, Opera, and Vivaldi
USB PASSWORD MANAGER
and lots of other timely and useful tools
Timers
WILL SYSTEMD REPLACE CRON?
FIREFOX BACK?
SAFER BOOT
Keeping control of the startup process
UEFI Tricks
Add a custom app that runs from the firmware
5 Automated Backup Tools
Pi FM Radio
Build a stairway to maker heaven
DDoS Attack
Avoiding a denial of service nightmare
Systemd
Getting more from the elusive new Linux init system
10 Top PDF Readers
Arduino Programming for Open Hardware Projects
Homegrown Facial Recognition
Build facial recognition into your own Python scripts
Maker Tricks
Monitor a beehive with a Raspberry Pi
Cover Your Ass(ets)
Back up your system files with CYA
DVD INSIDE! ALL 214 ISSUES ON A SEARCHABLE DISC!
HUGE SAVINGS! $29.90 VALUE!
PRIVACY
STOP SNOOPERS AND PROTECT YOUR IDENTITY
PRIVACY
Stop snoopers and protect your identity
Rockbox
Liberate your music player with free firmware
RSS Readers
View and organize all your favorite news sources
PiXtend
More interfaces for your Rasp Pi
Rasp Pi Sailboat
Ride the winds with the tiny Pi Zero
Ready to Go?
Smart queries in the powerful Go language
WWW.LINUX-MAGAZINE.COM

MakerSpace

Redis in-memory storage

Recollection

Include binary data and image files in your IoT projects with Redis, an open source, in-memory data structure store.

By Pete Metcalfe

For many Internet of Things (IoT) projects, a message-queuing system like MQTT (message-queuing telemetry transport) is all you need to connect sensors, devices, and graphic interfaces. However, if you have requirements for high throughput or you are storing special data types (e.g., binary data or image files), you should take a look at Redis.

Redis (*Re*mote *di*rectory *s*erver) [1] is an open source, in-memory data structure store that can be used as a database, cache, and message broker. It supports a wide range of data structures, such as strings, hashes, lists, sets, bitmaps, HyperLogLogs, and geospatial indexes. Redis servers can be loaded locally, or they are available as web-hosted solutions. Redis libraries are available for a wide variety of programming languages.

In this article, I show you how to set up a Redis system with two examples. The first is a flame scanner that connects an Arduino module to a Node-RED web dashboard. The second example is a Raspberry Pi weather station, in which a webcam image is stored in a Redis server and a PHP web page shows the data.

Getting Started Locally

To install Redis on Ubuntu systems, go to the terminal and enter:

```
$sudo apt-get update
$sudo apt-get install redis-server
```

Once Redis is installed, it can be started by:

```
$redis-server
```

The `redis-cli` command-line tool can be used for monitoring and testing; for example, to assign and read a value:

Listing 1: redis-benchmark

```
$ redis-benchmark -c 10 -t SET,GET
====== SET ======
100000 requests completed in 18.31 seconds
10 parallel clients
3 bytes payload
...
5460.60 requests per second

====== GET ======
100000 requests completed in 16.96 seconds
10 parallel clients
3 bytes payload
...
5895.53 requests per second
```

Author

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

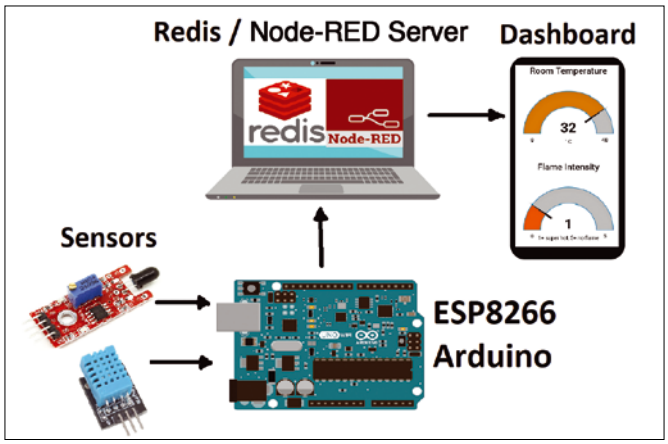


Figure 1: Arduino/Redis flame scanner diagram.

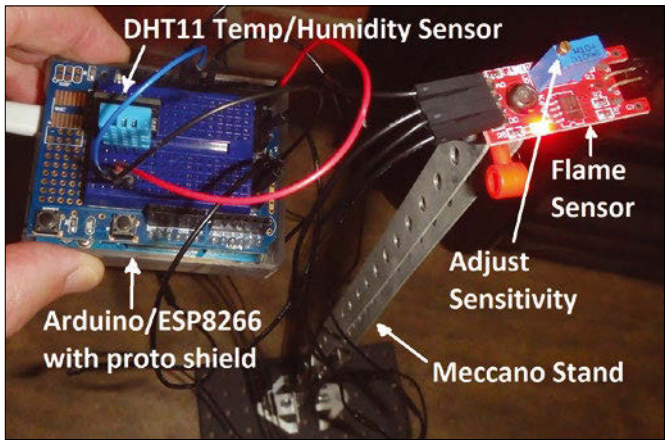


Figure 2: Arduino/Redis flame scanner setup.

```
$redis-cli
127.0.0.1:6379> set mytag1 12.3
OK
127.0.0.1:6379> get mytag1
"12.3"
```

The default Redis security is set to local access only. To open the server to remote connections, edit the `/etc/redis/redis.conf` file and either comment out all the `bind` statements (e.g., `#bind 127.0.0.1`) or add `bind` statements for all your valid IP addresses. After editing the config file, you need to restart the Redis server by entering the

```
sudo service redis-server restart
```

command.

Redis Performance

The Redis in-memory data store boasts some incredible throughput numbers. I wanted to test this out, so I loaded Redis on an old Acer Aspire One laptop with a 1.6GHz N270 Intel Atom processor and 4GB of RAM. A Redis tool called `redis-benchmark` allows me to do some up-front performance analysis configured for different messages, client counts, and test durations.

For my IoT project, I wanted to check whether my low-end Ubuntu laptop could handle 10 clients. My maximum throughput for GET and SET messages can be estimated by the `redis-benchmark` command shown in Listing 1.

Although I have many performance factors to consider, for my IoT system with 10 Arduino/Pi modules, I probably won't generate more than 50 SETs/sec. According to the test, my old Ubuntu laptop appears to be able to do 5,000+ SETs/sec, so I shouldn't have any problems using it for my IoT system.

Flame Scanner Example

My goal for the flame scanner example is to monitor my fireplace remotely with an ESP8266-compatible Arduino module and a Redis server to store the data (Figure 1). A lot of IoT dashboard options are available, but I like to use Node-RED because it is 100% standalone.

For my hardware setup, I mounted a low-cost infrared (IR) flame sensor [2] ($3) on some Meccano pieces, and I aimed it at the center of the fireplace. A DHT11 temperature/humidity sensor [3] ($4) was also connected to the Arduino module to record the room temperature (Figure 2). I did my testing with a Wemos D1 module and a proto shield, but any of the ESP8266-based Arduino modules could be used.

Redis supports a number of storage methods, with the most common methods being SET/GET and PUBLISH/SUBSCRIBE. For this example, I used the PUBLISH/SUBSCRIBE method to make it similar to a messaging system like MQTT.

Arduino Redis Library

The Arduino ESP8266 family of modules can send sensor data to a Redis server. To install the latest Redis version, use the Arduino Library Manager (Figure 3) and the DHT temperature sensor library.

Infrared flame sensors measure IR light generated by a flame, which makes them great for indoor projects; however, for

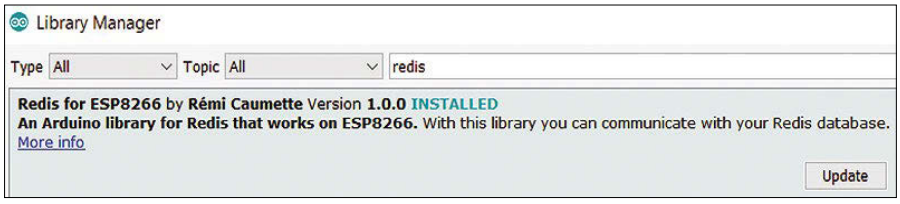


Figure 3: Installing Redis with the Arduino Library Manager.

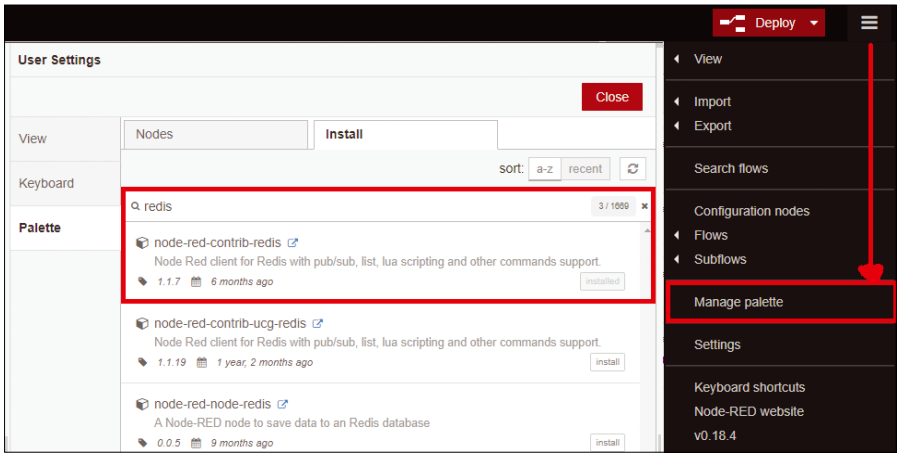


Figure 4: Installing Redis on Node-RED.

outdoor projects, you will need to add some shielding against ambient light. The flame scanner has both an analog (A0) and a digital (D0) connection. The analog A0 pin returns the flame intensity, where *1024* = no flame and *0* = hot flame. The digital D0 connection returns a *1* for no flame and *0* for a flame.

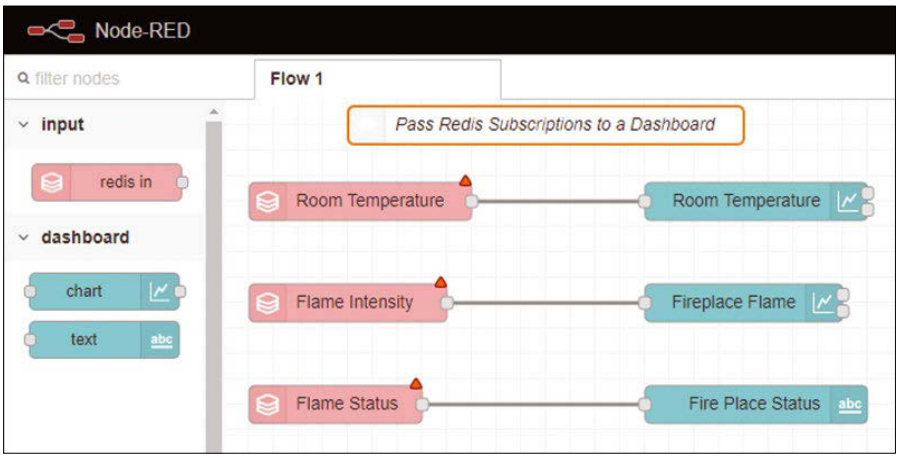


Figure 5: Node-RED flame scanner logic.

For Arduino code (Listing 2), three values are published to a remote Redis server: `roomtemp`, `flamestatus`, and `flameintensity`.

The `redis-cli psubscribe` command can verify that the Arduino values are being received at the Redis server. For example, Listing 3 shows how to subscribe to the `roomtemp` value.

Node-RED and Redis

Node-RED [4] is a visual programming environment that lets you create applications by dragging and dropping nodes onto a code flow window. Logic flows are then created by connecting the different nodes together.

Node-RED has been preinstalled on Raspbian Jesse since the November 2015 release, but Node-RED can be installed

Listing 2: Redis_flame.ino

```
01 #include <Redis.h>
02 #include "DHT.h"
03
04 #define DHTPIN 12 // Signal pin on DHT11
05 #define DHTTYPE DHT11   // DHT 11 is used
06 DHT dht(DHTPIN, DHTTYPE);
07
08 const int flameAiPin = 13; //AO pin on flame scanner
09 const int flameDiPin = 14; //DO pin on flame scanner
10
11 #define WIFI_SSID       "your SSID"
12 #define WIFI_PASSWORD   "your Password"
13
14 #define REDIS_ADDR      "192.168.0.121" // your Redis IP
15 #define REDIS_PORT      6379
16 #define REDIS_PASSWORD  ""
17
18 Redis redis(REDIS_ADDR, REDIS_PORT);
19
20 void setup()
21 {
22     Serial.begin(9600);
23     Serial.println();
24
25     WiFi.mode(WIFI_STA);
26     WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
27     Serial.print("Connecting to the WiFi");
28     while (WiFi.status() != WL_CONNECTED)
29     {
30         delay(250);
31         Serial.print(".");
32     }
33     Serial.println();
34     Serial.print("IP Address: ");
35     Serial.println(WiFi.localIP());
36
37     if (redis.begin(REDIS_PASSWORD))
38     {
39         Serial.println("Connected to the Redis server!");
40     }
41     else
42     {
43         Serial.println("Failed to connect to the Redis
                        server!");
44         return;
45     }
46 }
47 void loop()
48 {
49   char themsg[10];
50   // Get and publish the room temperature
51   sprintf(themsg,"%d",int(dht.readTemperature()));
52   redis.publish("roomtemp", themsg);
53
54   // Get and publish the flame intensity
55   sprintf(themsg,"%d",analogRead(flameAiPin));
56   redis.publish("flameintensity", themsg);
57
58   // Get and publish the flame status
59   if (digitalRead(flameDiPin) == 0) {
60     redis.publish("flamestatus", "FLAME");
61   } else {
62     redis.publish("flamestatus", "NO FLAME");
63   }
64   delay(5000);
65
66 }
```

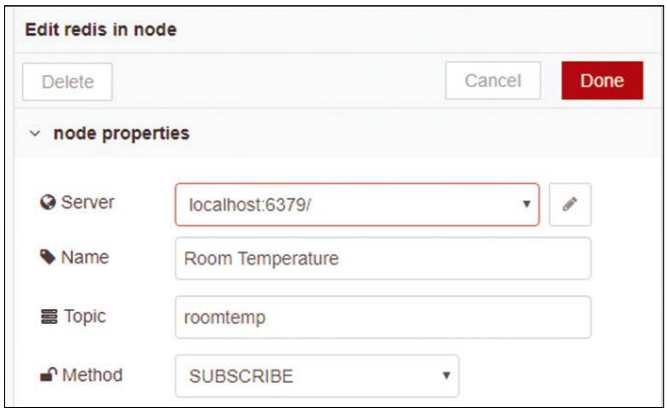


Figure 6: Configuring the Redis server in the redis in dialog.

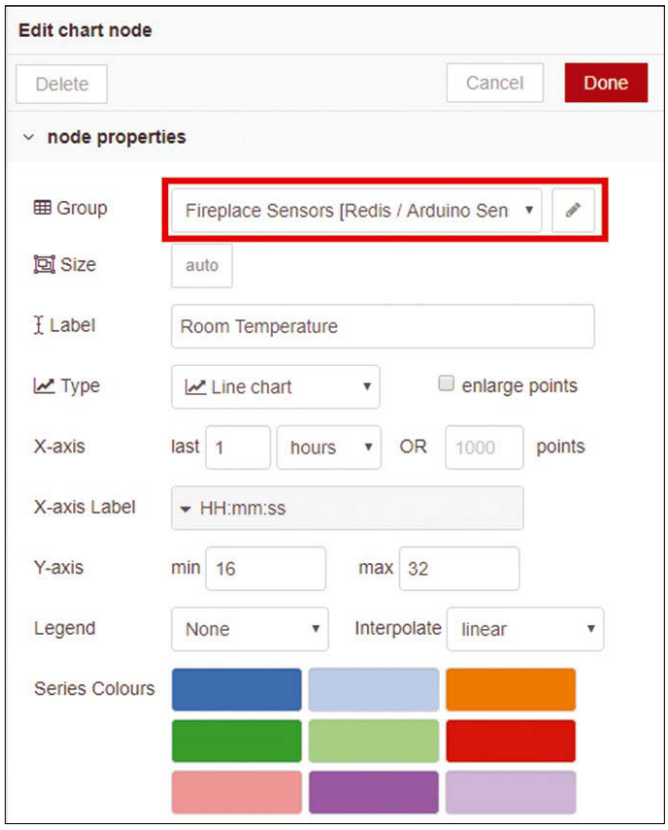


Figure 7: Creating and defining the dashboard in the chart dialog.

on Windows, macOS, and other Linux distros, as well, and you can find instructions online [5] for installing and running Node-RED on your specific system.

To install the Redis components, select the *Manage palette* option from the hamburger menu on the right (Figure 4), then search for *redis* and install *node-red-contrib-redis*. If you haven't already installed the dashboard library, then search for and install *node-red-dashboard*.

For my Node-RED flame scanner example, I'll look at subscribing to the three items (Figure 5) and show them on a web dashboard. The logic will require the following nodes: three `redis in`, two `chart`, and one `text`.

To configure the `redis in` node, double-click on the node to get the edit dialog (Figure 6), so you can configure the Redis server and define the topic to which to subscribe. For this project, I have three topics: *roomtemp*, *flameintensity*, and *flamestatus*. Finally, the method needs to be set to *SUBSCRIBE*.

Next, edit the `chart` (Figure 7) and `text` nodes. The key here is to create and define the dashboard group. Time spans, labels, colors, and sizing are also configured in this dialog box.

After the logic is complete, hit the *Deploy* button on the right side of the menubar to run the logic. The Node-RED dashboard user interface is accessed at *http://<ipaddress>:1880/ui*. Figure 8 shows my Node-RED dashboard with the flame scanner.

Listing 3: Subscribing to a Value

```
$ redis-cli psubscribe roomtemp
Reading messages... (press Ctrl-C to
                     quit)
...
1) "pmessage"
2) "roomtemp"
3) "roomtemp"
4) "21"
```

Weather Station Webcam

A Raspberry Pi can make an excellent weather station node. In addition to sensors (e.g., temperature and humidity), you can include webcams images (Figure 9).

For my test, I used low-cost USB webcams, but you can find some excellent Pi cameras, too. Images are generated with the `fswebcam` command-line tool, which you can install with:

```
sudo apt-get install fswebcam
```

The Python script in Listing 4 calls `fswebcam` and uploads an image to a Redis server. The webcam image is

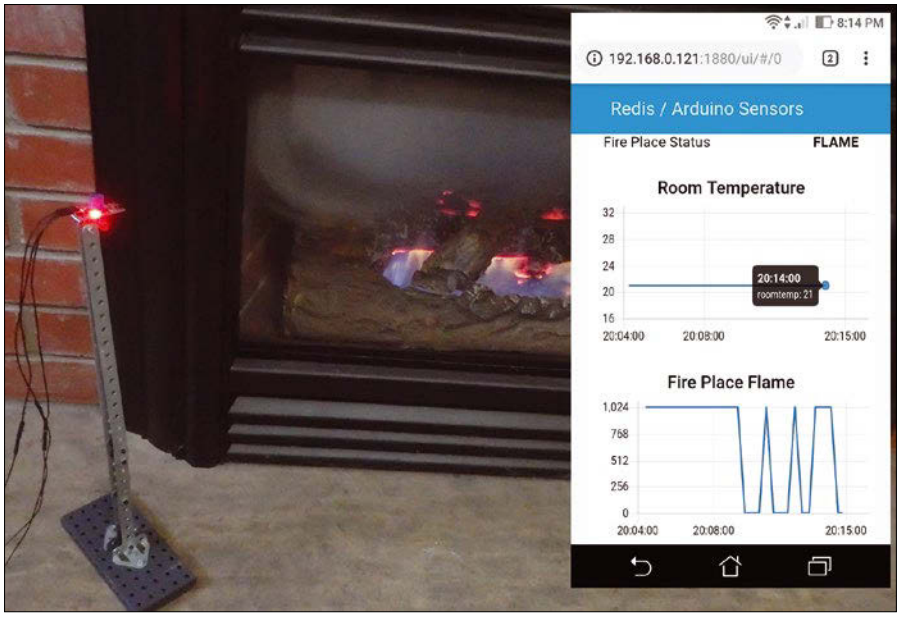


Figure 8: Node-RED flame scanner dashboard.

passed to the Redis server as a large binary variable using the SET method.

For my weather station example, the Python code periodically reads the temperature and humidity sensors and then takes a picture. This data was SET into a Redis server as `temp1`, `humdity1`, and `webcam1`.

I loaded the Redis server on the same hardware as my PHP/Apache web server, which allowed single-line testing with the `redis-cli` tool. Once I was happy with the results, I moved the command-line syntax directly into my PHP script.

To pull the image data from Redis, I issue a `redis-cli` command to GET raw `webcam1` binary data and write it to a JPEG file. Listing 5 is the complete PHP example, and Figure 10 shows a sample weather station web page.

