FREE DOUBLE-SIDED DVD

ATTENTION ALL HACKERS AND MAKERS!

NOW INCLUDING UBUNTU user

PRO LINUX MAGAZINE

LINUX PRO MAGAZINE

DECEMBER 2017

BITKEY

Keep your money safe with this special Linux for Bitcoin transactions

7 Hardware Analysis Tools

Pony

Frisky new language for coding concurrent operations

What's New in Knoppix 8.1

MakerSpace

- Rasp Pi wireless access point
- Build an HPC cluster with Pi Zeros

smenu

Easy tool for command-line menus

LINUXVOICE

- Learn security with Web Security Dojo
- SUSE travels to Prague for their yearly soirée
- Phipps: Getting serious about privacy
- maddog on spreading the word about FOSS

FOSSPicks

- Stellarium
- Sysdig

Tutorial

Devilspie2: Scripting window actions

Issue 205
Dec 2017
US$ 15.99
CAN$ 17.99

HETZNER ONLINE

RYZE ‘N’ SHINE

More parallelization. More power.

Dedicated Root Server AX60-SSD

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

monthly $ **65**

The ideal solution for highly-parallelizable processes.

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

www.hetzner.de/us

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

INSTALL THE PATCH

Dear Reader,

Once it was announced to the public, the news of the KRACK attack spread quickly over the Internet – a flaw in the handshake system for wireless devices that allows an attacker to compromise encryption. According to reports, the attack puts almost all devices that engage in WPA2-encrypted wireless networking at risk. Early warnings said Android was most vulnerable, but later updates reported that Windows, iOS, macOS, and OpenBSD were also at risk. Linux wasn't safe either, with several versions of the *wpa_supplicant* utility marked as vulnerable. Although the problem was publicly announced in October, vendors were warned privately in May, and many are already well along finding solutions.

The KRACK attack is particularly significant because *everybody* is using wireless networking. Many people younger than 21 just look bewildered if you show them a Cat 5 networking cable. Such things don't exist in their world, because wireless is everywhere – at the library, at the malt shop, in the home. One of the reasons for the recent explosion in wireless networking is that everybody trusts it now, and they trust it because they have this general sense that the glaring security issues that made everyone nervous about wireless in the first place have somehow been resolved. If you asked many security experts in the past couple years, they would say wireless networking is fine as long as you:

- use WPA2 wireless encryption
- use a strong password

But actually, it turns out those experts were overconfident.

By the time you read this, I hope word of the KRACK attack has already reached you and updates are available for all your wireless devices, including your phone, your tablet, your computer, and your wireless router.

If you haven't heard about KRACK, check with your OS or hardware vendor as soon as you can. The word is that this attack has not appeared in the wild so far, but now that the description is out there, someone is going to implement it, so you'd better hurry.

I hesitate to make this the main subject of the essay, because I often muse about this kind of thing, but I really do need to mention: Shouldn't we have expended a little more time and energy up front to explore this issue before the whole planet went wireless? I know, these kinds of problems are hard to spot, but seriously, this sudden discovery that a tool we depend on is not as secure as we thought it was yesterday seems to be happening a lot.

As for wireless networking, first we had WEP, which we said was secure at first, then we said, "uh … never mind, here's WPA, which we also said was secure, and then we said, "never mind, here's WPA2." Now it's …"uh, yeah, but we gotta fix WPA2."

"Well of course!" you impatiently inform me. "What's the problem? That's what always happens. Developers develop some software, then they tell you its safe and tested, then someone figures out it isn't safe, then the developers create a patch and tell you to install the patch and it will *really* be safe this time, then you go out and install the patch before some nefarious actor operating with anonymity uses the attack to steal your secrets, then the whole process starts over. What could be more obvious?"

And you're right, that is what always happens, but all I'm saying is, it's really weird if you look at it from a distance. If you believe in Everett's theory of multiple branching universes, you'd have to believe there's a version of us out there somewhere that has figured out a better way to develop software.

But until our universe catches up … INSTALL THE PATCH!

Joe Casad,
Editor in Chief

LINUX MAGAZINE

WHAT'S INSIDE

The mysterious world of cryptocurrency is quickly becoming part of everyday life. If you've been around computers and the Internet for the past five years, chances are you're either using Bitcoin or you're curious about it. This month we look at BitKey, a Debian-based Linux designed to serve in a hyper-secure Bitcoin cold storage environment.