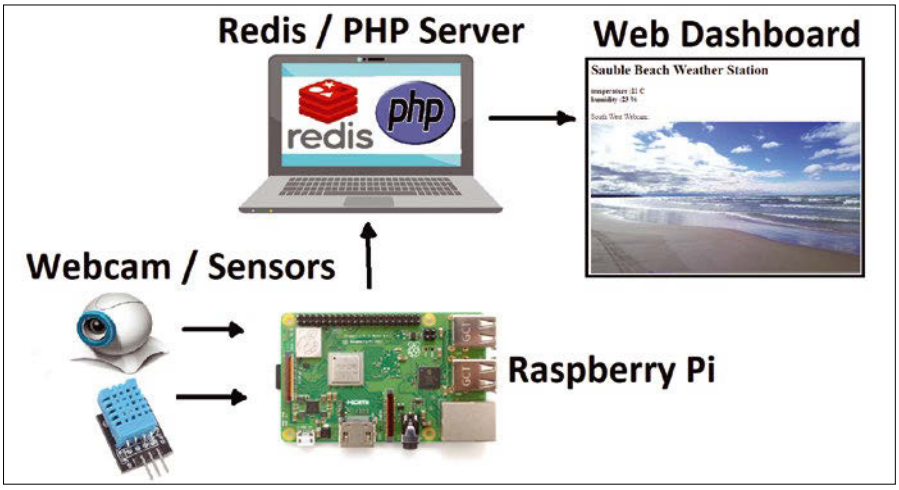


Figure 9: Raspberry Pi/Redis weather station.

Summary

The choices of technology for connecting sensors and devices to your IoT projects are many, but if you're looking for a high-performance storage system or a system that can support binary data, then you should consider taking a look at Redis.

My only comment would be that because Redis is an in-memory data storage system, you should run some up-front calculations on your image storage requirements to ensure that the Redis memory size allocation has been configured correctly. ■■■

Listing 4: image_2_redis.py

```
01 import os
02 import redis
03
04 # have webcam take a picture, save as image1
05 os.system ('fswebcam --no-banner -r 640x480 image1.jpg')
06
07 # connect to your redis hostname
08 r = redis.Redis('localhost')
09
10 img = open("image1.jpg","rb").read()
11 # save image file into redis, with reference webcam1
12 r.set("webcam1",img)
```

Listing 5: redis1.php

```
01 <!DOCTYPE html>
02 <html>
03 <head><title>Redis IoT Data</title></head>
04 <body>
05 <h1>Sauble Beach Weather Station</h1>
06 <?php
07 //  get the redis image and save it to a local file
08 echo exec ("redis-cli -h localhost --raw get 'webcam1' >
   /var/www/html/webcam1.jpg");
09 echo "<b>temperature :";
10 echo exec('redis-cli get "temp1" ');
11 echo "<br>humidity :";
12 echo exec('redis-cli get "humidity1" ');
13 echo "</b>";
14 ?>
15 <br><br>South West Webcam:<br>
16 <img src='webcam1.jpg' width="100%">
17 </body>
18 </html>
```

Info

[1] Redis: *https://redis.io*

[2] Arduino flame detection sensor: *https://www.dx.com/p/arduino-flame-detection-sensor-module-135038*

[3] DHT11 temperature/humidity sensor: *https://www.dx.com/s/dht11*

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

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

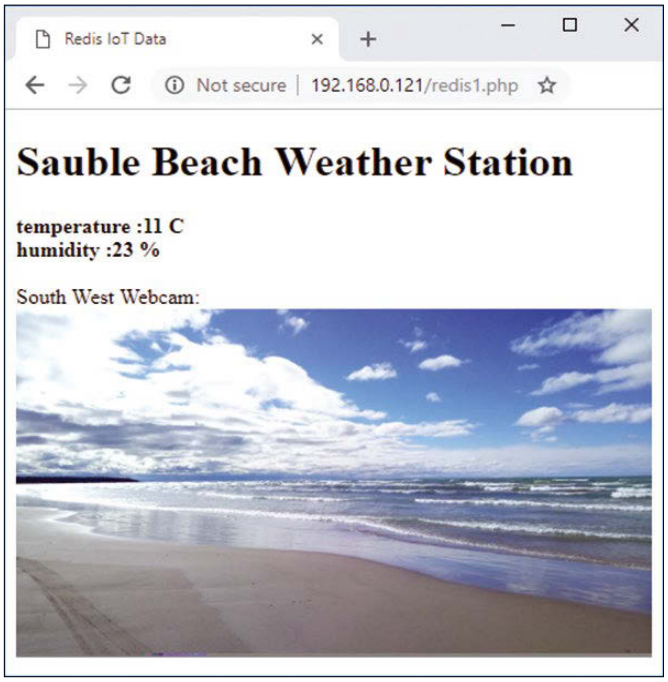
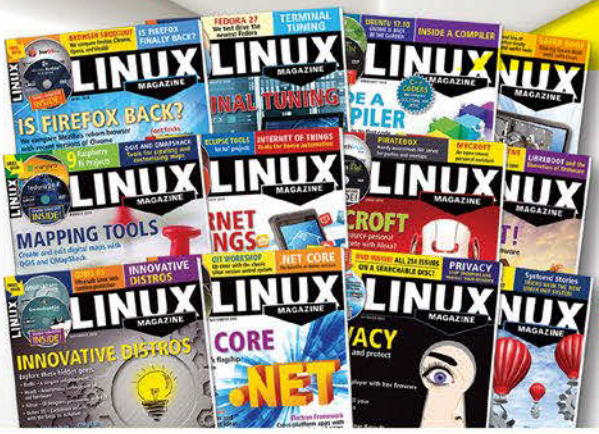


Figure 10: PHP IoT weather station page.

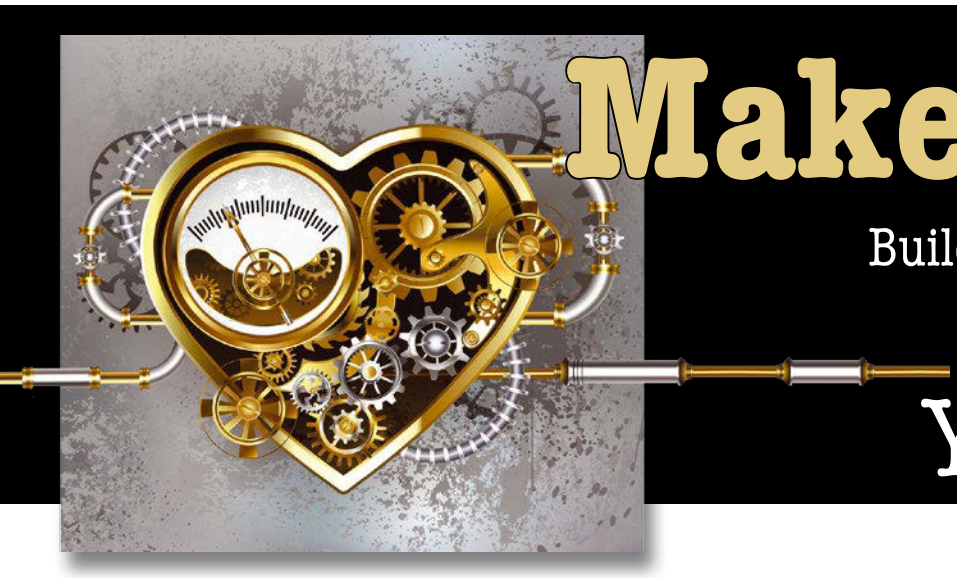
COMPLETE YOUR LIBRARY

Order a digital archive bundle and save at least 50% off the digisub rate!

ORDER YOURS TODAY!
shop.linuxnewmedia.com



You get an **entire year of your favorite magazines** in PDF format that you can access at any time from any device!



MakerSpace

Build your own pulsemeter

Close to Your Heart

A pulsemeter built with a Raspberry Pi, a digital-to-analog converter, and an optical sensor monitors your heart rate just as well as many far more expensive medical devices. *By Martin Mohr*

Monitoring your pulse helps you determine the health of your vascular system. If you are interested in fitness, it is one of the vital metrics in sports monitoring. Whether you are taking measurements out of scientific interest, as a metric to gauge the intensity of your workout, as a measure of fitness, or as a cautionary measure against excessive stress, a DIY pulsemeter provides quick and easy feedback.

The module used in the example here is based on an optical method that measures light absorption of the red blood cells in your veins to determine the rhythm of your heartbeat. Typically, you will want to deploy sensors like this at the wrist, the arch of the foot, temple, or neck, where the veins are just under the skin.

The sensor has an operating voltage of 3-5V, and the output voltage varies depending on how well you position the sensor. Occasionally, it will not output a value at all (e.g., if you press it against the skin so hard that it blocks the blood flow).

While taking a measurement, you should keep the sensor in the same position. The pulse sensor [1] uses green light, which is best suited for measuring at the wrist. Modules with red light, on the other hand, are used for fingertip measurements. The accuracy of the optical sensors is almost equivalent to that of an ECG.

Test Setup

Before setting up the test, you should note that the sensor provides analog values, whereas the Raspberry Pi only processes digital values. Therefore, you need an analog-to-digital converter (ADC). My choice was the MCP3008 [2], because its 10-bit resolution measures with sufficient accuracy.

The schematic (Figure 1) for the test setup is quite clear: It comprises only the MCP3008 ADC, which you connect to the Raspberry Pi through the serial peripheral interface (SPI; specifically, pins 19, 21, 23, and 24), and the sensor, which is connected to the MCP3008 analog input (CH0).

In my tests, I used two different oscilloscopes to monitor the analog values provided by the sensor, so I could get a feel for the pressure needed to get correct readings. In one test, I connected a BitScope Micro [3] to my desktop PC via USB, with the BitScope input channel connected to the *S* output of the pulse sensor.

The output is shown in the BitScope Meter software [4] (Figure 2). The 1.4Hz output frequency corresponds to a pulse of 84 beats per minute. The hertz unit (Hz) indicates the measured oscillations

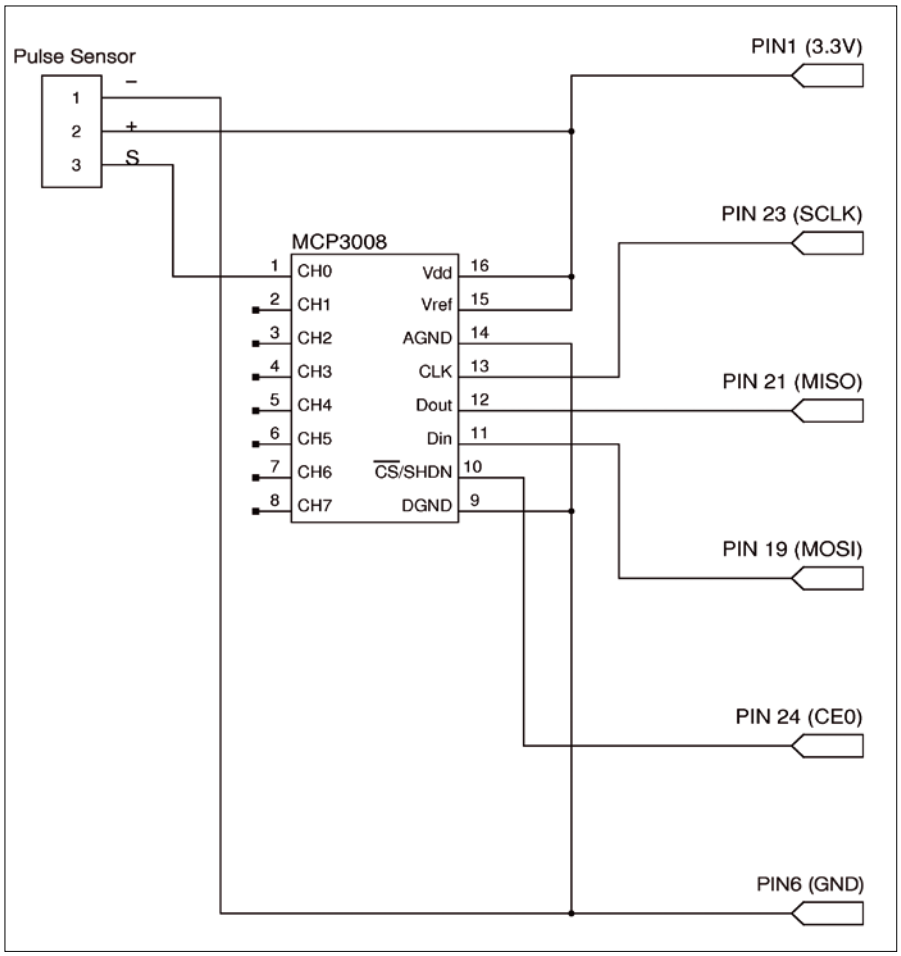


Figure 1: The heart rate monitor requires two components in addition to the Raspberry Pi: an ADC and an optical heart sensor.

per second; multiply this value by 60 to get the oscillations per minute.

BitScope is an interesting mini-oscilloscope for your PC. If you don't have a BitScope, though, you can use any oscilloscope (in Figure 3, I use a DSO138 kit [5]) to monitor the signal of the sensor.

Software

A current Raspbian Stretch Lite [6] provides the underpinnings of this project. After downloading, write it to an SD card with a tool of your choice. After booting, start the `raspi-config` tool and activate the Rasp Pi's SPI under *5 Interfacing Options | P4 SPI*. This setting is required to control the MCP3008.

Additionally, enable the SSH service under *5 Interfacing Options | P2 SSH*. Once running, you will be able to manage the Raspberry Pi with PuTTY [7] or from a desktop terminal.

Last but not least, you will want change the password for the `pi` account, either in `raspi-config` or with the `passwd` command. Running the commands in Listing 1 updates the software and installs all the programs you need to control the sensor. To ensure that you have loaded all components correctly, reboot the computer and launch the sample

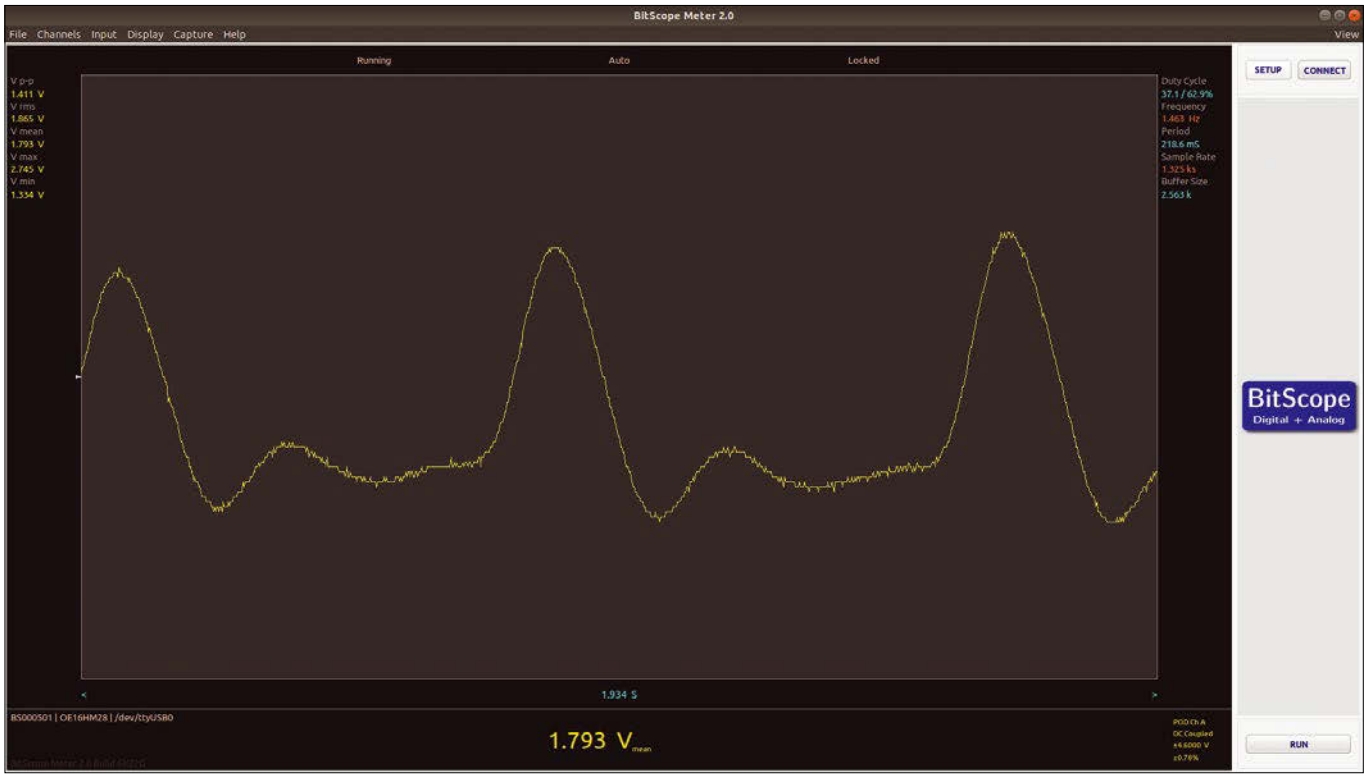


Figure 2: The BitScope Meter software displays the pulse measured by the sensor in hertz. A simple multiplication yields the heart rate.

Listing 1: Install and Update Software

```
sudo apt update
sudo apt upgrade
sudo apt install git python-dev

git clone https://github.com/doceme/py-spidev.git

cd py-spidev/

sudo python setup.py install
cd ..

git clone https://github.com/tutRPi/Raspberry-Pi-Heartbeat-Pulse-Sensor
```

Listing 2: Sample Program

```
$ cd Raspberry Pi Heartbeat Pulse Sensor
$ python example.py
No Heartbeat found
BPM: 80
BPM: 80
BPM: 83
BPM: 85
```

program (Listing 2), which is available on GitHub [8].

Conclusions

The pulse sensor presented here works quite accurately. Initially, you might need some time to figure out the pressure needed at the sensor to get clean readings. A small oscilloscope helps to get the desired result. ■■■

Author

Martin Mohr developed a preference for everything shiny in his early youth. After training as an electronics engineer and studying computer science, he concentrated on programming Java applications. With the advent of the Raspberry Pi, the old love for electronics awakened again.

Info

[1] Pulse sensor: *https://www.sparkfun.com/products/11574*

[2] MCP3008: *https://www.adafruit.com/product/856*

[3] BitScope Micro: *https://www.bitscope.com/product/BS05/*

[4] BitScope Meter: *http://www.bitscope.com/software/meter/*

[5] Oscilloscope kit: *https://www.nooelec.com/store/dso138.html*

[6] Raspbian Stretch Lite: *https://www.raspberrypi.org/downloads/raspbian/*

[7] PuTTY: *https://putty.org*

[8] example.py code: *https://github.com/tutRPi/Raspberry-Pi-Heartbeat-Pulse-Sensor*

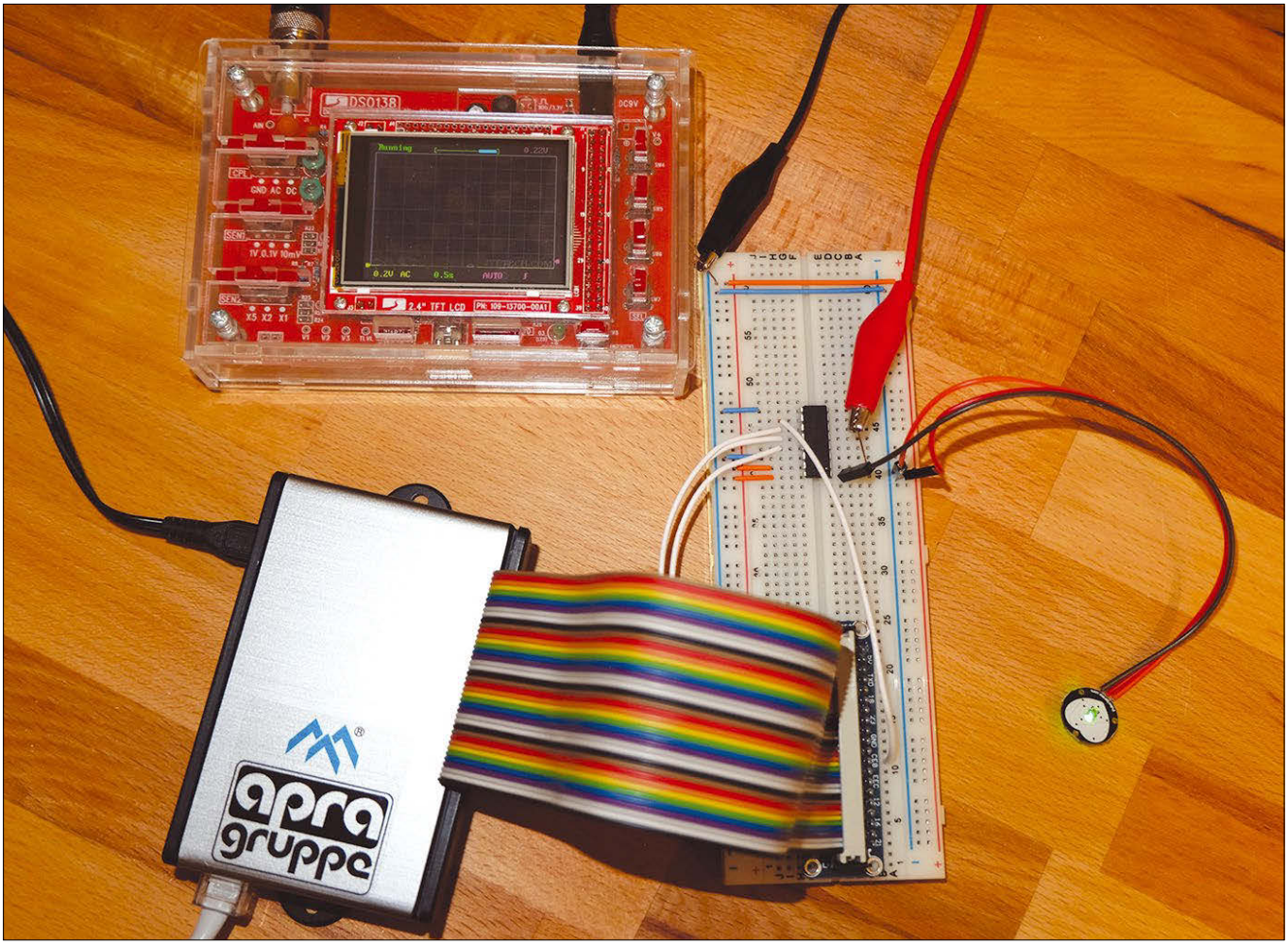


Figure 3: The oscilloscope displays pulses, indicating the accuracy of the sensor.

MakerSpace

Librem 5 Production Challenges

Freedom and Security

Despite a few ups and downs in development, Purism moves ahead in its quest to produce a free and secure phone. *By Bruce Byfield*

In August 2017, Purism made headlines with its plans to produce a secure, free-licensed phone called the Librem 5 (Figure 1) [1]. Further headlines followed with announcements of partnerships with other projects like KDE and Gnome. But how is development faring now? Recently, I talked with Todd Weaver, Purism's CEO, about the challenges of developing an open hardware phone (Figure 2).

Figure 1: : A mockup of the Librem 5 phone, scheduled for release in April 2019.

According to Weaver, Librem 5 development is constrained by Purism's commitment to software and hardware as well as security. To start with, as a social purpose corporation, Purism is "devoted to providing the highest quality hardware available, ensuring the rights of security, privacy, and freedom for all users" rather than being focused entirely on profit [2]. This principle means that Purism has set itself the goal of developing hardware that can

Figure 2: Todd Weaver, Purism CEO.

Lead Image © Nelli Valova, 123RF.com

be reproducibly built, does not track users, encrypts local data, and places all network communication fully under the user's control. Closely connected to these principles are the status of PureOS as a Linux distribution endorsed by the Free Software Foundation (FSF)[3], and the intention for the Librem 5 to achieve the Foundation's Respects Your Freedom (RYF) certification for hardware [4].

In addition, Purism strives to keep its development consistent with the policies of its partners, such as the privacy practices of the Electronic Frontier Foundation and the Gnome design guidelines, and the security policies of Debian, from which PureOS is derived. These constraints are usually consistent with each other, but they can differ at times in interpretation and emphasis, so some balancing may be required. "Society's technology genius is not lacking," Weaver says, "but our moral genius is. Purism is here to change the latter. Based on all research, hardware choices, and software development advancements, we are confident we will reach all our goals of freedom for society."

Technology Challenges

From the start, Librem 5 development was influenced by the desire to avoid what Weaver calls "the duology" – the phone market domination by Android and iOS. Purism maintains that neither duology member is compatible with Purism's security and freedom goals. However, perhaps an even greater challenge is that running the Librem 5 on a Linux version means foregoing the advantage of using existing app stores, each with thousands of downloadable apps. True, PureOS is based on Debian, which is often claimed to be the largest repository of Linux packages, but relatively few Linux apps are designed to run on phones.

At this stage, Weaver is not releasing details of exactly how Purism intends to answer this challenge. All he is currently saying is that "this is a simple equation of momentum and time. We have hundreds of development kits shipping to active developers. Plus, we have emulators, so we will see more and more applications being written and ported to PureOS. It is only a matter of time."

Another challenge is finding manufacturers. Purism already has established some relationships for assembling its line of high-security laptops, but the Librem 5 is the company's first venture into the phone market. "We have been able to use a lot [of existing suppliers]," Weaver says, "but we have also added new suppliers into the mix, since fabricating a phone from our schematics on up clearly requires adding new suppliers into the mix."

In addition, finding these new suppliers may be affected by the fact that Purism is a small company compared with companies like Apple or Samsung that dominate the phone market. "Some suppliers deal with gigantic volumes and only deal with a few customers," Weaver notes. Fortunately, "most suppliers deal with varying volumes based on other factors such as upfront fees and volume pricing."

One of the greatest challenges is that Librem 5 development was planned to take at least 18 months – more than enough time for market standards to change. Purism has been aware of this problem from the first days of its Librem 5 crowdfunding campaign and has been criticized several times for its reluctance to release final specifications.

Events, though, prove that reluctance only makes sense. Nicole Faerber, Purism's CTO (Figure 3), has blogged about Purism starting development with the NXP i.MX 6 System on a Chip (SoC). This model was ideal for the Librem 5, because it "did not require a proprietary firmware for normal phone cases [and included] an advanced free etnaviv GPU driver. The fact that the SoC has been on the market for quite some time also meant that the drivers were stable and solid."

Development with the i.MX 6 began before the crowdfunding campaign in August 2017. By September, it had reached the point where early prototypes [6] were working and capable of running Debian Unstable, Wayland, and Gnome (Figure 4). Yet already rumors of the new generation i.M 8 existed [7].

Early in 2018, Weaver and Faerber saw a demonstration of the i.MX 8, and, Faerber writes, concluded that the "i.MX 8 was well suited for the Librem 5 and [would] be available in larger quantities early enough for our initially scheduled shipping date of January 2019. The decision to upgrade to a newer generation of the SoC (from

Figure 3: Nicole Faerber, Purism CTO.

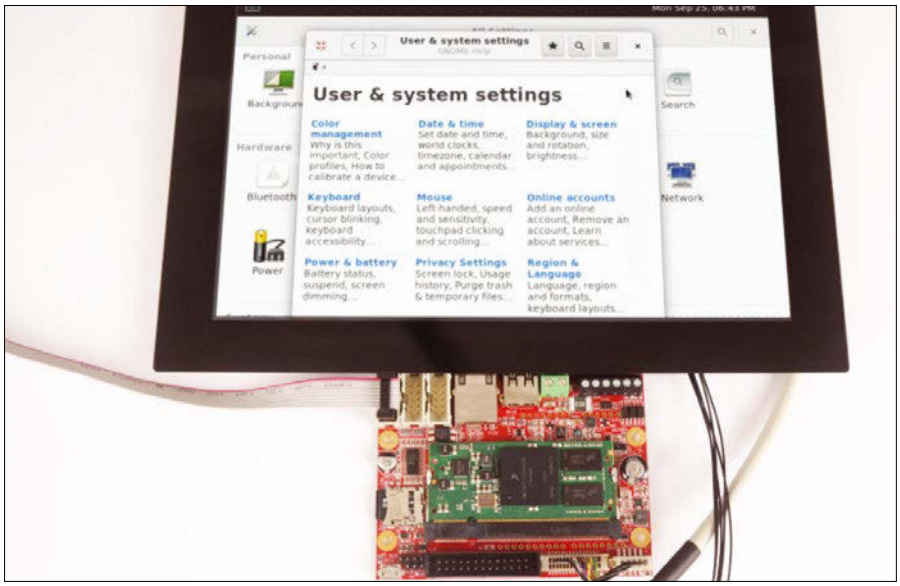


Figure 4: An early Librem 5 prototype.

the same hardware family) brings the benefits of a modern 64-bit CPU, more GPUs on the silicon, lower power consumption, and a host of other upgrades, while still allowing us to maintain our strict requirements to seek the FSF's RYF endorsement. We believe that the risk trade-off is worth it given the modern upgrades the new SoC provides us."

In the same blog, Faerber describes how, around the same time, Purism realized that the cellular modem, which Faerber describes as "arguably the most complex part of a mobile phone" ran on "a blackbox proprietary operating system," using the same RAM as the SoC and thus created a security risk. Purism decided it needed to isolate the cellular modem but faced the problem that nearly all cellular modems suffered from the same potential security flaw. Purism had no choice except to redesign to keep the cellular modem from using the same RAM.

These explanations of specification changes were accompanied by the inevitable: delays in the release of the Librem 5 development kit and in the release of the Librem 5 itself. Originally planned for January 2019, the Librem 5 is now scheduled for April 2019.

All these problems are compounded by the fact that manufacturing is being done in Asia. Similar to other open hardware companies like Keyboardio, Purism has found it necessary to visit manufacturers regularly to keep the project on course. "We are looking at doing more and more fabrication within the US under our own roof." Weaver's hope is that "as we grow we gain leverage to have tighter control of the entire supply chain." Meanwhile, traveling is just another part of the price to pay for developing a new product.

Such delays would not be unusual in most development projects. According to Weaver, "most of the delays we see are attributable to unforeseen events, such as part shortages, shipping losses, deaths, and weather," the same sort of delays that can plague any product development. "Short-term disadvantages are clearly choices of long-term gain."

Coming Attraction

Purism is developing other products at the same time as the Librem 5. Current projects include the Librem Key, a physical security measure, and, coming in February 2019, a new product called Liberty Services.

But like the Librem laptops, the Librem 5 phone is intended to be what Weaver calls "a flagship product." Hearing Weaver talk about the phone's prospects, it quickly becomes apparent that the ideals behind it are as important as the product to him. The Librem 5 is almost certain to find a niche market. Whether, though, it can become "the catalyst to influence change in society" remains to be seen. At this point, all that can be said is that the Librem 5 is by far the most ambitious attempt to build an open hardware phone yet – and, so far, the ups and downs of development are not seriously slowing down its production. ■■■

Info

[1] Librem 5: *https://shop.puri.sm/shop/librem-5/*

[2] Social purpose corporation: *https://puri.sm/about/*

[3] FSF's list of free distributions: *https://www.gnu.org/distros/free-distros.html*

[4] RYF certification: *https://www.fsf.org/resources/hw/endorsement/respects-your-freedom*

[5] SoC model: *https://puri.sm/posts/librem5-2018-09-hardware-report/*

[6] Early prototype: *https://puri.sm/posts/librem5-roadmap-to-imx8/*

[7] i.MX 8: *https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/i.mx-applications-processors/i.mx-8-processors/i.mx-8-family-arm-cortex-a53-cortex-a72-virtualization-vision-3d-graphics-4k-video:i.MX8*

Time marches on, but all those old multimedia playback devices won't last forever. Are you starting to worry that someday you won't have the hardware you need to play you're favorite analog audio and video recordings? This month we take you through the steps of digitizing a VHS tape.

Also, if you're ready to lock down your security with SSH, but you don't want to worry about memorizing shell commands, you'll enjoy our article on the GUI-based EasySSH, a handy tool that lets you start an SSH session with a mouse click and manage multiple connections from a single interface.

Dial up your command-line skills with this month's tutorial on Bash shell variables, and give your adorable cat photos cool laser vision with the Natron special effects editor.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE

Doghouse – RISC-V 65
Jon "maddog" Hall
As open source software development for the RISC-V architecture moves ahead, maddog says stop complaining and start contributing to the project.

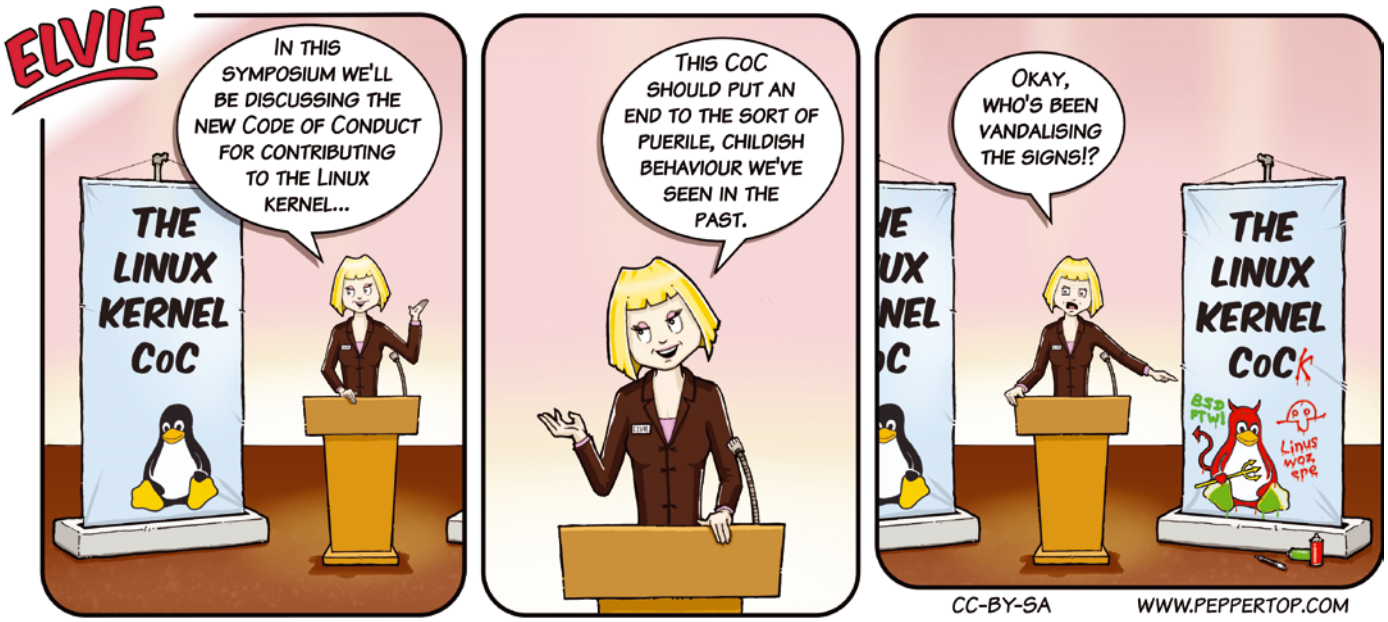
EasySSH 66
Ferdinand Thommes
EasySSH lives up to its name and starts SSH connections at the click of a mouse.

File Conversion 70
Rubén Llorente
The hardware to play old VHS tapes or audio cassettes is becoming more difficult to find. We'll show you how to convert those old analog treasures to digital format for future enjoyment.

FOSSPicks 78
Graham Morrison
Graham nearly made it through an entire month of FOSSPicks without an esoteric audio discovery. And then he found VeeSeeVSTRack.

Tutorials – Shell Scripting 84
Marco Fioretti
You do not need to learn low-level programming languages to become a real Linux power user. Shell scripting is all you need.

Tutorials – Natron 90
Paul Brown
Natron gives you the power to apply sophisticated effects to your videos.

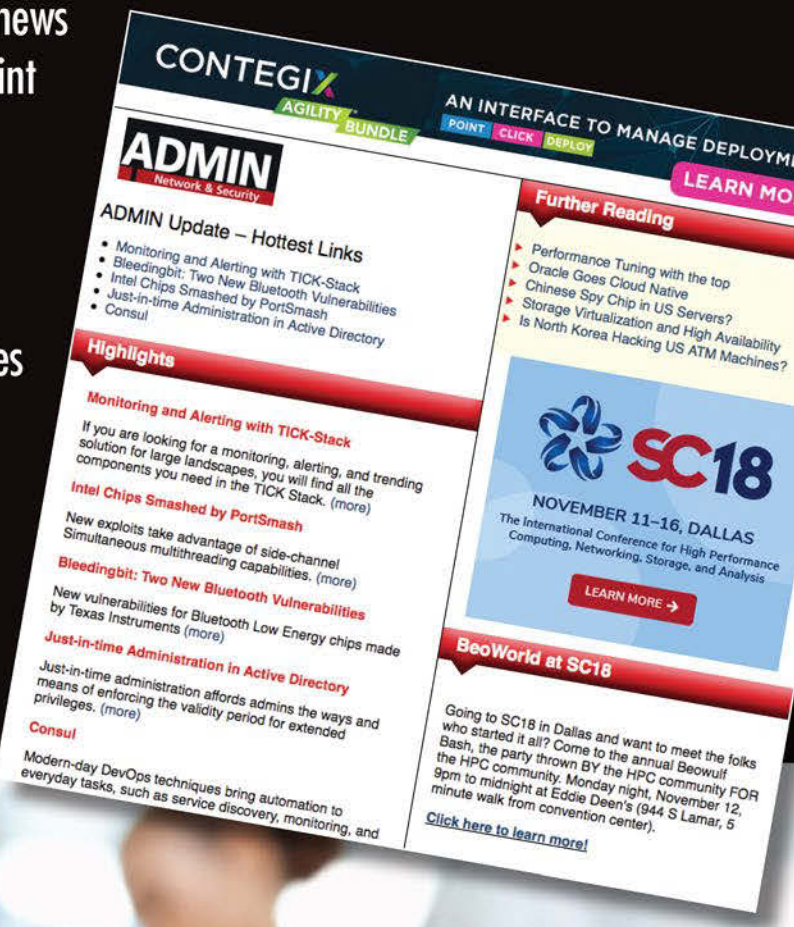


Too Swamped to Surf?

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

Subscribe today to our free ADMIN Update newsletter and receive:

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



ADMIN
Network & Security

© Yuri Arcurs, fotolia.com

www.admin-magazine.com/newsletter

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®.

As open source software development for the RISC-V architecture moves ahead, maddog says stop complaining and start contributing to the project.

BY JON "MADDOG" HALL

RISC-V architecture development

Recently a colleague sent me a link to an online article about the RISC-V Foundation and the Linux Foundation agreeing to work together to advance Linux and other open source software on top of the RISC-V architecture.

RISC-V, for those of you who do not know about it, is a relatively new architecture that originated at the University of California, Berkeley (the same people who developed the Berkeley Software Distribution, also known as BSD).

Having years of history and experience behind computer science, these architects "started from scratch" and were able to develop a relatively clean architecture that could support multiple address space sizes, as well as multiple usage cases (servers, portable devices, embedded systems, Internet of Things, etc.). This design allows trade-offs in the chips for speed and power without moving away from the basic architecture.

One of the other features of the open architecture is that it can be used without the oppressive licensing fees of some other architecture vendors. It allows for hardware vendors (either commercial or non-commercial) to create chips that follow the basic architecture but also add extensions as desired.

The article was published on December 1, 2018. Since I received the link on December 1, 2019, enough time had passed for the normal peanut gallery of commenters to make their thoughts known.

Many of these comments seem to have come from people who have never produced a piece of sophisticated hardware in their life (and perhaps never even produced a piece of sophisticated software). While some of the people had intelligent and insightful knowledge to provide, other commenters were often ignorantly rude.

As an example, when it was pointed out that the project was still in its early stages, people wanted to know when they could have "their chip" or "their board" for whatever proposed purpose. As a response, other commenters would start to detail the issues around taking a proposed architecture and turning it into real silicon, not the least of which is fitting it to a manufacturing process, scheduling the production, and setting up the supply chain for getting the chips onto a board and into the hands of the "customers." While these steps are in progress, it takes time (and money) to formulate. This process is rolling out, but apparently not fast enough for the anxious (and rude) commenters.

Then came a discussion about how much the boards might cost. Many of the commenters were shocked by the prices of "developer boards," boards that would go into the hands of people working on the compilers, operating system, and libraries to help make the entire system available. In the free software space, a lot of these people, even today, are volunteers, and not able to afford a chip and board that are made in relatively low production numbers.

Therefore, companies like SiFive [1], which is one of the first companies to actually produce silicon and boards supporting the architecture, are not charging "Raspberry Pi" prices for their high-end development boards. Instead, they have innovative programs that allow developers to get the boards needed to do real work, as well as generate the funds necessary to create new boards. In addition, SiFive has boards that allow people to experiment with the architecture at a lower price than SiFive has to charge for their higher end boards.

Many of the commenters were complaining about the business plans of companies that used closed-source tools to make creating the silicon easier and faster. While these companies are using closed-source tools, other companies and groups were creating open source techniques, which may take more person months but that were still available for those who wished to use completely open source techniques.

Finally, there were the naysayers, who (when they finally started to comprehend the roadmap to bringing out this architecture) questioned why it was necessary and lamented how long it would take.

For the naysayers, I have these thoughts: Linus started the Linux Kernel project in late 1991, and it was more than two years (early 1994) until it reached version 1.0 level. Distributions appeared in late 1993 and gained speed in mid-1994. It was not until 1998 that commercial databases started to support their products on top of GNU/Linux (a sign of acceptance in the commercial Unix space) and 2000 when Linux was used for embedded systems. It still took a decade for GNU/Linux to be accepted by a lot of businesses.

The year 2019 will mark the 25th anniversary of the release of version 1.0 of the Linux kernel and the creation of the Beowulf supercomputers (using Linux at their core). The latter project now drives all 500 of the fastest computers in the world.

Some things take time and money. In the scope of things, RISC-V is moving at the speed of light. Instead of complaining, join the project. ■■■

Info

[1] SiFive: *https://www.sifive.com*

Simplifying SSH

EasySSH lives up to its name and starts SSH connections at the click of a mouse.

BY FERDINAND THOMMES

Most experienced Linux users are familiar with Secure Shell (SSH), with some being regular users. The protocol supports encrypted connections with remote devices via TCP/IP according to the client-server principle. Typically, you connect to servers that do not have a graphical user interface (GUI). However, with the appropriate bandwidth, you can manage tools with a GUI using X forwarding.

Easier than SecPanel

Most Windows users are familiar with the PuTTY open source terminal emulator as an interface for SSH, whereas Linux users usually interface with SSH on the command line. Although PuTTY is also available for Linux, SecPanel [1], which has been under development for many years, has gained better acceptance in the Linux world (Figure 1).

SecPanel stores frequently used connections in profiles and creates them at the push of a button. If you are looking for a less complex alternative, EasySSH [2] is worth a look. When installing EasySSH, it is important to make sure that you download the correct tool (see the box "Same Names").

Even if you prefer to work at the command line, it makes sense to automate repetitive steps, such as establishing an SSH connection. The more servers you contact, the more useful you will find an easy-to-use GUI tool that remembers each host's data.