Also inside:

- **Hardware Analysis Tools** – we study some leading Linux tools for diagnosing hardware problems. (page 24).
- **HDF5** – this hierarchical filesystem supports several programming languages and is ideal for Big Data and machine learning environments (page 36).

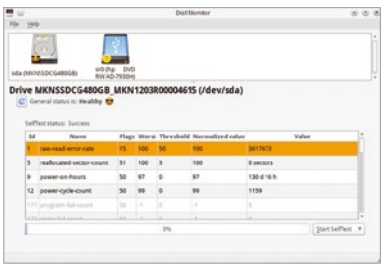
This issue also unveils our new MakerSpace section, with articles on a Raspberry Pi access point and a tool for building an HPC cluster with tiny Pi Zero computers.

SERVICE

NEWS

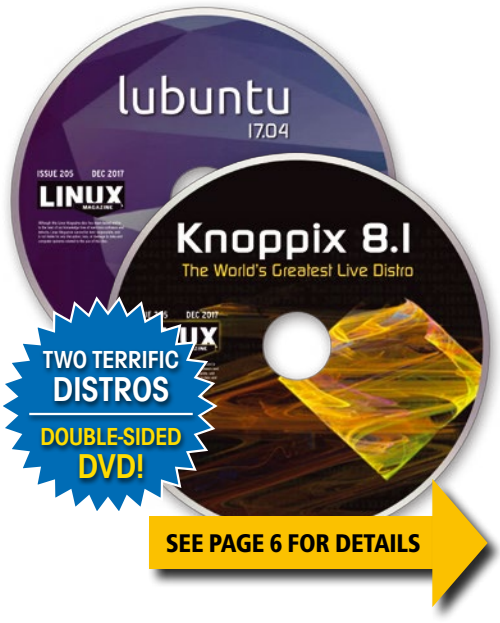
REVIEWS

COVER STORY



IN-DEPTH

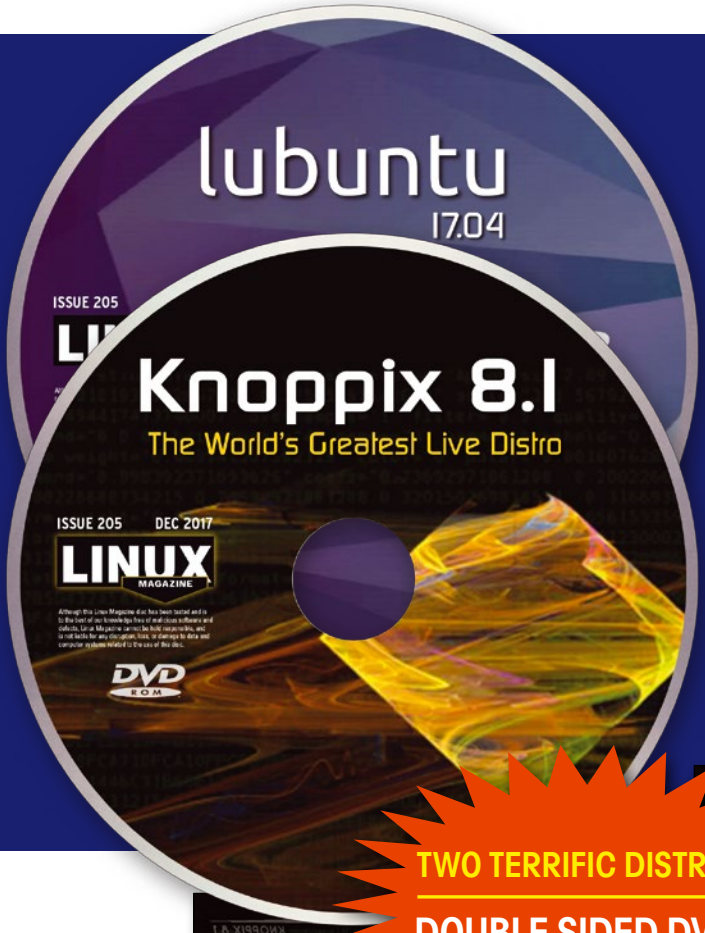
MAKERSPACE



SEE PAGE 6 FOR DETAILS

LINUXVOICE

On the DVD



Knoppix 8.1 (Multiarch Live)

Creator Klaus Knopper consistently delivers one of the fastest and most flexible Live Linux distributions. Knoppix starts and runs almost five times faster from a USB flash disk than from a CD or DVD, so you'll find the `flash-knoppix` program in the menu under *Knoppix / Install KNOPPIX to flash disk* very useful. Emphasizing "Everything 3D," Knoppix 8.1 includes OpenSCAD – a 3D construction tool for those with no drawing skills; Slic3r, to prepare G-code printing for 3D models; Blender, for extensive 3D modeling and animation programming; and FreeCAD, for drawing 3D models.