Restrictions

EasySSH, which has only been under development for a few months, currently has a major limitation: To reduce the act of opening an SSH con-

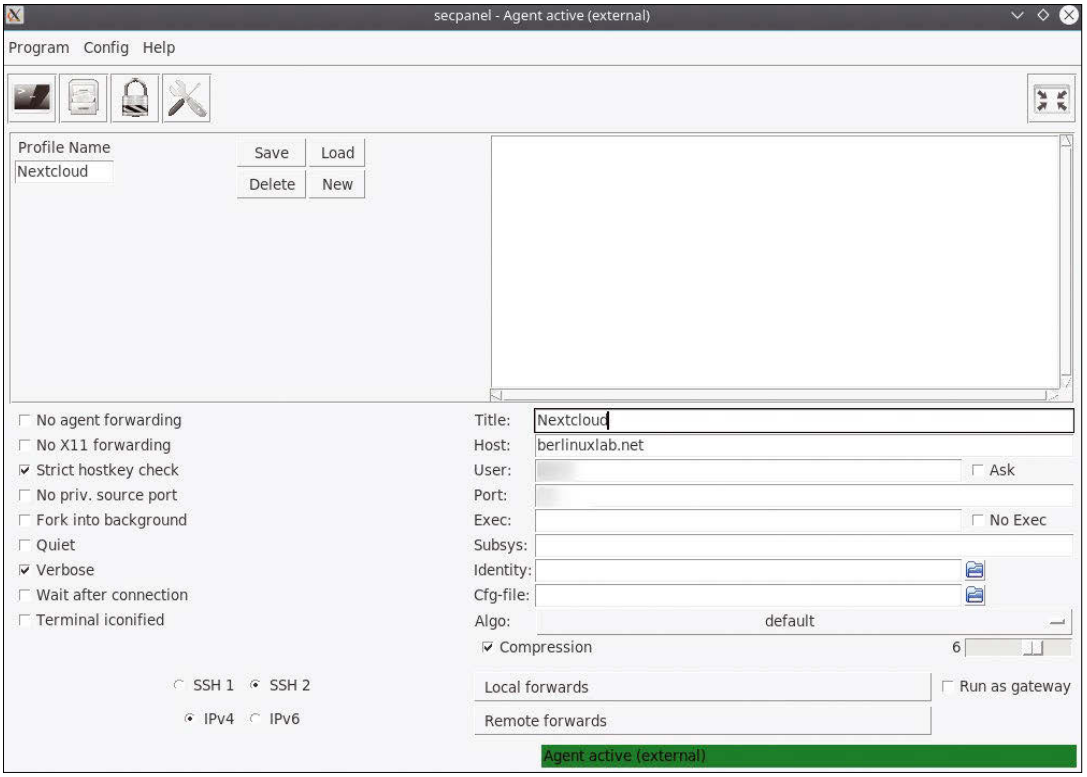


Figure 1: Although SecPanel offers more functions than EasySSH, it is also more complicated in daily use.

Same Names

EasySSH is an easy-to-use tool that acts as an SSH connection manager. It was written by the Elementary OS distribution team but is also suitable for other distributions. Do not confuse the tool with the `easyssh` script, which implements functions of the SSH protocol in the Go programming language.

nection to a single mouse click, the program saves the access credentials, including the password, in plaintext in the local configuration file. Consequently, it is only suitable for computers that are in a trusted environment or that have an encrypted disk. However, the developer has promised that the configuration file will soon be encrypted and protected by a master password. Once security can be guaranteed, there is no drawback to using this program.

Since EasySSH is not found in the package sources of major distributions at this time, you must install it manually. For easier installation, you can use a Flatpak (current version 1.3.4) available on Flathub [3], the Flatpak app store. Additionally, Arch Linux users can download EasySSH from the AUR.

Flatpak Installation

To install from Flathub, go to the Flatpak Quick Setup [4] and follow the instructions for your distribution. Next, reboot your system. Gnome users should then be able to install the program with Gnome Software (Figure 2); KDE supports the integration of Flatpak applications as of Plasma 5.13 in the Discover graphical package manager.

Installation at the command line is faster. To do this, enter the

```
flatpak install
  flathub com.github.muriloventuroso.easyssh
```

command as a normal user. The system first imports the required run-time environments and then EasySSH itself. The run times offer benefits with later Flatpak installations.

How It Works

Once installed, start EasySSH. The main window has a lot of empty space, which is only interrupted by the name of the program and the *Add Connection* button in the middle. Clicking on the hamburger menu at top right opens a frugal Preferences dialog, which is divided into *General* and *Appearance* tabs (Figure 3).

Under *General*, you can define the storage location of the host configuration. By default, this is `/var/lib/flatpak`, with the configuration folder itself (as for all Flatpaks) residing in the user's home directory under `~/.var/app`.

Initially, you do not need to change anything here. It is also possible to synchronize the SSH configuration under `/var/lib/flatpak/apps/`. In the

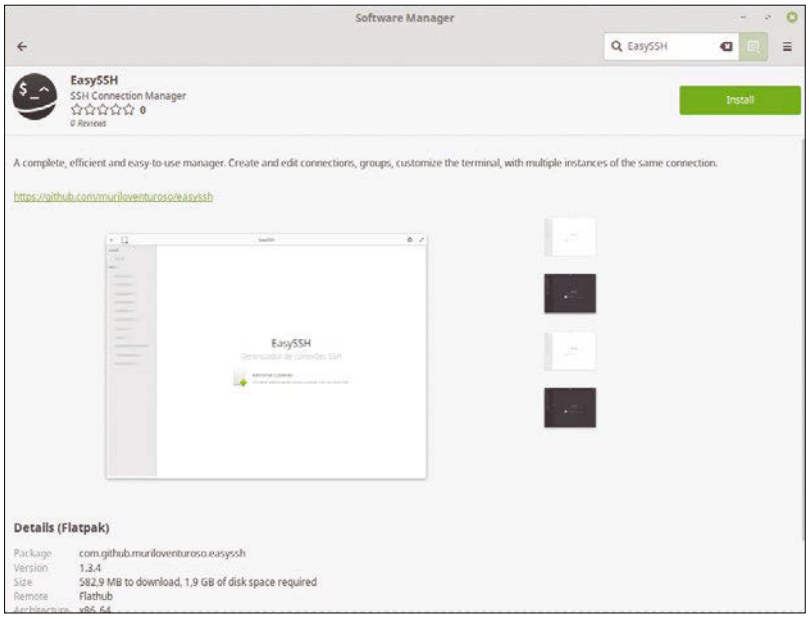


Figure 2: Flathub is the central app store for Flatpaks. First configure access to this parallel ecosystem and then install EasySSH.

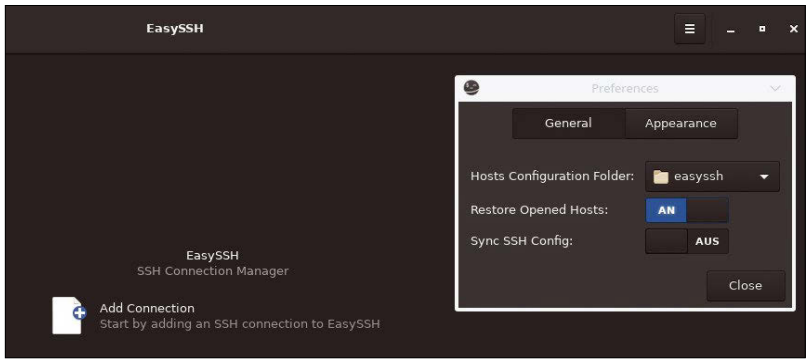


Figure 3: Clicking on the EasySSH hamburger menu lets you access a dialog to configure the application.

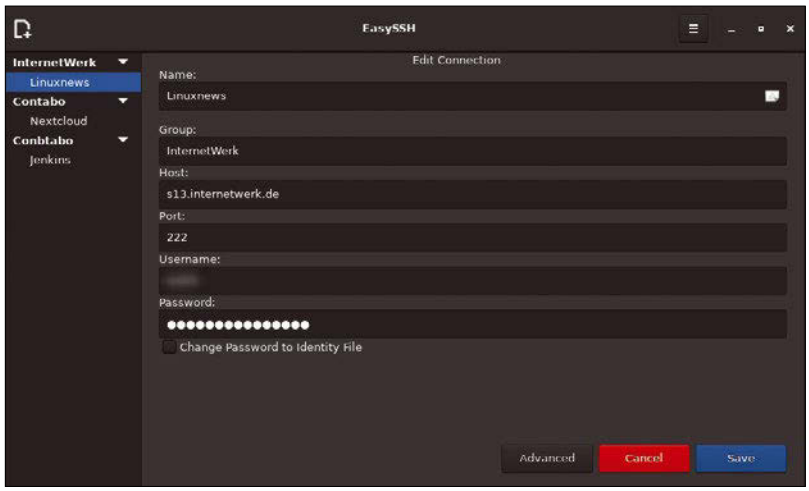


Figure 4: A new connection is quickly created. EasySSH stores all necessary access data and reduces the connection overhead to a single mouse click.

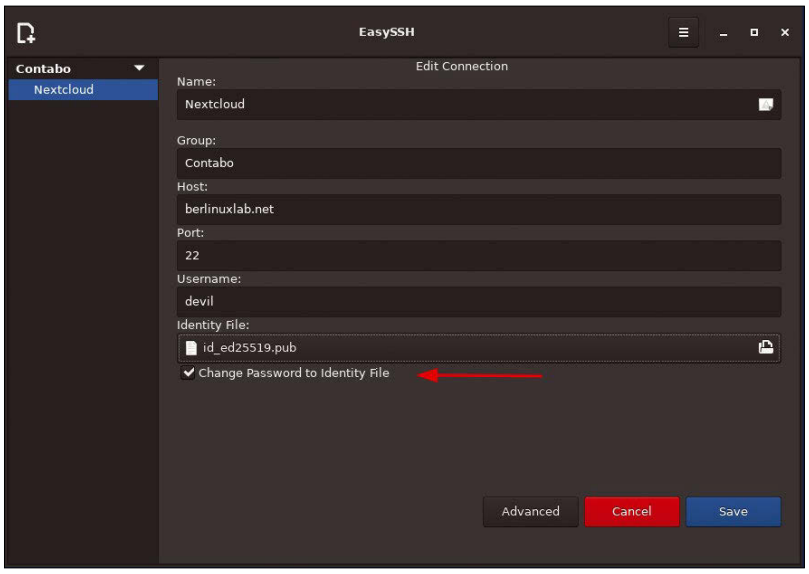


Figure 5: Clicking the checkbox lets you use an SSH key, which is more secure than a password.

Appearance tab, you can set a dark theme as default, the color of the terminal background, and the terminal font.

New Connection

To set up a new connection, just click on the corresponding button or the plus sign in the upper left-hand corner. A dialog will then appear prompting you for the information required to establish an SSH connection. You are free to choose your name and group. Organizing this in groups improves the overview if you use several accounts on one server. The host, port, user, and password details are then queried (Figure 4).

A more secure way to connect to servers via SSH is to replace the password with an asynchronous cryptographic key pair, commonly known as an SSH key. The public key is on the server, while the private key remains on the local computer from which you connect.

Alternatively, EasySSH supports the following method: Select the *Change Password to Identity File* checkbox (Figure 5) in the Edit Connection dialog. Navigate to the public key with the `.pub` extension, which is located in `~/.ssh` by default (see the "Tip" box), and select it.

If you do not yet have a key pair or want to create a new pair for EasySSH, then run

```
ssh-keygen -t rsa -b 4096
```

in the terminal. (See instructions online (e.g., [5]) for more information on creating SSH keys).

Multiple Connections

After saving the data, a bar appears on the right, listing all the connections you entered (sorted by group and name). A click on an entry lets you delete, edit, and open a connection. If the login information is correct, a terminal opens that, if a password is used, establishes the connection in a tab without further prompting (Figure 6).

If you entered an SSH key in the settings, the software prompts you for the matching password in the terminal (if you protected the key). A click on the plus sign in front of the tab opens a second connection to the same server.

Tunneled

If you enter a new connection and click on *Advanced* below it, you will see another functionality of EasySSH: It is capable of establishing SSH tun-

Tip

Most file managers do not display hidden files by default. However, with Ctrl+H or the context menu, you can usually switch to displaying all the files in the directory.

Figure 6: EasySSH opens the SSH connection with a single mouse click. Keep in mind, the password currently appears in plaintext.

nels, which means you can forward a local network port to another port or the same port on a remote computer. The data is routed through a secure SSH connection between the two computers.

Forwarding ends either at the remote computer (the gateway) or at another computer behind the gateway. Building a tunnel makes sense, for example, if you work on unsecured networks while traveling (e.g., in a hotel or Internet café) and therefore require an encrypted transfer option.

Many universities and companies also block outside access to important computers in the network, for which you can use a tunnel through which you first connect to a server. To do this, establish a tunneled connection manually in the terminal as shown in Listing 1. In EasySSH, enter the local and destination ports for *Source Port* and *Destination*.

Conclusions

EasySSH provides a single-click SSH connection manager. According to the developer, the program does not need any future improvements: The application does its job without any recognizable bugs. It quickly becomes apparent that the developer first took his own application scenarios into account in developing EasySSH. However, the repository on GitHub is very active, which gives hope for a quick remedy to this program's initial teething issues.

In practice, EasySSH is simple and easy to use. Although some important functions are still missing (e.g., password encryption), EasySSH is worth keeping an eye on for future developments. ■■■

Info

[1] SecPanel: *http://www.linux-community.de/ausgaben/linuxuser/2006/12/ssh-bequem/*

[2] EasySSH: *https://github.com/muriloventuroso/easyssh*

[3] Flathub: *https://flathub.org/about*

[4] Quick Setup: *https://flatpak.org/setup/*

[5] Create an SSH key: *https://www.ssh.com/ssh/keygen/#sec-Creating-an-SSH-Key-Pair-for-User-Authentication*

Listing 1: Establishing a Tunnel Connection

```
ssh -L <local port>:<target computer>:<target port> -l <user name> <gateway>
```

Saving Your Analog Data from Oblivion

If you have old VHS tapes or audio cassettes lying around, the hardware to play these analog formats is becoming more difficult to find. Here's how to convert those old analog treasures to digital format for future enjoyment.

BY RUBÉN LLORENTE

Transferring VHS tapes, audio cassettes, and other analog home media formats to a digital format, such as Ogg or Matroska, can be a complex and expensive process with archival-grade conversions. In this article, I show you a simple and inexpensive method for digitizing your VHS tapes that is perfect for personal use.

Shopping List

To convert a VHS tape to a digital format, you need two pieces of equipment: a playback device for the original medium and a capture device to read the playback device's output. In addition, you need transcoding software to process the data that the capture device retrieves from the playback device. For a list of what I used in this project, see the "Software and Hardware Requirements" box.

For the playback device, you can find a used VCR for less than EUR100 online ($100-$200+ for NTSC/PAL/multisystem). Keep in mind that your chosen playback device must match the color encoding system of the VHS tapes you intend to transfer. Trying to play a PAL VHS on an NTSC device won't work (see the box "PAL vs. NTSC").

For a capture device, I use a USBTV007 EasyCAP (Figure 1), which is inexpensive (less than EUR15/may be difficult to find in the US), performs captures of acceptable quality, and has Linux support. Keep in mind that "EasyCAP" is not a commercial brand name; it is a popular term used by Chinese manufacturers to designate cheap, simple USB capture cards. Finding a specific EasyCAP model can be challenging, since manufacturers usually don't provide the full specification list in their product description or similar use cases. See the box "The Other EasyCAP."

You will also need an SCART to RCA adaptor (Figure 2), which may be purchased for less than EUR10 ($8). The SCART connector is plugged into the VCR output, and the RCA plugs into the EasyCAP connectors. Most RCA connectors are correctly color coded (Table 2).

In addition, I used a laptop with an i5-2467M CPU for my tests, but you can use a computer with less horsepower. A dual-core CPU of 2.50GHz for each core is the minimum require-

The Author

Rubén Llorente is a mechanical engineer, whose job is to ensure that the security measures of a small clinic's IT infrastructure are both legally compliant and safe. Additionally, he is an OpenBSD enthusiast and a weapon collector.

Software and Hardware Requirements

Hardware:

- Firstline VCR-602
- USBTV007 EasyCAP
- SCART to RCA adaptor

Software:

- GStreamer 1.6.4
- FFmpeg 3.2.4
- SoX 14.4.2

PAL vs. NTSC

PAL and NTSC are two color encoding systems – originally intended for analog television – that also apply to VHS tapes, players, and recorders:

- Phase Alternating Line (PAL) is used in most of Europe, Australia, the Middle East, Asia, and a big part of South America [1]. PAL media run at 25 frames per second (fps) and have a frame size of 720x576. In this article, the example commands are for PAL media.
- National Television System Committee (NTSC) is used in North America, Japan, parts of South America, and a few other countries [2]. NTSC media run at 29.97fps and have a frame size of 720x480. If you try to reproduce the steps in this article with NTSC devices, you will need to replace these values in the examples.

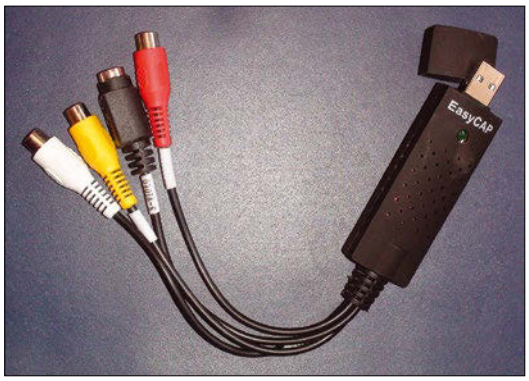


Figure 1: This EasyCAP device works as a cost-effective capture device.

Table 1: Pixel Formats

GStreamer	FFmpeg
YUY2	yuyv422
UYVY	yuv422p
YV12	None known

Table 2: RCA Color Codes

Color	Connection
White	Left audio
Red	Right audio
Yellow	Video

ment. However, tests showed that the procedures described in this article will result in a barely acceptable loss of frames in the encoding when using this weak of a CPU. Using a drive with a high write speed is also advisable, since the VHS will be transferred and saved to the file-system in real time.

In terms of software, I used GStreamer with the Ugly plugin for the capture (see the "Why GStreamer?" box) and FFmpeg to transcode. FFmpeg requires x264 and *libdfk-aac* support. Finally, the SoX sound processing utility is used to clean up the transferred media's audio.

Hardware Detection

First, power up your computer and connect all the hardware as previously described. Then, make sure your operating system properly identifies the capture device.

The USBTV007 EasyCAP will show up as two different capture devices: one for audio and one for video. This model does not work with the PulseAudio sound server, so you might need to tell PulseAudio to ignore the device so that ALSA can manage it instead. To do this, use the graphical tool `pavucontrol`: Move to the *Configuration* tab and select the *Off* profile for USBTV007. If you prefer to work from the command line, then enter,

```
pactl set-card-profile $card_number off
```

where `$card_number` is the identifier of your EasyCAP device for PulseAudio. If you don't know what it is, you can find out by typing:

Figure 2: This SCART to RCA adaptor has an extra pin for audio input, which I won't be using.

The Other EasyCAP

Many EasyCAP devices other than the USBTV007 work under Linux. For more information about these other devices, see the LinuxTV wiki [3], but keep in mind that it is a bit outdated. (In fact, the wiki is so outdated that the EasyCAP model used here is reported to be unable to capture sound at the time of writing.)

Some of these EasyCAPs might require different command-line switches than the ones provided in this article's examples to work. If you have trouble getting your EasyCAP to work, try using a different pixel format option than YUY2 with GStreamer, such as UYVY or YV12 (Table 1). If you use any of these, you might want to supply a different `pix_fmt` switch to FFmpeg, although this is not really necessary .

Why GStreamer?

Many tutorials suggest capturing your VCR or cassette player's output directly with FFmpeg or MPlayer. Instead, I chose GStreamer to capture the VCR output and pipe it into FFmpeg, because GStreamer has better error tolerance when dealing with faulty media feeds. This means GStreamer will work better with cheap capture devices that provide bad frame rates (e.g., 25.02fps instead of 25fps for PAL video). This is specially true when muxing the output to formats that support time stamps, such as Matroska [4].

```
pactl list cards |
  grep -E 'device.product.name|device.string'
```

To list your currently detected audio inputs, enter the command:

```
arecord -l
```

This will display the working capture devices detected by ALSA.

Video inputs will show up as files named `/dev/video*` (e.g., `/dev/video0`, `/dev/video1`, etc.). If you have more than one video input – for example, you have a webcam in addition to an EasyCAP – and you are not sure which one is which, you can extract information from each device with the `v4l2-ctl` command. The following command will display some known properties of `/dev/video0`:

```
v4l2-ctl --device=/dev/video0 --list-inputs
```

The Capture Process

Once everything is connected and detected, it is time to capture the video. To do this, insert a VHS tape into the VCR and play it, and then turn on your capture software, which takes the VCR's output and dumps it into a file on the fly.

The raw video and audio stream can take up a lot of storage, probably more than 100GB, which is why most people prefer encoding that stream into something more manageable as it is captured. Because the computer is capturing the raw VCR output in real time, your encoder must be both fast and CPU-friendly, so as not to affect the data capture negatively. If you try to use an encoding configuration that is too hard on the CPU, it will not have enough time to process each piece of incoming data before the next one arrives, which will result in lost frames and data loss.

Listing 1 performs a lossless capture of the analog input. While not very practical, Listing 1 is provided as a reference. Notice that the buffers are set to zero in order to avoid problems during the real-time video capture. The pixel format, YUY2, is set to the format that the EasyCAP feeds to the computer. Audio is captured in a lossless format at a sampling rate of 48KHz in stereo. The output is dumped to a Matroska file. Listing 1 is tuned for PAL devices (if you are using NTSC, see the "PAL vs. NTSC" box.)

Listing 1: Compressionless Capture Command

```
gst-launch-1.0
  -q v4l2src device="$videodevice" do-timestamp=true norm="PAL" pixel-aspect-ratio=1
       ! video/x-raw,format=YUY2,framerate=25/1,width=720,height=576
       ! queue max-size-buffers=0 max-size-time=0 max-size-bytes=0
       ! mux.
     alsasrc device="$alsadevice" do-timestamp=true
       ! audio/x-raw,format=S16LE,rate=48000,channels=2
       ! queue max-size-buffers=0 max-size-time=0 max-size-bytes=0
       ! mux.
     matroskamux name=mux
       ! queue max-size-buffers=0 max-size-time=0 max-size-bytes=0
       ! filesink location=vhs.mkv
```

Listing 2: Compressed Capture Command

```
ffmpeg -loglevel 32 -t 01:40:00
       -i <( gst-launch-1.0 -q v4l2src device="$videodevice1"
             do-timestamp=true norm="PAL" pixel-aspect-ratio=1
             ! video/x-raw,format=YUY2,framerate=25/1,width=720,height=576
             ! queue max-size-buffers=0 max-size-time=0 max-size-bytes=0
             ! mux.
            alsasrc device="$alsadevice" do-timestamp=true
             ! audio/x-raw,format=S16LE,rate=48000,channels=2
             ! queue max-size-buffers=0 max-size-time=0 max-size-bytes=0
             ! mux.
            matroskamux name=mux
             ! queue max-size-buffers=0 max-size-time=0 max-size-bytes=0
             ! fdsink fd=1
           )
       -c:v libx264 -preset ultrafast -x264opts crf=18:keyint=50:min-keyint=5
       -pix_fmt yuyv422
       -c:a flac
       -f matroska file:vhs.mkv
```

In Listing 1, replace "$videodevice" with a video input device (e.g., /dev/video1). The "$alsadevice" (in the alsasrc device line) describes an audio input such as hw:1,0 or hw:2,0. The ALSA audio inputs are always named in a "hw:$card,$device" format.

Listing 2 performs lossy compression while carrying the analog stream's capture, resulting in a smaller file. It is configured to capture video for 1 hour and 40 minutes, but you can stop the recording by pressing *q* at any time.

Listing 2 offers a good trade-off. GStreamer captures the VHS output and pipes it to FFmpeg for transcoding (see Table 3). It will encode the raw stream without a noticeable loss of quality and will diminish the output file size by a factor of ten. Because it is CPU-friendly, it allows the capture of the analog data without frame loss. The resulting file is, however, still too big for most users. It is also likely to suffer from audio and video noise, since old analog data is susceptible to acquiring defects because of the media's age. Therefore, the captured video will require post-processing.

Table 3: FFmpeg Options

Option	Meaning
`-t 01:40:00`	Sets a duration for the recording.
`-c:v libx264`	Uses the x264 video codec.
`-preset ultrafast`	Makes the encoder work as fast as possible.
`crf=18`	Sets the quality of the encoding. Lower is better, but also implies larger files.
`keyint=50`	Maximum interval between keyframes. Low values make it easier to seek specific positions in the video [6].
`min-keyint=5`	Minimum interval between keyframes.
`-pix_fmt`	Pixel format (the same as EasyCAP).
`-c:a flac`	Audio is encoded as FLAC, which provides reasonable compression without reduction of sound quality.
`-f matroska`	Mixes the audio and the video into a Matroska multimedia container.

Audio Cleanup

When you capture the audio of a VHS tape with the process outlined here, it is likely to be noticeably noisy. In addition to the noise contained in the VHS and generated by the VCR (see the box titled "Removing Audio Noise from Old VHS Tapes"), the capture card also produces its own noise. Especially true when using cheap capture devices, the capture card will always introduce a

Removing Audio Noise from Old VHS Tapes

Some tapes are so worn that you might want to remove more than the capture card noise. If the VHS suffers from a constant hiss throughout the reproduction, you may want to try the following approach.

Instead of extracting a noise sample by the method described for capture card noise, take a sample during a portion of the VHS tape that should be silent. Many commercial tapes have silent parts at the beginning and end. To capture the hissing noise that occurs throughout the tape, record a few seconds of such segments. Then use this sample for generating a noise profile, as described in the article.

This process is highly destructive. Some might call it butchering the VHS, and they may be right. This will remove the hiss, but it may introduce sound artifacts that can be worse than the original noise. Use more conservative noise reduction values (less than 0.2) if you take this route. Experiment until you get the result you want.

barely audible buzzing sound to the resulting recording.

To mitigate the noise introduced by the capture card, you can use SoX. First, capture an audio segment that only contains the noise introduced by the capture card. With your EasyCAP connected to the VCR when it is not generating any sound output (e.g., when the tape is paused), capture a few seconds of an audio segment as follows:

```
gst-launch-1.0 ➥
  -q alsasrc device=$alsadevice ➥
  ! wavenc ! fdsink | ➥
  sox -t wav - -n trim 0 1 noiseprof noiseprofile
```

This will create a file named `noiseprofile` that contains a description of the noise introduced by EasyCAP. This profile can be used for reducing the noise in the audio you capture.

Now, the tape's digitized audio can be extracted from its file and cleaned with:

```
ffmpeg -i vhs.mkv ➥
       -acodec pcm_s16le ➥
       -vn vhs_tmpaud.wav
sox vhs_tmpaud.wav ➥
    vhs_tmpaud-clean.wav ➥
    noisered ➥
    general_noise.prof 0.21
```

A clean version of the audio will be recorded to `vhs_tmpaud-clean.wav`. The last value of the `sox` command above defines the denoiser's aggressiveness. Higher values mean more noise reduction, but at the expense of sound fidelity. Denoising is a destructive procedure that may wipe information away. Values that are too high will cause audible sound artifacts that are in fact worse than the original noise. Values between 0.21 and 0.31 are reasonable.

Finally, the audio is muxed back with the video stream. In this example, a video-only stream is extracted from `vhs.mkv` and stored in `vhs_video_only.mkv`. Then this video stream is combined with the clean version of the audio just generated, and the whole file is stored as `vhs_whole.mkv`:

```
ffmpeg -i vhs.mkv -vcodec copy ➥
       -an vhs_video_only.mkv
ffmpeg -i vhs_video_only.mkv ➥
       -i vhs_tmpaud-clean.wav ➥
       -map 0:v -map 1:a ➥
       -c:v copy ➥
       -c:a copy vhs_whole.mkv
```

Table 4: H.264 Presets

ultrafast
superfast
veryfast
faster
fast
medium
slow
slower
veryslow
placebo

The Transcoding Process

You now have a video file with clean audio that uses about 15GB of disk space. This file may be transcoded into some more practical format using any encoder, such as HandBrake or MEncoder. However, I used FFmpeg in my example, because it is easily available and well documented.

Table 5: Some Possible Tune Options

Tune	Purpose
animation	Cartoons
film	High-quality movie content
grain	Source material with a lot of grain

Listing 3: Transcoding into the Final Format

```
ffmpeg -i vhs_trim.mkv
       -vf "crop=(iw-10):(ih-14):3:0,pad=iw+10:ih+14:(ow-iw)/2:(oh-ih)/2,hqdn3d=2:1:2:3"
       -c:v libx264 -flags +ilme+ildct -profile:v high -tune:v animation
       -preset veryslow -crf 26 -c:a libfdk_aac -b:a 224k -f matroska vhs_final.mkv
```

For the excess video usually found at the beginning and end of a recording, trim it out with the following command:

```
ffmpeg -i vhs_whole.mkv ➥
       -acodec copy -vcodec copy ➥
       -ss $start_position ➥
       -to $end_position vhs_trim.mkv
```

Start and end positions are values in the form `hours:minutes:seconds`, such as `01:30:50`.

Listing 3 transcodes the captured video file and the cleaned audio file into `vhs_final.mkv`.

The second line in Listing 3 is a set of video filters that are applied to the encoding; `crop` removes the overscan at the picture's borders, and `pad` replaces it with simple black bands. `hqdn3d` is a video denoiser (see the "Removing Video Noise" box).

The `veryslow` H.264 preset ensures that the best quality-to-size ratio is obtained at the expense of encoding time (for other presets, see Table 4). The `ilme` and `ildct` flags ensure that interlacing information is preserved. The `tune:v` option serves to adjust the H.264 encoder to the content type being transcoded; I used `animation` in Listing 3 (for other tune options, see Table 5). `libfdk_aac` encodes the sound into Advance Audio Coding (AAC) at 224Kbps. (See also the "Patent-Encumbered Codecs" box.)

The `profile:v` switch selects an encoding profile. Files encoded with a `high` profile setting will be playable by most modern multimedia appliances. The `baseline` profile may be used in order to ensure compatibility with older appliances, at the expense of compression efficency. [5]

Wrapping Up

The last thing you need to do is provide (optional) metadata to the file. Listing 4 shows an example `metadata.xml` file. Matroska metadata is not well documented, but if you want to delve deeper, see some examples online [7].

Figure 3: Before (top) and after (bottom) the application of an aggressive denoiser.

The final step is to merge the metadata with the Matroska file, as follows:

```
mkvpropedit "vhs_final.mkv" ➥
    --edit info ➥
```

Removing Video Noise

VHS tapes often contain grainy artifacts. This video noise makes the file more difficult to compress with video encoders and delivers poor output quality, as well. Unless your source material is very high quality with no noticeable defects, applying a denoiser video filter to your captured file might be a good idea.

I used the high-quality denoise 3D (hqdn3d) filter in Listing 3. Hqdn3d accepts four different parameters that regulate its aggressiveness. These parameters are `luma_spatial`, `chroma_spatial`, `luma_tmp`, and `chroma_tmp`. Each accepts values from `0` to `255`. `luma_spatial` and `chroma_spatial` affect the dissipation of static noise (the noise that is analyzed in each frame without taking other frames into account). `luma_tmp` and `chroma_tmp` describe the treatment of noise that shows up through multiple frames.

Denoising is a destructive action. To remove the grain and video defects, the video is smoothed and can become blurry if an aggressive filter is used (Figure 3). Remember that fine details, such as hair or wrinkles, may be mistaken for video noise and subsequently removed by the filter. For this reason, it is better to use conservative values for the denoiser. Hqdn3d's documentation recommends not using values greater than `10` for the `luma_spatial` and `chroma_spatial` and than `13` for `luma_tmp` and `chroma_tmp`. These suggested limits are still very high. You might want to experiment with different numbers until you achieve a result you like.

Patent-Encumbered Codecs

H.264 and AAC video and audio compression standards, respectively, are certainly unsuitable for hard-core FOSS proponents who live in regions where software patents apply or plan to distribute content to those regions. If codec licensing or patent trolling is a problem, alternative codecs can be used.

VP8, Google's attempt to establish a patent-free competitor to H.264, is good enough, but it has two problems. First, it is less popular, so you are less likely to find support for it in home appliances. Second, Google has the bad habit of booting projects up, backing them with lots of resources, then suddenly realizing they are not profitable and dropping them altogether. This is the current state of VPX codecs. While VP8 is still useful, you can't count on Google to update the specification or their libraries.

Vorbis and Opus are good lossy audio codecs that can replace AAC, but as with VP8, they lack support from many domestic multimedia appliances.

Matroska is an open multimedia container that does not need replacement to keep your movie collection FOSS friendly.

Listing 4: Example Metadata File

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Tags SYSTEM "matroskatags.dtd">
<Tags>
  <!-- movie -->
  <Tag>
    <Targets>
      <TargetTypeValue>50</TargetTypeValue>
    </Targets>
    <Simple>
      <Name>TITLE</Name>
      <String>Happy Rottweiler</String>
    </Simple>
    <Simple>
      <Name>DIRECTOR</Name>
      <String>Rubén Llorente</String>
    </Simple>
    <Simple>
      <Name>DATE_RELEASED</Name>
      <String>1992</String>
    </Simple>
    <Simple>
      <Name>ORIGINAL_MEDIA_TYPE</Name>
      <String>VHS</String>
    </Simple>
  </Tag>
</Tags>
```

```
    --set "title=Happy Rottweiler" ➥
    -t global:metadata.xml ➥
    --edit track:a1 ➥
    --set "language=spa" ➥
    --set "flag-default=1"
```

This command sets the file's title *Happy Rottweiler*, merges the metadata file `metadata.xml` with the `vhs_final.mkv` file, and sets the audio track number `1` language value to Spanish. The only audio track is set as the default audio track.

Conclusion

Saving your old VHS tapes or audio cassettes is quite easy with minimal hardware expenses. Although video and audio encoding and analog to digital conversion is surrounded by a lot of black magic, this article should provide a starting point to converting your old VHS tapes for personal use.

You can easily use the procedure presented here to save audio cassettes just by capturing and encoding the audio by selecting a suitable codec, such as Ogg, FLAC, or Opus, and a suitable container. Matroska does the trick (in fact, files with the `mka` extension are Matroska files with audio-only content), but its use for this purpose is not widespread.

If the steps presented here look too cumbersome to perform manually, you can always use the LinuxTV's V4L capturing script [8] to automate the process. ■■■

Info

[1] PAL region: *https://en.wikipedia.org/wiki/PAL_region*

[2] NTSC region: *https://en.wikipedia.org/wiki/NTSC#Countries_and_territories_that_are_using_or_once_used_NTSC*

[3] Different EasyCAP devices: *https://linuxtv.org/wiki/index.php/Easycap*

[4] Why GStreamer is better at encoding: *https://www.linuxtv.org/wiki/index.php/GStreamer#Why_is_GStreamer_better_at_encoding*

[5] Compatibility: *https://trac.ffmpeg.org/wiki/Encode/H.264#Compatibility*

[6] Keyframes: *https://en.wikibooks.org/wiki/MeGUI/x264_Settings#keyint*

[7] Matroska metadata examples: *https://www.matroska.org/technical/specs/tagging/example-video.html*

[8] V4L capturing script: *https://linuxtv.org/wiki/index.php/V4L_capturing/script*

FOSSPicks

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

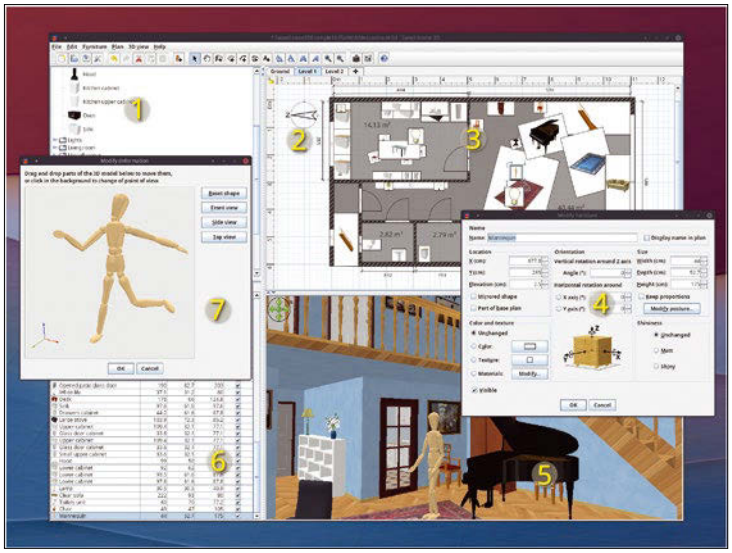
Graham nearly made it through an entire month of FOSSPicks without an esoteric audio discovery. And then he found VeeSeeVSTRack. BY GRAHAM MORRISON

Home designer

Sweet Home 3D 6

Looking at a screenshot of its Java-driven UI, and perhaps at its name, it's easy to dismiss the home designer Sweet Home 3D as something slightly gimmicky that might be more useful as a toy or for entertainment. But this is definitely not the case. In fact, we know people who have used this software to design a real home and then go out and build it, from the outside structure right through to the inside room layout. Not only does it let you design the external structure of your buildings using exact measurements, it lets you plan your living space, filling it with furniture and all the other things you'd expect. This is important, because, if you've oriented your building correctly, it allows you to see where the light will fall, your view from a certain position, or whether you can squeeze past the grand piano into the kitchen. If you need to know where to place the windows to see the mountains from your bedroom, this is the application to use.

It all starts with a simple plan. You create rooms by creating 2D polygons, just as you would with a drawing app like Inkscape. Select *Add walls* and you can snap vertical walls onto your creation. As you make all these changes, the 3D preview updates in real time, so you can see what your abode will look like from a first-person view. This extends to when you start furnishing your creation by dragging and dropping elements from the vast library of models that are categorized by room. There are two living room armchair types, for example, and three staircase types. Double-click on one of these objects, and you can change its size, its orientation, its shininess, and even its texture and material. It's easy to lose hours trying to get everything exactly right, or changing options to see what things look like in a different color or with a different theme.



1. Model list: A vast number of models can be dropped into your room designs. 2. Plan: Create your perfect layout and then add 3D walls. 3. Doors and windows: Work out exactly where best to place doors and windows for access. 4. Modify models: Scale and rotate your models according to your mildest whims. 5. 3D view: The first person view lets you walk around your dream house. 6. Exact dimensions: Drag and drop elements or manually enter their dimensions and location precisely. 7. Mannequin: Drop a mannequin into your scene for scale and space.

Sweet Home 3D has been in development since 2005, which is why it has such a rich set of features and such a huge library of objects. Version 6 is a major update with lots of new models, many replacing older and more dated versions, all released under a GPL or CC-BY license. This added detail extends to changeable screens on the laptop model and the picture frame, and even a mirror that actually reflects. But our favorite is the mannequin object that you can drop into your scene. Double-click on this and select *Modify deformation*, and you can drag the mannequin's limbs around to put them into the exact position you need, whether that's lounging on the sofa or playing the arcade machine.

When you've finished designing your perfect home, the 3D view menu allows you to export the scene as an OBJ file. This can then be used by other 3D applications, such as Blender, to create a photorealistic render of your home. Or as we did, load it in Unity 3D and use SteamVR to walk around your dream house in virtual reality – a perfect, and cheaper, way to make sure your designs make sense.

Project Website
http://www.sweethome3d.com/

Virtual modular synth

VeeSeeVSTRack

Create and run the most powerful modular synthesizer directly within your favorite audio apps.

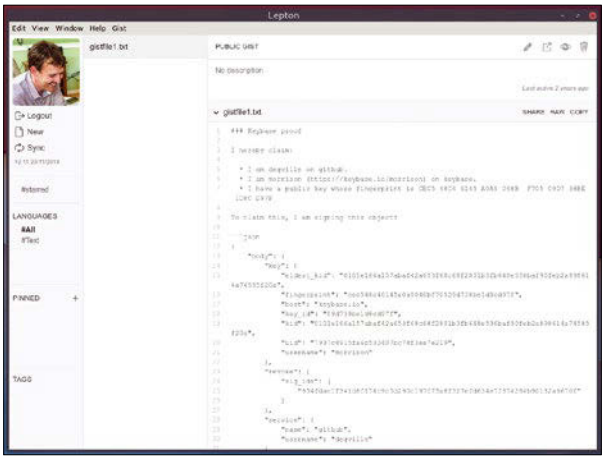
We were one of the first publications to cover VCV Rack, the open source modular synthesizer platform. Releases have become prolific, feature-packed, and even commercial, and we have to hold back from featuring it every single issue. If you've not used VCV, download it now. It allows you to plug virtual recreations of real hardware into a virtual rack and wire them to create any signal path you wish, from crazy experimental music, to deep and convolving reverb effects. It's incredible, and it's not surprising that since we first featured it, VCV Rack has become one of the most important open source audio projects of the last decade, with many module makers releasing virtual versions of their own, often open source, models.

But as great as VCV is, it doesn't integrate too well with the latest Linux audio applications. In particular, it's a standalone application rather than a plugin for your audio application of choice. On Windows, a developer known as "bsp" has put considerable effort into making VCV Rack work as a VST plugin, resulting in VeeSeeVSTRack. To handle the multiple instances of VCV, bsp even needed to create a custom drawing library, but it was worth it. On Windows, you can now use VCV directly within your favorite audio application, rather than always having to find a way of piping audio into your chosen tool. And with help from bsp, this wrapper and the new drawing library have been ported to Linux, giving us access to the most powerful audio processing tool you can imagine, directly within your VST-compatible host. VeeSeeVSTRack is amazing, and we had no trouble dropping the pre-build binary into our VST plugin path and using it within Ardour. But it should also work with other Linux VST compatible apps, such as Bitwig Studio and Qtractor. Give it a go!

Project Website
https://github.com/bsp2/VeeSeeVSTRack/

Gist editor

Lepton



Gists are a great way of sharing code ideas or keeping your own notes.

If you've been on GitHub at all, you'll have noticed that something often referred to as "gists" have become very popular. These gists are a kind of coding note or recipe that consists of either a single file or part of a file that you keep aside or share. However, they don't need to be code; they can be anything you can conveniently note down using Git and Markdown. A gist could just as easily be the beginnings of a novel as it could be a script for clearing out your unused Docker containers. There are public gists that can be discovered and searched, and private gists you can keep to yourself or whoever you happen to share the link with (gists are never totally private). But there isn't a convenient way of working with your gists outside of the GitHub environment, and that's exactly what Lepton provides from your desktop.

When first launched, Lepton asks you to authenticate the use of the application with your GitHub account. With that done, you see a simple interface split into a main view for your public and private gists and their descriptions, and a panel on the left that allows you to sort through a collection using tags, languages, and whether they're pinned. All of this is basically mimicking the functionality of the web interface, which isn't surprising considering this is an Electron application running React. But as with the average Electron Twitter client, it feels better integrated with your desktop when you have a single window – especially when you start to learn the keyboard shortcuts – and the Markdown editor is much nicer than the web version, especially when you fold the gists you're not working on away.

Project Website
https://github.com/hackjutsu/Lepton

Command-line charting

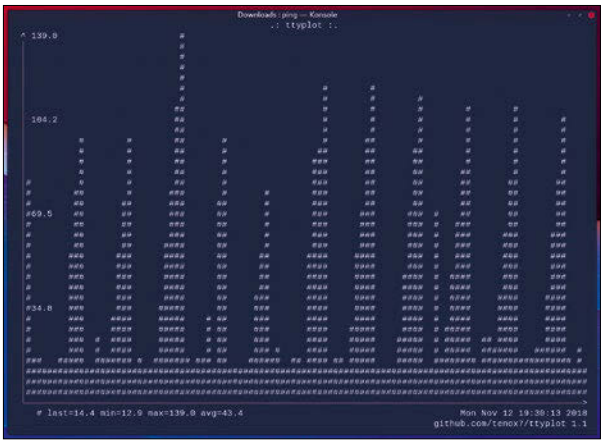
ttyplot

A month doesn't go by without the release of a new command-line monitoring tool. This isn't a bad thing, because the command line forces developers to think differently if they want to compete in an already saturated space. And this is what `ttyplot` does. It's not exactly a monitoring tool, at least not in the way you think of something like `htop`. Instead, it's more of a general tool like Gnuplot, albeit without the huge learning curve. That's because, while it does display data on a chart, it doesn't define what that data should be. Instead, it's intended to sit at the end of a command-line pipe, charting the numbers that arrive at its standard input, much like a multimeter might track incoming voltage. With just a few command-line options to handle two plots, the characters used to render charts, and hard and soft values, it's very easy to use. Mostly you just throw numbers at it and monitor the output.

One of the best examples comes from using `ttyplot` with the `ping` command. If you reduce its output to a single floating-point number, such as with