Lubuntu 17.04 (32-bit Live)

The Zesty Zaphus release of this Ubuntu-flavored distro features the Lightweight X11 Desktop Environment (LXDE). Lubuntu is just what you need for your older machines with fewer resources, although it runs just fine on modern hardware, as well. Lubuntu aims to be "lighter, less resource hungry, and more energy-efficient by using lightweight applications and LXDE." The latest release ships with Linux Kernel 4.10.

TWO TERRIFIC DISTROS

DOUBLE-SIDED DVD!

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

ADDITIONAL RESOURCES

[1] Knoppix 8.1: *http://www.knopper.net/knoppix/knoppix810-en.html*

[2] Lubuntu 17.04: *http://lubuntu.net/blog/lubuntu-1704-zesty-zapus-released*

REAL SOLUTIONS FOR REAL NETWORKS

FREE DVD in every issue!

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

Learn the latest techniques for better:

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

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

6 issues per year!

ORDER NOW: shop.linuxnewmedia.com

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

Kubernetes 1.8 Announced

The Kubernetes community has announced the release of Kubernetes 1.8, which comes with many new features. Kubernetes, an implementation of the Borg system that Google runs internally to power its own clusters, has become one of the hottest open source projects and the de facto container management and orchestration tool.

This release is as much about the code as it is about technology. According to Kubernetes, "In addition to functional improvements, we're increasing project-wide focus on maturing process, formalizing architecture, and strengthening Kubernetes' governance model."

One of the highlights of this release is graduating role-based access control (RBAC) to a stable stage. RBAC allows admins to restrict system access to authorized users, adding another layer of security. In Kubernetes, RBAC allows cluster administrators to define roles dynamically to enforce access policies through the Kubernetes API.

This release also comes with a beta support for filtering outbound traffic through network policies augmenting existing support for filtering inbound traffic to a pod. These two new features offer admins powerful tools for enforcing organizational and regulatory security requirements within Kubernetes.

Kubernetes 1.8 is available for download on GitHub (*https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG.md#v180*).

Final Ubuntu Desktop 17.10 Beta Arrives

Canonical has announced the release of the final beta of Ubuntu 17.10, code named Artful Aardvark. With this release, Ubuntu code names have gone back to the beginning of the English alphabet, which is apt, because with this release, Ubuntu is kind of starting fresh. Canonical dropped its desktop ambitions earlier this year, signaling the shutdown of efforts like Unity. This is the first release of Ubuntu that comes with Gnome as the official and default desktop environment and shell.

However, Canonical has ensured that people upgrading from the previous release of Ubuntu running Unity 7 will not be in for a shock. Ubuntu developers have worked on adding some custom features and functionalities so established workflows are not overly affected.

Will Cooke, Director of Ubuntu Desktop at Canonical said, "… we've spent time making sure that the people who having been using Unity 7 for years don't have to change their workflow too much. The most obvious example of this is the Ubuntu Dock (based on Dash To Dock and developed upstream)."

Ubuntu is also adopting Wayland as the default display server for the desktop, depending on the hardware. However, users can switch between Wayland and Xorg. Beyond these cosmetic changes to help existing Ubuntu users, Canonical is sticking to default Gnome settings and features. Some of the new features include the newly designed Gnome Settings. Ubuntu 17.10 brings support for all driverless printers, which means no need to install drivers.

Canonical has also discontinued its own Ubuntu Store, and it now defaults to Gnome software, which also allows it to update the system itself.

With this release, you can also move away from distro-specific RPM and DEB packages and use bundled Snap packages. Unfortunately, the rest of the desktop Linux world is rallying behind Flatpak, so it will be interesting to see if Canonical drops Snaps on the desktop and adopts Flatpak.

You can download the beta from the official Ubuntu page (*http://releases.ubuntu.com/17.10/*).