```
ping 8.8.8.8 | sed -u 's/^.*time=//g; s/ ms//g'
```

and then pipe this into `ttyplot`, you'll see a histogram being plotted in real time as the latency of the remote server changes. It's perfect for the average embattled sysop who still needs to monitor response time and availability, especially if you need to tweak the exact kind of statistics you want to track. And if you want to track more than one, simply use something like `tmux` to run as many instances as you need, in whatever configuration best suits you.



Plot anything you can gets numbers out of, including ping times, audio volume, and crashing bitcoin values.

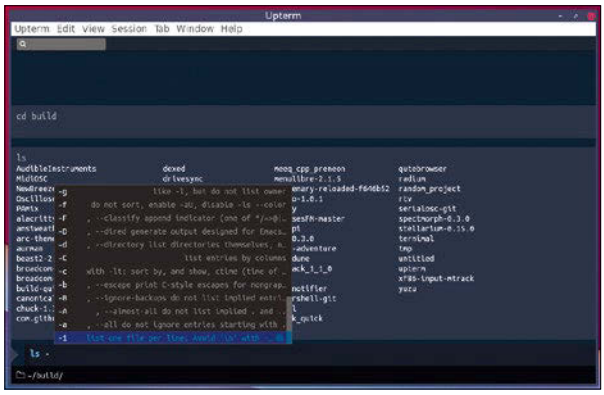
Project Website
https://github.com/tenox7/ttyplot

Terminal emulator

Upterm

The next level up from debating which is the best Linux desktop is debating which is the best shell. Bash is often the default, and for most of us, there's seldom a reason to try anything else. It's perfect for all kinds of tasks, from editing text to manipulating files. But there are many other shells that can be more helpful when you have more specific or technical requirements. Zsh is a very popular option, for example, because it expands on Bash's capabilities and can use a very helpful plugin system, often used by developers to streamline their workflow. Upterm is another and newer alternative, but one that's on the cutting edge of features whilst possibly being of most use to command-line beginners. Running atop Electron from the Node.js package manager, Upterm is never going to replace the standard Linux default running on your freshly booted server's framebuffer, but it may help if you spend a lot of time on the command line from the desktop.

Upterm describes itself as both a terminal emulator and an interactive shell. The interactive part is the first thing you notice. As you type `cd` to change directory, all the possibilities magically appear like a Google autosuggestion. This is really useful and potentially more user-friendly than having to press the Tab key to get the same list, but it also lists command arguments and, most impressively, the single sentence includes text hints for most arguments. This means that Upterm helps with both navigation and how you use the tools you want to type, which is ideal for beginners or for those of us who forget the command-line nuances of `git` or `vim`.



If you need some help learning command-line commands and syntax, Upterm will give you hints every step of the way.

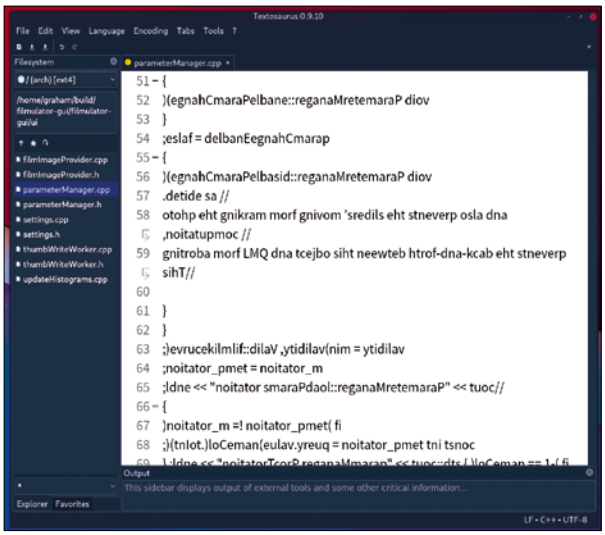
Project Website
https://github.com/railsware/upterm

Text editor

Textosaurus

Yes, there are many, many text editors for Linux. Almost as many as there are performance-monitoring tools, although we've been unable to find a link between the two. But it's also true that there isn't a Linux equivalent to the venerable Notepad++ on Windows. Nano gets close, but it requires you to use the command line, making Nano not the easiest editor to use. There's also the aging Gedit, but now there's Textosaurus, which aims to be a new desktop equivalent. Thanks to being built on both Qt and the Scintilla text editing framework, it's completely cross-platform, so it can even replace Notepad++ on Windows as well as it does on Linux. It even features a very similar layout and design.

To help with its cross-platform credentials, it uses UTF-8 internally, so your text should remain legible whatever platform or locale you're using, and many input encodings are supported. It also features menu options to convert end-of-line characters into something that works, which is often still a problem when working with text files generated in Windows. The syntax highlighting looks fantastic and will even print, while the UI remains very easy to use. You can move parts of the UI around, as you can with many KDE apps, but Qt and the bundled Scintilla are the only dependencies. There are lots of small utility functions too, such as a menu full of MIME tools, JSON beautifying, and Markdown preview. You'll also find advanced features, such as being able to record and play back macros, multiple cursor editing, and ligatures (special characters replacing multiple characters, such as "&"). Textosaurus feels like a lot of toolbox text functionality bundled into a fast and efficient text editor, which is perfect if you need the same editor on multiple platforms.



Textosaurus runs on both Windows and Linux and is packed full of features, although we're not sure why you'd use ***Invert Text***.

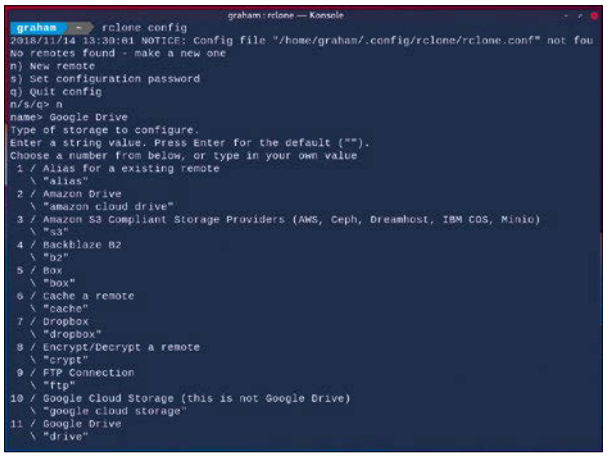
Project Website
https://github.com/martinrotter/textosaurus

Cloud backup

rclone

Whenever we write about backing up data, it's always with the caveat that it's a tedious but essential process. But two things have made it less tedious over the last decade – the first is cloud storage, as it means you no longer have to source your own off-site silo for your data, and the second is rsync. Rsync is an amazing tool that duplicates the contents of one filesystem to another and also does a great job of only pushing deltas rather than copying entire files each time. But it's complicated and requires considerable work if you need the source or destination to be somewhere in the cloud, and this is where rclone steps in.

Rclone's elevator pitch is "rsync for cloud storage," but to do this, it needs to know how to talk to lots of different cloud storage providers. Fortunately, it does – nearly 40 of them, including Dropbox, Google Drive, Amazon S3, Nextcloud, SFTP, WebDAV, and even your local filesystem. Adding a destination is an interactive process, started by typing `rclone config`, selecting *New remote* followed by your chosen cloud storage. This interactive process makes it easy to make the destination "read-only," for instance, and will typically open your default browser to allow you to authenticate the new client. It's quick and easy, and the excellent online documentation includes transcripts for each service if you get stuck. Backup is then as easy as typing `rclone copy` followed by the remote and local locations. Another brilliant feature supported on most clouds is server-side copy. This allows you to copy between two remote locations without the file or files passing through your local machine. It's perfect for transferring large amounts of data between Google and Amazon, for example, without affecting your local bandwidth at all.

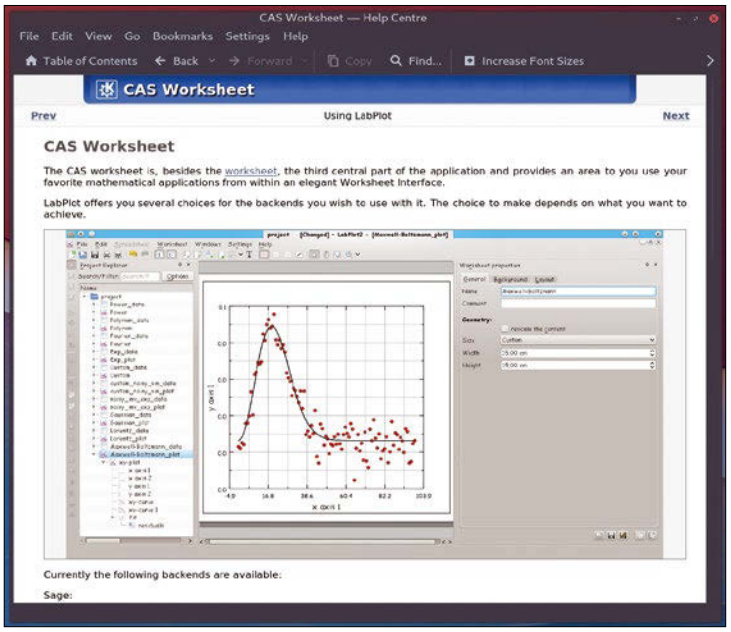


Apart from backup, one of the best uses for rclone is sending freshly scanned PDFs to Google Drive, where they'll automatically be processed by its fabulous OCR.

Project Website
https://rclone.org

Data management

LabPlot 2.5

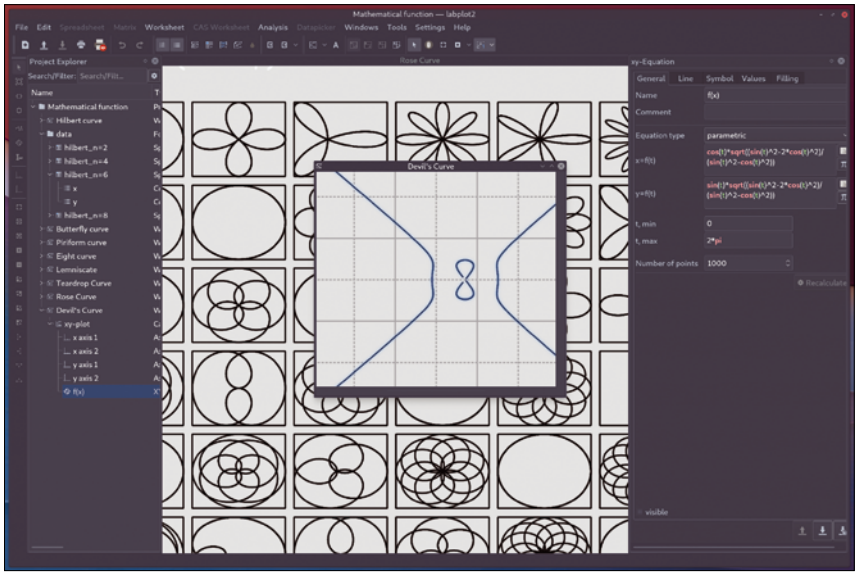


LabPlot is one of those rare open source projects with excellent documentation.

Whether it's your own list of comma-separated fermentation temperatures or the uptime of your server, you can visualize data on Linux many ways. The *de facto* choice for making visual sense of varied data sets is Gnuplot, but it's not the easiest tool to use. Gnuplot is typically run from the command line, and good output requires scripting. Plus, Gnuplot doesn't do anything to help you manage your data; if you're serious about tracking changes, you'll have multiple sets from the same sources. This is where LabPlot can help. It's a KDE application that's not only an effective GUI to your data plots, but also a GUI for managing multiple data sets and projects within a single application.

You start off by creating a new project. It could be for your homebrew temperatures or for your own laser-ranging retroreflector moon-distance experiment. Either way, LabPlot helps you keep both sets of data separate, as well as each set of readings, within a single project. But what's really clever about the way LabPlot manages a project is that it doesn't need to contain just data sets. In fact, you can choose from several different data source types, including matrices, spreadsheets, and CAS worksheets. The latter is a way of integrating external tools, such as R, KAlgebra, Lua, and Python, within LabPlot, and the wider "worksheet" type can be used to group different kinds of worksheet objects, such as plots and labels. You can also import the data you've already gathered, thanks to its support for ASCII, binary, HDF5, NetCDF, and FITS formats, with many options to control the import process.

When it comes to making a plot, you'll find cartesian, polar, and parametric equation types, and a brilliant functions browser – all missing from Gnuplot – that lets you browse many different function libraries; the standard set includes powers, cube roots, and sign functions, alongside other common options. Different libraries offer elliptic integrals, zeta functions, and many different distribution functions to help you make sense of your data. For data analysis, there's linear and nonlinear regression, Fourier transforms and filters, and various interpolation filters. You obviously still need to know what you're doing with data, but you don't necessarily need to know how to use LabPlot to get meaningful results without too much effort. The GUI works around the data hierarchy, much like programmer's code in an IDE. Worksheets are listed per project, and selecting one will allow you to change the size of the plot, the background, and layout. Selecting or creating a plot within a worksheet unlocks most of the toolbar, which is useful for zooming in and out, scaling, and labeling each axis. Output includes PDF, SVG, and PNG, as well as LaTeX, and you can choose to output either an entire worksheet or just a selected subset of data. It's a brilliant tool if you need to work with data.



LabPlot features 2D plotting, data analysis, and professional output, all with an excellent project management workflow.

Project Website
https://labplot.kde.org/

Action RPG

Flare: Empyrean Campaign

You know when a game opens with the soothing sounds of a lute that you must be engaging in a role-playing game (RPG). And that's exactly the case with Flare: Empyrean Campaign, a wonderful open source "action" RPG campaign built with the Flare game engine. Action, in this sense means that the game isn't turn based and plays out in real time. You move, cast spells, and engage in combat directly as the events happen. It's reminiscent of games like Baldur's Gate and, more closely, Diablo 1 and 2. You start by selecting your character and your character class; this affects your starting statistics and inventory. You're then dropped into the story.

The Empyrean Campaign features immersive and beautiful isometric environments that are the backdrop to exploration and adventure. You wander about discovering people and places, finding quests along the way. Quests usually involve attacking lots of skeletons and zombies in a certain location before fighting a final boss and returning to the quest source to get a reward. It's classic hack and slash and spell casting. Like Diablo, there are waypoints for fast travel, and you move between areas when small arrows indicate a direction. Windows open to allow you to manage your inventory, trade with other characters in the game, or see your character stats. The game mechanic should feel very familiar if you've played these kinds of games before. Flare, the game engine behind the game, originally started out as an engine for the first Flare game, now imaginatively called "Flare the Game." The game engine has been split from the old projects to enable projects like the Empyrean Campaign to be made. And if this campaign is anything to go by, this split has been hugely successful.

The Empyrean Campaign is built atop the Flare game engine, which is a great way to build Diablo-like games for yourself.

Project Website
http://flarerpg.org/index.php/mods/flare-empyrean/

Command and conquer

OpenRA

In a previous issue, we looked at a game engine called OpenDUNE that allowed you to play the original Dune II real-time strategy game on a modern Linux computer (and lots of other things besides). It now seems only fair to look at OpenRA, because it does the same for Westwood's famous sequels to the original Dune II – Command & Conquer (C&C): Red Alert and Command & Conquer: Tiberian Dawn, along with Dune 2000. All of these are classics and stay true to the original minimal design and maximum playability of the original Dune II. But there's another good reason to look at OpenRA now: The original games' publisher, EA, has announced it intends to release remastered versions of those games, and they've been petitioning the OpenRA community for ideas and feedback. That certainly makes a change from the usual approach of legal threats over intellectual property.

Like OpenDUNE, though, you still need to own and have access to the data files that came with the original games, although the game does offer to download the files for you when you first launch it. If you've not seen Red Alert for a couple of decades, it's quite a shock if you're using a high-DPI display; it's like you're looking at the game from orbit. Fortunately, the screen resolution can be changed, or you can run in windowed mode, and the games themselves don't feel so old at all. Like many old games, the gameplay itself has been very finely tuned because you couldn't fall back onto a 3D cutscene, and it's a breath of fresh air playing it in the same era of Red Dead Redemption 2, with its hundreds of miles of populated wild west, playable casinos, flora, and fauna. Unlike that blockbuster, OpenRA can still be modded, and there are plenty of terrains, scenarios, and games hosted by the community, which can only grow larger with the reissue of remastered originals. Fingers crossed.

There's still a sizable community playing C&C, which means you can find a game online 20 years after its release.

Project Website
https://github.com/OpenRA/OpenRA

Custom Shell Scripts

You do not need to learn low-level programming languages to become a real Linux power user. Shell scripting is all you need.

BY MARCO FIORETTI

For most people, shell scripting is all you need to become a very, very powerful Linux user: Text processing, backups, image watermarking, movie editing, digital mapping, and database management are just a few of the many tasks where shell scripting can help you, even if you are just an end user.

This tutorial is the first installment in a series that explores why and how to write shell scripts – whatever your Linux needs and skills are.

No previous shell experience is necessary to use this tutorial. This installment describes what shell variables are, how the shell handles them, and some simple ways to use and process shell variables to get the job done. Before getting started, however, a brief introduction to a few fundamental concepts is necessary.

Scripts or Programs?

Software programs can either be compiled or interpreted. Applications like Firefox, digiKam, or LibreOffice fall in the first category. Compilation makes programs much faster, because it translates their human-readable source code to low-level instructions directly executable by a processor. Compiled programs, however, are also much longer and more difficult to write.

Interpreted programs, which are also called *scripts*, are more limited and often slower. However, the learning curve and development time is remarkably shorter and smoother because, when launched, their source code is passed to a command interpreter, which parses and executes it one line at a time. This spares you from dealing with memory management, full declaration of variables, and a bunch of other low-level issues.

The default command interpreter on most GNU/Linux distributions, which is also available for other operating systems, is the Bash shell [1]. To be precise, "shell" refers to a whole category of interpreters. This tutorial covers Bash and uses the terms "shell" and "Bash" interchangeably, because Bash is the default shell in Linux. However, most of what you will learn in this series will be usable, without modifications, with all the other shells you can easily install on Linux systems.

You can use Bash interactively or in scripts. Interactive mode is what happens at any Linux command-line prompt, unless you set your Linux system to use another shell.

Working at the command line is enough for quick, one-time actions, but the real power of a shell, and the topic of this tutorial, is automation. You can save long sequences of instructions in a plain text file, and the shell will execute it as *one* software program. All you need to do to make that happen is to mark the file as executable (with the `chmod` command) and give it the right header. To see what I mean, here is the Bash script version of the "Hello World!" program:

```
#! /bin/bash
echo "Hello World!"
```

Save it into a file called `hello` and type `hello` at the prompt, and it will answer just that: *Hello World!*. The first line is what marks the whole file as a script to be executed by the Bash interpreter, which on Linux is the compiled program installed as `/bin/bash`. The second line simply tells that interpreter to launch the `echo` program to output the *Hello World!* string.

Experimenting with Shell Variables

The Bash environment makes available both predefined parameters and ways to create and process custom parameters. Strictly speaking, for Bash, a parameter is "an entity that stores values," and a variable is a parameter identified by a name.

The shell syntax for variables may seem weird, but it is easy to use. To assign a value to a variable, just write its name. To use it, you must prepend a dollar sign to the name. In both cases, re-

member to avoid spaces before and after the equals sign:

```
MYNAME=Marco # value is assigned
YOURNAME=$MYNAME  # value is *used*
```

(Note: Throughout the series, `#>` means the Linux command-line prompt; whatever follows the prompt is anything you choose to type there, not just into a script, to quickly check for yourself what you've learned.)

Besides a value, variables can also have attributes that specify, or limit, what you can do with them to avoid errors. A variable's attributes must be declared before using it in any way. The statement

```
declare -i WEIGHT
```

declares that the variable `WEIGHT` can only have integer (`-i`) values. Consequently, these two statements, which have the same meaning for a human reader,

```
#> WEIGHT=80
#> WEIGHT=eighty
```

do not mean the same thing to the shell; only the first statement will be accepted and executed by the shell. The other will cause an error, instead, which prevents you from wrongly assigning a character string to something that should only store integers.

Another handy attribute is `-r` (`readonly`):

```
#> declare -i -r WEIGHT=90
```

Adding that attribute makes any further attempt to assign values to the `WEIGHT` variable fail:

```
#> WEIGHT=70
bash: WEIGHT: readonly variable
```

Special Characters and Variables

The Bash shell has a lot of predefined variables and special characters. So many, in fact, that there is no way (or need, frankly) I can describe them all in one article. For now, I will only show a few samples to give you an idea of what is available:

- `$PWD` and `$OLDPWD` store the working directory where the script is currently running and the previous directory (old) where it was before that, respectively.
- `$HOSTTYPE` and `$OSTYPE` contain the CPU architecture and operating system on which Bash is executing. On my own system, they have the following values:

```
#> echo $OSTYPE running on $HOSTTYPE
linux-gnu running on x86_64
```

The most used predefined shell variable is surely `$HOME`, which represents the home directory of the user running the script. The same value is also represented by the tilde character, making these two statements equivalent:

```
#> BACKUP_DIR="$HOME/backup"
#> BACKUP_DIR="~/backup"
```

The tilde character is not just a synonym of `$HOME`, however: `~+` is equivalent to `$PWD`, and `~-` to `$OLDPWD`.

Variables like `$HOME`, `$PWD`, and `$OLDPWD` may seem superfluous to a novice, but they are really important in many shell scripts. It is thanks to `$HOME` that a script always can save files in the same place (`$BACKUP_DIR`) – no matter in which directory the script is executed.

`$PWD` is equally useful, at least, whenever a script moves from folder to folder during execution, and you want to know each time it happens.

Previously, I mentioned that a shell interpreter parses your script one line at a time. Next, the shell splits the current line into *words* and then looks at them one at a time to decide what to do next. Actually, the shell does that in other cases as well, but I'll deal with those cases later.

To split a line, or anything else, into single words, the shell needs to know what, exactly, separates one word from another. The answer is in the special shell variable `IFS` (internal field separator): Any character sequence or any single character inside `IFS` marks the end of a word and the beginning of the next.

The default `IFS` value is the character triplet space, tab, and newline, but you can change the default to whatever you want.

This capability is extremely handy whenever you need to separate fields inside each line of a CSV file or in any generic string.

Here is an example, taken straight from my recent *Linux Magazine* tutorial on how to embed dynamic headlines in any Linux desktop [2]. If you have a variable called `HEADLINE` that contains several fields separated by the pipe character (`|`),

```
Red Hat Reports $823 Revenue|Navarro: ➲
  Kavanaugh should step aside|Debian, ➲
  Ubuntu... Leaving Users Vulnerable
```

you can save each of those headlines separately, with the single command

```
IFS='|' read -r -a NEWS <<< "$HEADLINES"
```

which, albeit cryptic, means:

1. Set `IFS` to the pipe character.
2. Split `$HEADLINES` into separate fields, delimited by `$IFS`.
3. Copy each of those fields into a separate element of an array called `NEWS`.

I will cover arrays and other more complex shell data structures in the next tutorial installment. For now, I'll look at another great property of the `IFS` variable.

If the `IFS` value is null, no word splitting occurs. At first glance, this may seem a totally useless, or irrelevant, property. In practice, the opposite is true. Setting `IFS` to null is the standard Bash way to read and parse text files, one *complete* line at a time. This other snippet of code, from my previous tutorial [2], shows how to do it:

```
while IFS= read -r  line
do
# process the current COMPLETE line, saved in $line
done < $SOMEFILE
```

Because `IFS` is set to null (because there is nothing immediately after the equals sign), each full line of text from the file `$SOMEFILE` is loaded as a single string into the shell variable `$line`. If `IFS` has any other value, each line of `$SOMEFILE` will be split into different words before your script ever knows what the whole line looks like.

Variables and Quoting

You can play with shell variables in different powerful ways. Very likely, most of the time you will just need to *quote* those variables properly, so I'll look first at the three kinds of quotes you can use in Bash.

Single quotes (') tell the shell that everything inside them should be used "as is," without any processing. Double quotes (") tell the shell to replace the name of every variable inside the quoted string with its current value. Finally, backquotes (`) tell the shell that the content of the string is a command that should be executed (but only after replacing variables with their values, as in the double quotes case!). An alternative syntax for backquotes is:

```
#> $(command)
```

The difference among the three types of quotes is evident in the following example, which you can try out on your system:

```
#> echo '$PWD'
$PWD
#> echo "$PWD"
/home/marco/
#> echo `$PWD`
bash: /home/marco/: Is a directory
```

The last command causes an error because the backquotes make the shell grab the value of `$PWD` (`/home/marco/`) and then execute `/home/marco/` as if it were an executable program, which of course it is not.

Basically, variables in backquote-delimited strings let you build and execute different commands every time they are parsed, depending on the current status of (possibly many) other variables. This is a very powerful feature, especially when you consider that you can nest those strings into each other. In that case, remember to escape the inner backquotes with backslashes.

When quoting, you also have to be careful when you concatenate variables and constant strings. The reason is shown in these commands, which try to assemble a date from its year, month, and day components:

```
#> YEAR='2018'
#> MONTH='11'
#> DAY=10
#> echo "$YEAR$MONTH$DAY"
20181110
#> echo "$YEAR11$DAY"
10
#> echo "${YEAR}11$DAY"
20181110
```

The first `echo` statement works as expected: The shell just concatenated the values of `YEAR`, `MONTH`, and `DAY` to create a date in `YYYYMMDD` format. The second `echo` only prints `10` because this time the shell only sees two variables, namely `$DAY` and `$YEAR11`, which is undefined. To concatenate a combination of variables and constant strings, you must enclose the name of the variable in braces, as shown in the third `echo` command.

You can make a variable point to another by combining quotes with exclamation marks:

```
#> MYNAME=Marco
#> HISNAME=MYNAME
#> echo "his name is $HISNAME"
his name is MYNAME
#> echo "his name is ${!HISNAME}"
his name is Marco
```

Shell Expansion

The first thing the shell interpreter must do every time it parses a single whole line of a script is to expand it. Expansion consists of looking at each part of the line that is not a Bash built-in command to determine if and how it should be transformed into something else.

This procedure is needed because the actual content and structure of the current line might be different every time that line is parsed. The simplest example of this situation is a line using a variable

that corresponds to the current time. In other words, the interpreter cannot know what to do with a line of script without fully expanding it first.

The Bash man page [3] says, "there are seven kinds of expansion performed, in this order: brace expansion, tilde expansion, parameter and variable expansion, command substitution, arithmetic expansion, word splitting, and pathname expansion."

This may seem messy, but you already saw most of these expansions: parameter and variable expansion, command substitution, word splitting with `IFS`, pathname expansion, and tilde expansion with `$HOME` and `~`.

Brace expansion lets you create sequences, or variations of strings with the least possible typing:

```
#> ATTENDEES=
  `echo 'Mr '{John,Frank,William}' Rogers'`
#> echo $ATTENDEES
Mr John Rogers Mr Frank Rogers Mr William Rogers
```

Another part of shell parsing that may be considered equivalent to expansions is the way backslashes are handled.

Escape characters (i.e., characters preceded by a backslash) have one of two special meanings. If they are alphabetic characters, they are "escape sequences" that tell the shell to perform some action. For example, `\a` means "ring the alert bell," `\n` means "go print to the next line,", and `\t` means "print a horizontal tab."

If the backslash appears before other backslashes and quotes, it tells the shell just to print those characters, without interpreting them: If you type `\\`, `\'`, or `\"`, the shell will just see and use a single backslash, a single quote, or a double quote, respectively.

Math with Variables

The Bash shell can use variables to do math. By no means is Bash the best tool to do this, but it can do enough things to spare you using another program when you need to do some quick calculations, perhaps as part of a bigger shell script. I will discuss shell math in a later installment, but initially I want to introduce one numeric shell variable, `$RANDOM`, for two reasons. First, I want to show an example of arithmetic expansion. Second, `$RANDOM` can be quite useful, even in scripts that are not about numbers. Listing 1 shows a small script using `$RANDOM` taken from a web page [4] entirely devoted to this variable.

Every time you invoke it, `$RANDOM` returns a random integer number between 0 and 32,767. The other variables `$FLOOR`, `$CEILING`, and `$RANGE` "translate" that range to make `$RESULT` a random number between 70 and 230. Lines 4, 6, and 7 show an initial partial example of how to perform math in Bash via arithmetic expansion.

Listing 1: $RANDOM script

```
01 #! /bin/bash
02 FLOOR=70;
03 CEILING=230;
04 RANGE=$(($CEILING-$FLOOR+1));
05 RESULT=$RANDOM
06 let "RESULT %= $RANGE";
07 RESULT=$(($RESULT+$FLOOR));
08 echo $RESULT
```

Text Processing with Variables

The most popular interpreters, and corresponding scripting languages for processing text strings, are probably Perl and Python. The current versions of Bash, however, include enough string processing operators to be more than adequate for the simplest and most common tasks. The main ones are:

```
#> NAME=Marco
#> echo ${NAME%co}
Mar
#> echo ${NAME/co/36}
Mar36
#>  echo ${NAME^^c}
MarCo
#> NAME=MARCO
echo ${NAME,,*}
marco
```

The statements above show how the shell can perform removal, substitution, and case modification, respectively, on parts of a variable.

The `%` operator removes the shortest matching pattern (`co` in the example) from a variable, starting from its end. The slashes delimit which part of the string should be replaced, and which other string to use. The caret (`^`) and comma (`,`) operators convert any occurrence of the matched characters to uppercase (`c`) or lowercase, respectively (the asterisk is a shortcut for "any character"). All these operators have other formats and variants: To know how they work, check the Bash man page [3].

It is also possible to extract and replace specific parts of a string, whatever their value. I will show how to do it, together with another very useful feature of the shell, with a final, simple script (Listing 2).

This code is a really bare-bones formatter of names and phone numbers: No matter how you write them, the script returns name and surname with the first letter capitalized and the phone number in a standard format.

Listing 3 shows what you get when you run the script from Listing 2.

As is, the script is not complete, because it only formats one entry at a time. In a future installment, I will show how to work on multiple records. Even in

Listing 2: Replacing Specific Parts of a String

```
01 #! /bin/bash
02
03 FIRSTNAME=$1
04 SURNAME=$2
05 PHONE=$3
06
07 cat<<INITIAL_DATA
08
09 Initial Data:
10  First Name   : $FIRSTNAME
11  Surname : $SURNAME
12  Phone Number : $PHONE
13
14 INITIAL_DATA
15
16 FIRSTNAME="${FIRSTNAME,,}"
17 FIRSTNAME="${FIRSTNAME^}"
18
19
20 SURNAME="${SURNAME,,}"
21 SURNAME="${SURNAME^}"
22
23 PHONE_1="${PHONE:0:3}"
24 PHONE_2="${PHONE:4:3}"
25 PHONE_3="${PHONE:6}"
26
27 cat<<FORMATTED_DATA
28
29 Formatted Data:
30  First Name   : $FIRSTNAME
31  Surname : $SURNAME
32  Phone Number : ($PHONE_1)
   $PHONE_2-$PHONE_3
33
34 FORMATTED_DATA
35 exit
```

Info

[1] Bash: *www.gnu.org/software/bash/*

[2] "Tutorial – Desktop News Feeds" by Marco Fioretti, *Linux Magazine*, issue 217, December 2018, pp. 90-95: *http://www.linuxpromagazine.com/Issues/2018/217/Read-Me*

[3] Seven expansion types: *https://tiswww.case.edu/php/chet/bash/bash.html*

[4] Generate random numbers in a given range: *https://bytefreaks.net/gnulinux/bash/bash-get-random-number-that-belongs-in-a-range*

[5] Code in this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/219/*

this form, however, the code is useful as a first approach to text formatting and parsing, because the same tricks may be adapted, for example, to process generic spreadsheets in CVS format.

In Listing 2, lines 3 to 5 show how you can pass different parameters to a script every time you run it. The variables `$1`, `$2`, and `$3` contain the parameters that were passed in from the command line. You may use them directly, but for readability, it is much better to copy them into `$FIRSTNAME`, `$SURNAME`, and `$PHONE`.

Line 7 marks the beginning of a so-called "here document," which ends on line 14; `cat`, a Linux command, simply prints out whatever you pass to it.

A here document is a really cool way to use variables and generate shell output. You can think of a here document as a pseudo-file or, more appropriately, template that is embedded into a shell script. The template ends with a line containing the same string used to mark its beginning (`INITIAL_DATA`).

Listing 3: Output from Listing 2

```
#> phone-formatter LEia ORGaNA 5553236856

Initial Data:
                First Name   : LEia
                Surname      : ORGaNA
                Phone Number : 5553236856