Linus Torvalds Invites Attackers To Join the Kernel Community

Torvalds_Linux_2014_CC BY-SA 4.0

Last week at the Open Source Summit, Linus Torvalds sent an open invitation to security hackers and attackers to join the Linux kernel community.

Torvalds is not a huge fan of the "security community" because he doesn't see the issues in black and white. He maintains that bugs are part of the software development process and they cannot be avoided, no matter how hard you try: "constant absolute security does not exist, even if we do a perfect job," said Torvalds in a conversation with Jim Zemlin, the executive director of the Linux Foundation.

In a previous conversation with us, Torvalds said there are way too many people out there who continue to search for bugs and holes in software to attack.

Torvalds is fascinated by how smart some of these security hackers are. They always find something, in a very clever way, somewhere that no one thought could have been a security hole.

"As a technical person, I'm always very impressed by some of the people who are attacking our code," Torvalds said. "I get the feeling that these smart people are doing really bad things … I wish they were on our side because they are so smart, and they could help us."

Torvalds said that the kernel community would be in much better shape if they could get as many of those smart people before turning to the dark side. He wanted them to help improve the code instead of attacking it.

"I'm encouraging the people who are interested in security to come to us instead of attacking us," said Torvalds.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

Resource monitoring for remote applications • Jeff Layton
Remora combines profiling and system monitoring to help you get to the root of application problems by revealing its use of resources.

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

Scalable network infrastructure in Layer 3 with BGP • Martin Loschwitz
Large environments such as clouds pose demands on the network, some of which cannot be met with Layer 2 solutions. The Border Gateway Protocol jumps into the breach in Layer 3 and ensures seamlessly scalable networks.

End-to-end monitoring – Measuring what the user perceives • Jürgen Vigna
Continuously monitoring the performance of applications helps ensure the service quality in multilayered cloud services.

Software-defined wide area networks • Julian Frede
A natural consequence of software-defined storage and software-defined data centers is the software-defined wide area network, or the Internet connections between locations and cloud services.

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

Packaging apps to run on any Linux device • Chris Binnie
Canonical's Snapcraft (Snappy) package manager creates a self-contained application that works across Linux distributions. We show you how to install, publish, and run a simple snap.

ZAP provides automated security tests in continuous integration pipelines • Chris Binnie
Canonical's Snapcraft (Snappy) package manager creates a self-contained application that works across Linux distributions. We show you how to install, publish, and run a simple snap.

Oracle Donates Java EE to the Eclipse Foundation

Oracle is donating yet another open source technology that it acquired from Sun Microsystems. After discussions with IBM, Red Hat, and a few open source foundations, Oracle has chosen the Eclipse Foundation as the rightful home for the Java Enterprise Edition (Java EE) platform.

"The Eclipse Foundation has strong experience and involvement with Java EE and related technologies. This will help us transition Java EE rapidly, create community-friendly processes for evolving the platform, and leverage complementary projects such as MicroProfile. We look forward to this collaboration," said David Delabassee, Software Evangelist at Oracle.

To ensure smooth transition to the new home, Oracle has made certain changes to its proposal.

The company will relicense Java EE technologies and related GlassFish technologies to the foundation. This would include Reference Implementations (RIs), Technical Compatibility Kits (TCKs), and associated project documentation.

Oracle is also recommending a new name and new branding for the platform within the foundation. However, for continuity, the company intends to enable the use of existing *javax* package names and component specification names for existing Java Specification Requests (JSRs)