Formatted Data:
                First Name   : Leia
                Surname      : Organa
                Phone Number : (555) 236-6856
```

Inside the template, you can put as many variables as you like, and every time you use it, it will contain the current values of all those variables.

You can use here documents to custom generate any kind of complex, pre-formatted data on the fly, from database records to sequences of commands to pass to any other program.

Lines 16 to 21 just apply the case modification tricks already presented, and lines 23 to 25 show how to extract substrings. The final part of the script uses another here document to display the reformatted input values.

Conclusion

In this installment, I have covered the basic components that make shell scripts so flexible and usable. Next month, I'll play with more complex data structures like shell arrays and hashes. In the meantime, I highly recommend that you copy the provided script [5] and experiment both with the string processing commands and the here document layout to determine how much you can customize them. Happy scripting! ■■■

The Author

Marco Fioretti (*http://mfioretti.com*) is a freelance author, trainer, and researcher based in Rome, Italy. He has been working with free and open source software since 1995 and on open digital standards since 2005. Marco also is a board member of the Free Knowledge Institute (*http://freeknowledge.eu*).

A Webzine for High-Performance Computing Specialists

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

ADMIN Network & Security

admin-magazine.com/hpc

Natron Nodes

Natron gives you the power to apply sophisticated effects to your videos, but its node-based interface can be a bit confusing. This tutorial will help you get a grasp on the basics.

BY PAUL BROWN

Let's get this out of the way: Natron [1] is not a video editor. The creators' website says it is "open source compositing software for VFX and motion graphics." This is one of those instances where saying what something is doesn't help understand what it does.

I'm guessing you are okay with the "open source" and the "motion graphics" (aka video clips) part of the definition, right? It is the "compositing software for VFX" that probably needs more explaining. VFX stands for "video effects." As for "compositing", it is just a fancy way of saying "mushing two or more things from different sources onto the same video frame".

Say you have a clip of your kitty, and you want to manipulate it so that it shows her shooting laser beams from her cute little eyes (Figure 1). Waiting for this to happen spontaneously would probably take some genetic or cyborg engineering. Instead, you would be better advised to use some movie-making magic. You could take the clip of your cat (source number one) and then create an animated clip of laser beams, maybe using 3D design software such as Blender [2] (source number two). With both of your sources, you then would use a compositing application to merge them together. That is what Natron does … among many other things.

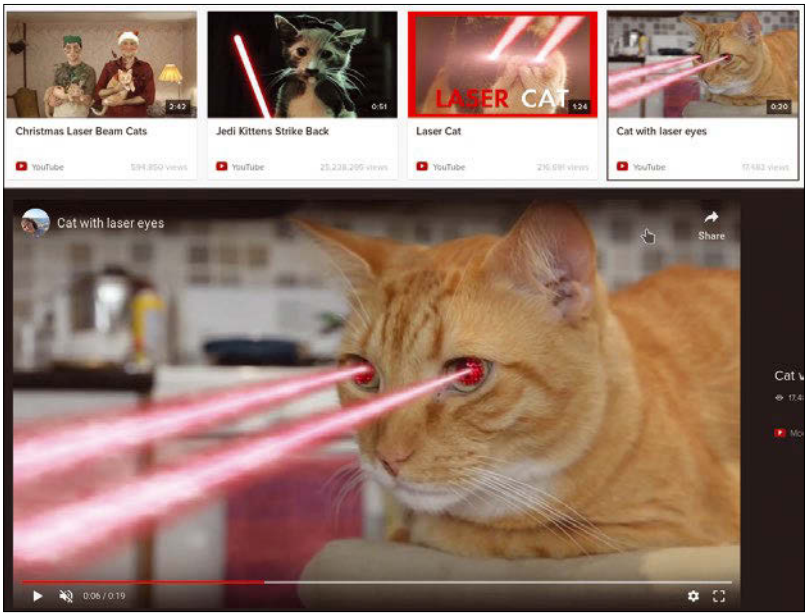


Figure 1: Of course, kitties shooting laser beams from their eyes is a thing on the Internet.

To recap: In Natron you work with individual clips, merge them, apply effects, and so on. You then paste the clips together to make full scenes in a video editor like Kdenlive [3]. Your workflow takes the following steps:

1. Raw footage
2. Compositing
3. Scene editing
4. Full movie editing

Getting Started

You'll find Natron in most repositories of popular Linux distributions: if not in the main repos, in specialized or user-maintained ones (think Ubuntu's PPAs or Arch's AURs). Or, you could just use Natron's own installer or Flatpak from the project's page.

Either way, once you boot it up, you'll be presented with a screen similar to what you can see in Figure 2 (see also the "Natron's Interface" box), except you will only see one node, the *Viewer1* node, in the *Node Graph* tab (bottom left).

To load your first clip, you need to add an *Image* | *Read* node, because everything in Natron is done with nodes. To add the *Read* node, right-click on *Node Graph* and pick *Image* and then *Read* from the pop-up menu. Or, you could go to the vertical menubar running down the left side of the Preview pane (top left). Either way, once you pick the *Read* node, Natron will open a file navigation window that will allow you to choose a clip, image, or sequence of images as a source.

To open a video file or static image, just navigate to it, select the one you want, and click on the *Open* button. To open a sequence of frames, make sure your files are numbered (e.g., `frame001.png`, `frame002.png`, `frame003.png`, etc.). Under the pane that lists your directories in the file navigator, you'll

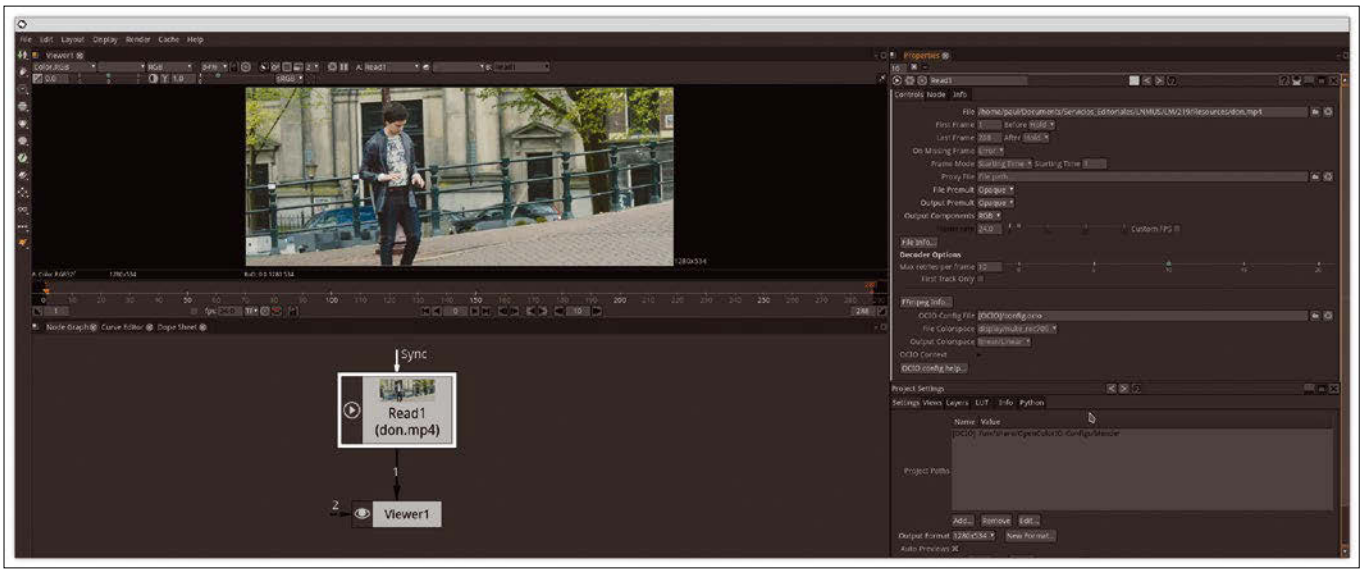


Figure 2: Natron with a clip already loaded.

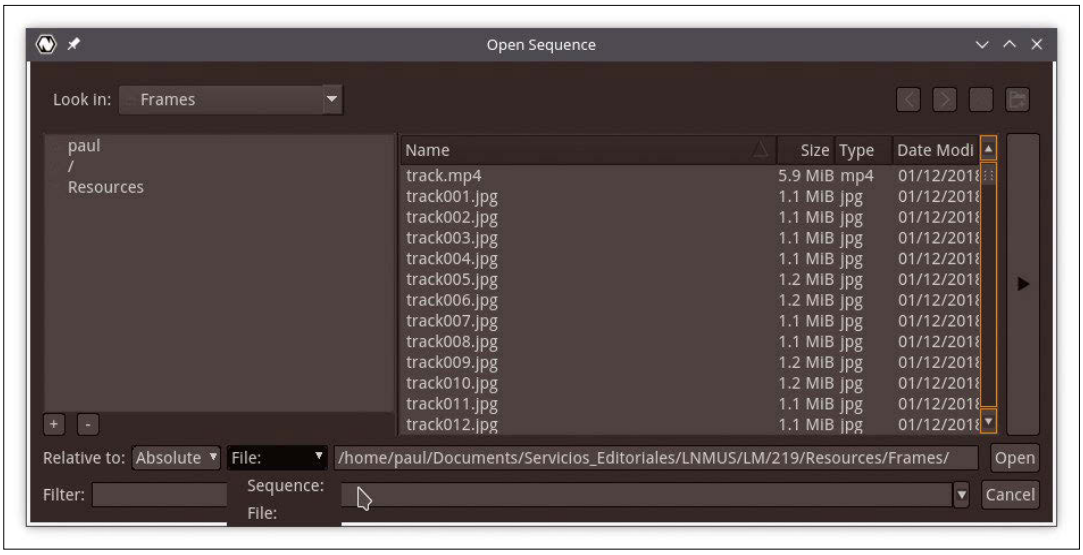


Figure 3: Choose *Sequence* In the file explorer to load a sequence of frames.

Natron's Interface

Natron's default interface is laid out broadly into three areas:

1. At the top left is the Preview pane. This is not a static area just for checking on your progress. As you will see later, tools and widgets appear here depending on which node you have selected at any given time. Scroll with the middle button on your mouse to zoom in, or click and hold it to zoom around.
2. Directly underneath the Preview pane is the Node Graph, where a lot of the action happens; you open your nodes here and chain them together to create the effects you need. As with the Preview area, you can zoom in on your nodes by scrolling with your mouse wheel or pan around by clicking and dragging with the middle button. Click on a node to select it; click and drag to move it. Click on an arrow's shaft and drag it to a node to connect the two. Click and drag on the shaft again to disconnect an arrow.
3. On the right, you have the node Properties pane, which is where you adjust parameters, tweak your nodes, and create *keyframes*. The boxes are piled one on top of the other in no particular order. Scroll up or down to find the one you are looking for, or double-click on a node in the Node Graph area to bring its property box to the top of the Properties stack.

You can move all these elements around to better suit your working style, but regardless of how you end up configuring your layout, I would advise using a *big* monitor. You are going to need the screen real estate.

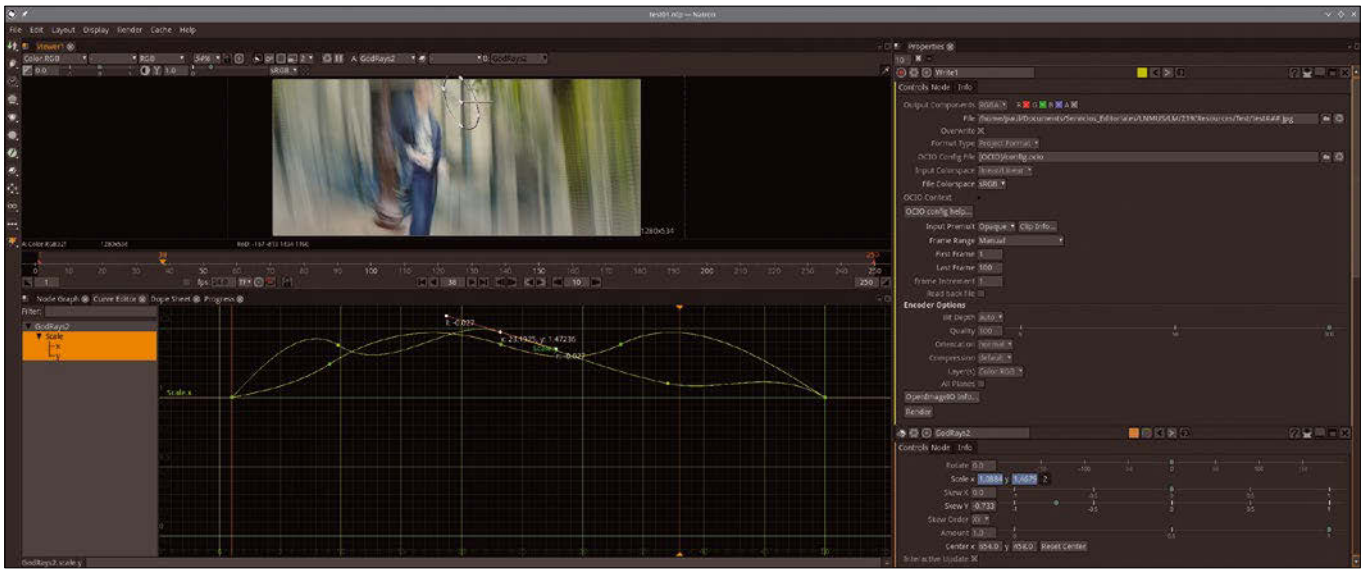


Figure 4: All effects are keyframeable. The ***Curve Editor*** shows how you can edit keyframes and tweak how they are connected.

see a drop-down menu with the *File* option selected. Click that and you'll also see the *Sequence* option (Figure 3). Click on that, and your list of frames will show up as `frame###.png` numbered from 1 to 500 (if you have 500 of them). Selecting and clicking *Open* will bring them in as a single file.

The *Read* node will appear showing a tiny thumbnail of your clip. Usually, if this is your first clip, it will connect automatically to the *Viewer* node, and the first frame will appear in the Preview area in the upper half of the window (Figure 2).

Once you have your source and viewer connected, you can start applying effects. Right-click on and try *Filter* | *GodRays*, for example. Apart from the node popping up in the *Node Graph* tab, a bullseye will appear in the center of the Preview area and a new property box shows up for the node in the Properties stack. You can drag and push the bullseye around on the Preview or pull on the handles and radii to make a frame look like it is exploding out from the bullseye's center. You can also use the values in the property box. Try changing the *Scale* value to *1.2*, for example.

The cool thing is that most of what you see in the node's property box is keyframeable. Use the arrows in the Preview player to move to the

Keyframes

A keyframe is used to control key moments in a digital animation. You set values for an effect or draw a character in a certain position in one frame and make it a keyframe. If you change the values or draw the same character in a different position in another frame further along in the timeline and make it a keyframe, your animation software will interpolate all the intermediate frames to make the animation look smooth.

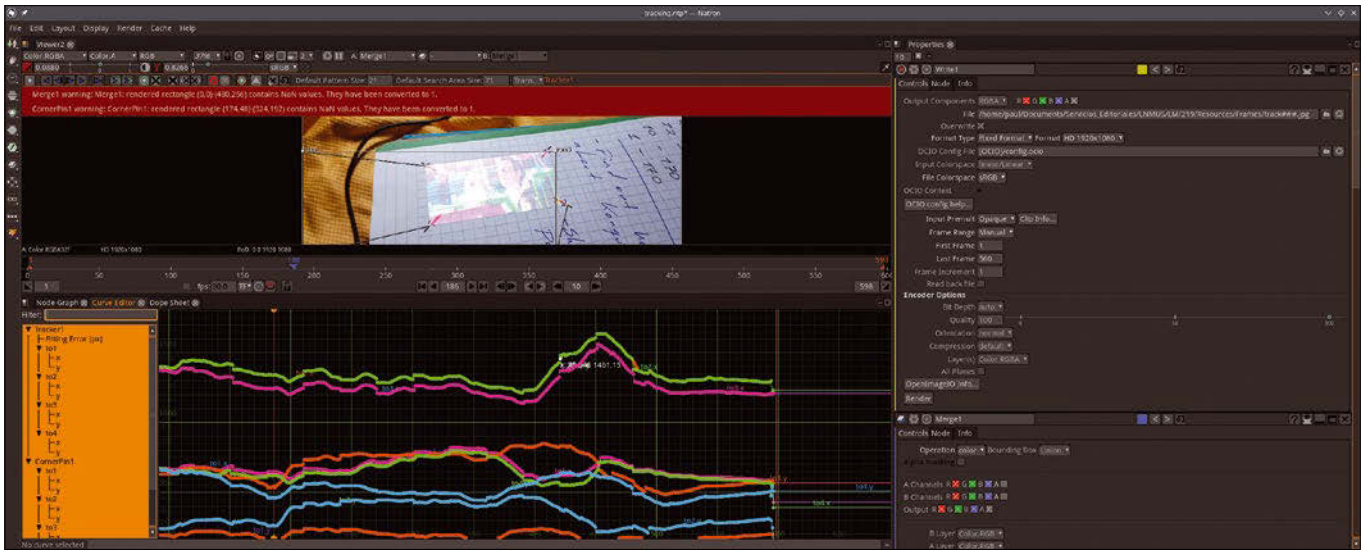


Figure 5: Click on the components on the left to fine-tune individual aspects of each effect.

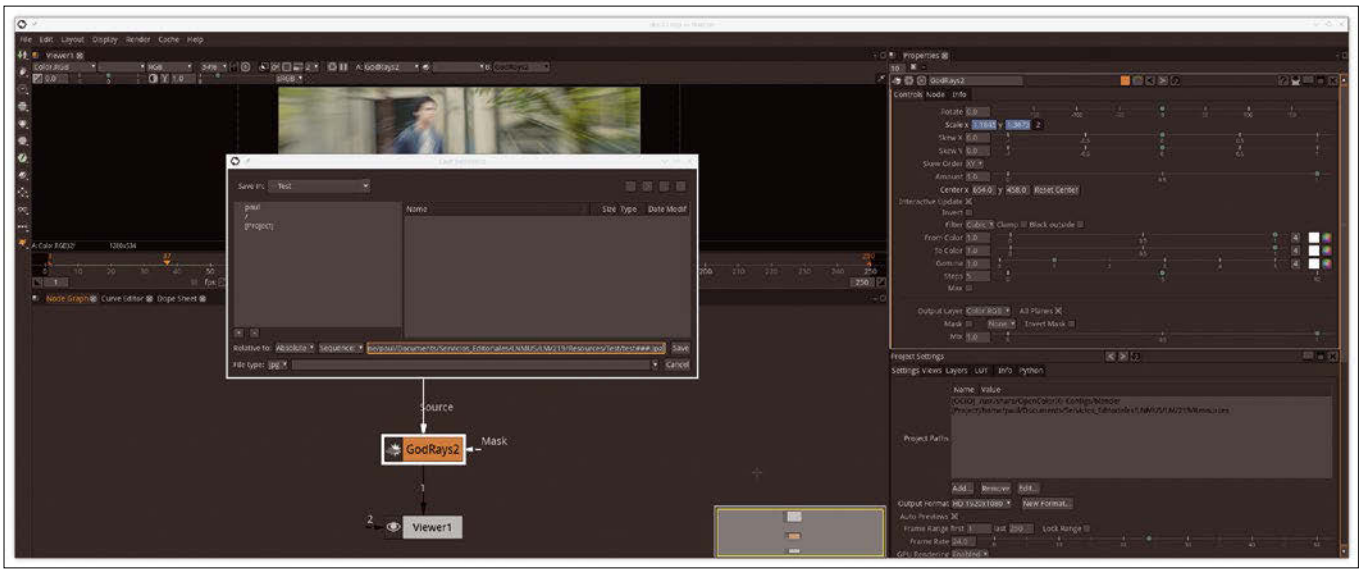


Figure 6: When you add a ***Write*** node, Natron asks you how and where you want to save your sequence.

first frame of the clip. Locate the *Scale* parameter in the GodRays' property box. Set it to *1* and right-click on the value. A menu will pop up. Select *Set Key* and the parameter's background will turn dark blue. Congratulations: You have just set your first keyframe (see the "Keyframes" box if you are unfamiliar with what these are).

Move to frame 50. You will notice that the parameter's background in the property box is a lighter blue. That means it is not a keyframe … yet. Modify the value so it is *1.5*, and the background will go to a darker blue indicating that, just by modifying the value, you have created another keyframe.

Play your clip back from frame 1 to frame 50, and you will see how GodRays progressively gets stronger as the video progresses (see Figure 4).

Figure 4 shows another cool feature: the *Curve Editor*. The next tab to the right of the *Node Graph*, the *Curve Editor* tab allows you to adjust the degree an effect is applied to keyframes (represented by a dot on the curve) by dragging them. You can also add more keyframes by double-clicking on empty space on the curve. The new keyframe will give you handles that allow you to adjust the curve.

Also notice how the GodRays' *Scale* component has two components in the *Curve Editor*: the *x* scale and the *y* scale. You can select and work separately with each – just click on the component you want; it will appear highlighted in orange and, the other components will disappear.

All of these features give you an OCD-worthy level of control over how you want each effect to look at every point of your clip. Being able to

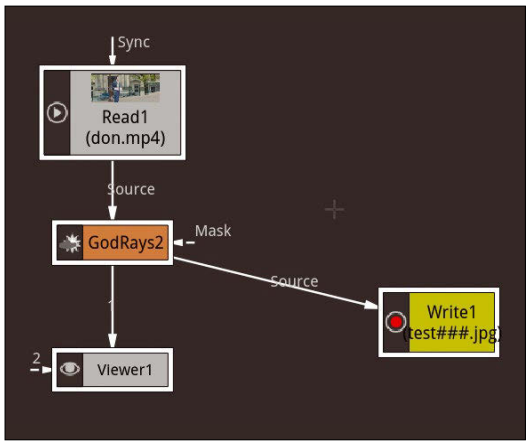


Figure 7: Link the ***Write*** node to the last node before the ***Viewer*** node.

highlight the exact effect you want to change is also very useful when you have a lot of them interacting in a complex way (Figure 5).

Rendering

At some point, you will want to convert your clip into something you will be able to use elsewhere. For that, you use a *Write* node. Right-click in the *Node Graph* tab, and choose *Image | Write*. A file-navigator window will pop up and you can pick the folder and the name of the file you want save in the *File* text box (Figure 6).

Of course, once the *Write* node is in place (Figure 7), you can modify things to your heart's content changing the parameters in the node's box in the Properties pane. That said, the safest thing to do here is go with the default configuration. This will create a sequence of JPEG images in the folder of your choice. To make sure Natron doesn't complain, choose a name like `frames###.jpg`. The `###` part of the name will allow Natron to create a sequence of images with names like `frame001.jpg`, `frame002.jpg`, `frame003.jpg`, and so on.

You should go with the default configuration for two reasons: One is that not all combinations that show up in the *Write* node property box will work well together. If you change something like the file type, colorspace, and *Layer(s)*, it is quite likely you will end up with empty frames. The second reason is that you don't want to render to an MP4, OGV, WebM, or anything like that, because all these formats use lossy compression codecs. Remember that your output from Natron will usually be going into Kdenlive for the final edit, so the less information you lose from your clips, the better.

Besides, Kdenlive works just fine with image sequences. When you want to load in a new sequence as a clip to Kdenlive, just remember to tick the *Import image sequence* checkbox in the lower left-hand corner of the file navigator.

If you do want to preview your clip in a video player, FFmpeg makes a short job of converting the sequence with:

```
ffmpeg -i frames%03d.jpg -r 25 my_clip.mp4
```

Audio

Once you start exploring Natron, you might wonder where all the audio controls are. Simply put: There aren't any. Natron is exclusively designed for working with video. To synchronize your audio tracks and apply filters, again, you are better served by Kdenlive.

The Future

I'm not going to lie, folks: The future of Natron is bleak. The institution that offered support for Natron up until recently cut its ties with the project, and the two main developers have had to move on to other paid jobs. There is currently nobody willing to take over the project, despite how far the previous developers have brought the code.

Sure, Natron is buggy (tip: *Save* often). How could it not be? With only two developers supporting a massive set of features in the last couple of years, it is surprising it works as well as it does. The point is that the groundwork is done.

Natron's crux is the thing that makes it attractive; that is, the enormous amount of things you can do with it is also what puts off would be maintainers and new developers. It is nearly guaranteed that a lot of the code is going to be cryptic and under-documented. Getting up to speed just to correct the most obvious bugs is going to require climbing a steep learning curve.

Regardless, Natron needs developers. Better yet: It needs an institution willing to adopt the project and provide a stable home for it. Otherwise, Natron will end up disappearing. So, if you or anybody you know can help, get on it now, while the last modifications are still fresh and before Natron succumbs to bit rot.

Do it for free software; do it for the kitties.

Up Ahead

Natron is complex and powerful, and one article doesn't do it justice. In a future issue, I'll show you how Natron can make your video creations much more interesting, you'll see a real-life project using some of Natron's coolest features. ■■■

Info

[1] Natron: *https://natrongithub.github.io/*
[2] Blender: *https://www.blender.org/*
[3] Kdenlive: *https://kdenlive.org/*

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*.

SCaLE 17x

Date: March 7-10, 2019

Location: Pasadena, California

Website: *https://www.socallinuxexpo.org/scale/17x*

SCaLE – the 17th annual Southern California Linux Expo – is the largest community-run open-source and free software conference in North America. SCaLE 17x expects to host 150 exhibitors this year, along with nearly 130 sessions, tutorials and special events.

Cloud Expo Europe 2019

Date: March 12-13, 2019

Location: London, England

Website: *https://www.cloudexpoeurope.com/*

Cloud Expo Europe is the UK's leading event for connecting technologists, business leaders, and senior business managers with experts, solutions and services to help accelerate digital transformation plans. Meet with leading technology innovators and service providers and access a wealth of knowledge on emerging trends, tech deep dives, and market forecasts.

CloudFest 2019

Date: March 23-29, 2019

Location: Europa Park, Germany

Website: *https://www.cloudfest.com/*

CloudFest 2019 – everything you loved about WHD.global, only bigger, bolder, and reflecting the entire cloud ecosystem. Join more than 7,000 cloud-industry decision-makers to take over an amusement park for an immersive networking and deal-making experience. Use code CF19LUR to register free!

Events

Linux of Things (linux.conf.au)	January 21-25, 2019	Christchurch, New Zealand	https://linux.conf.au/
Enigma 2019	January 28-30, 2019	Burlingame, California	https://www.usenix.org/conference/enigma2019
Univention Summit	Jan 31-Feb 1, 2019	Bremen, Germany	https://www.univention-summit.de/
Fosdem '19	February 2-3, 2019	Brussels, Belgium	https://fosdem.org/2019/
FAST '19	February 25-28, 2019	Boston, Massachusetts	https://www.usenix.org/conference/fast19
SCaLE 17x	March 7-10, 2019	Pasadena, California	https://www.socallinuxexpo.org/scale/17x
SMART IOT 2019	March 12-13, 2019	London, England	https://www.smartiotlondon.com/
Open Source Leadership Summit	March 12-14, 2019	Half Moon Bay, California	https://events.linuxfoundation.org/events/open-source-leadership-summit-2019/
Icinga Camp Berlin	March 14, 2019	Berlin, Germany	https://www.icinga.com/events/icinga-camp-berlin/
Chemnitzer Linux-Tage	March 16-17, 2019	Chemnitz, Germany	https://chemnitzer.linux-tage.de/2019/en/
CloudFest 2019	March 23-29, 2019	Europa-Park, Germany	https://www.cloudfest.com/
SREcon 19 Americas	March 25-27, 2019	Brooklyn, New York	https://www.usenix.org/conference/srecon19americas
SUSEcon 2019	April 1-5, 2019	Nashville, Tennessee	https://www.susecon.com/
Cloud Foundry North America 2019	April 2-4, 2019	Philadelphia, Pennsylvania	https://www.cloudfoundry.org/event/nasummit2019/#
Open Networking Summit (ONS)	April 3-5, 2019	San Jose, California	https://events.linuxfoundation.org/events/open-networking-summit-north-america-2019/

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	34
Swapnil Bhartiya	8, 16, 24
Paul Brown	90
Zack Brown	12
Bruce Byfield	44, 60
Joe Casad	3
Mark Crutch	63
Marco Fioretti	84
Jon "maddog" Hall	65
Charly Kühnast	47
Rubén Llorente	70
Vincent Mealing	63
Pete Metcalfe	50
Anzela Minosi	20
Martin Mohr	56
Graham Morrison	78
Mike Schilli	40
Ferdinand Thommes	30, 66

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
© Konstantin Shaklein, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 3090 5128

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
2721 W 6th St, Ste D
Lawrence, KS 66049 USA

Publisher
Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

Printed in Nuremberg, Germany by hofmann infocom GmbH on recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 2721 W 6th St, Ste D, Lawrence, KS, 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 2721 W 6th St, Ste D, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany.

Approximate
UK / Europe Feb 02
USA / Canada Mar 01
Australia Apr 01
On Sale Date

Issue 220 / March 2019

VirtualBox Hacks

Virtualization is part of life now. Millions of Linux users fire up virtual machines to test new software, try new operating systems, or experiment with alternative configurations. Next month we show you some secrets for doing more with the popular open source virtualization tool VirtualBox.

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***

Lead Image © balein, 123RF.com

SUPERMICRO

Save Money.
Save Mother Earth.

RESOURCE-SAVING SERVERS

Build your data center with technology that is
BIG on **SAVINGS** and **LOW** on environmental **IMPACT**

60% Space Savings | 50% Energy Savings | e-Waste Savings

25 Years
OF GROWTH & INNOVATION

Learn more at
www.supermicro.com/WeKeepITGreen

intel XEON PLATINUM inside

SUPERMICRO