Microsoft Is Building a Programming Language for Quantum Computers

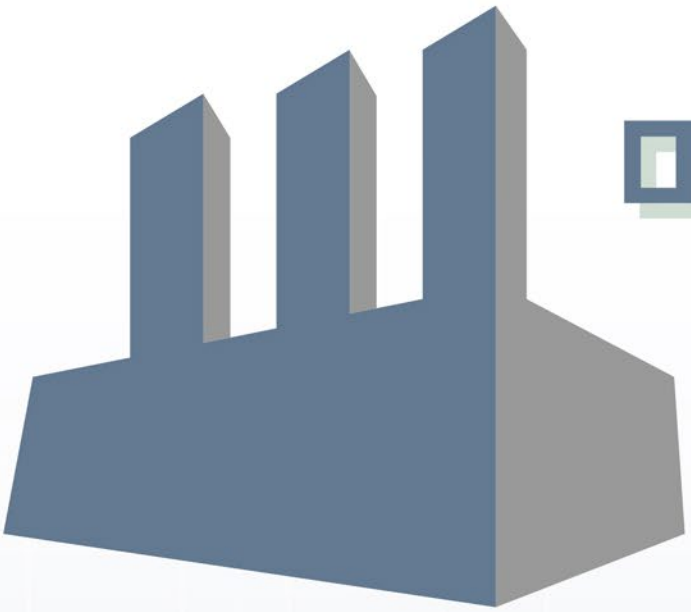
At the Ignite Conference, Microsoft announced that later this year it will release a new quantum computing programming language that is very tightly integrated with Visual Studio. The language is designed to work on both a quantum simulator and a quantum computer.

Microsoft has been working on quantum computers for decades. The company hired Michael Freedman some 20 years ago to continue his work on topology. Microsoft's quantum computing work is based on the work Freedman has done over time. Eventually Microsoft has started to see some results of the work it has been doing for ages.

Krysta Svore, principal researcher at Microsoft Research, said that a programming language that can run in a simulated environment will help people understand how to harness quantum power for different types of problems.

One big difference between the work Microsoft is doing on quantum computing and the rest of the industry, is that the company doesn't want to build a quantum computer for display in labs. Microsoft wants to deliver a full-fledged topological quantum computing system.

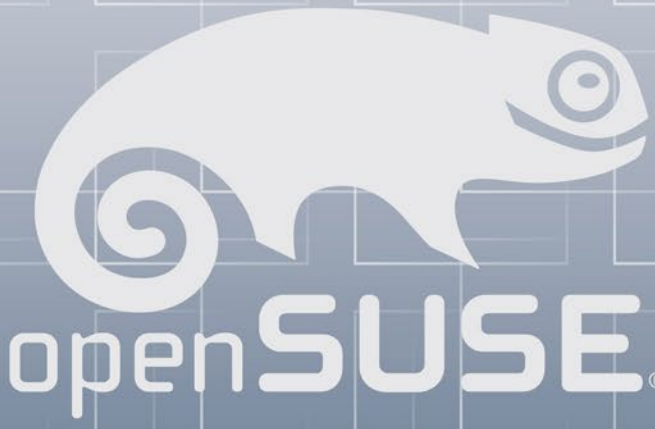
According to Allison Linn, Senior writer, editor, and multimedia storyteller at Microsoft, it's a system that includes everything from hardware capable of consistently running calculations that require tens of thousands of logical qubits to a complete software stack that can program and control the quantum computer.



open build service

A generic system to build and distribute packages from sources in an automatic, consistent and reproducible way

openbuildservice.org



Zack's Kernel News

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

By Zack Brown

ZACK BROWN

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

Replacing the Random Number Generator

Stephan Müller ran into difficulties when he tried to do a wholesale replacement of the Linux random number generator (LRNG). A good source of random numbers is crucial for securing running systems against certain kinds of attacks. Stephan felt that the existing RNG code suffered from design flaws that required a full rewrite.

In particular, he said that the old `/dev/random` implementation had once been sufficient, but now was having trouble providing good randomness for embedded systems and other newer hardware on the market. Stephan felt that LRNG could work as a simple drop-in replacement for `/dev/random` so that user code would never notice the change.

However, regardless of the value of Stephan's implementation, Greg Kroah-Hartman said that making such a big change all at once, to such a crucial piece of the kernel, was not a good idea. He suggested submitting a series of smaller patches that would gradually implement what Stephan had in mind.

But Stephan said he'd tried that and rarely got any response to his smaller patches. Not even a thumbs-down. Also, he said, any patches that were more than just cleanups, but were actual logic changes, were going to be relatively big. There was no getting around that.

But Greg insisted that "Evolution is the correct way to do this; kernel development relies on that. We don't do the 'use this totally different and untested file instead!' method."

Stephan pointed out that "The offered patch set does not rip out existing code. It adds a replacement implementation which can be enabled during compile time. Yet it is even disabled per default." He said that many other areas of the kernel used such an approach, and it seemed to be a standard accepted practice.

Meanwhile, Theodore Ts'o said that, in fact, he had been taking a variety of Stephan's patches for quite some time and incorporating them into the existing `/dev/random` code, but he hadn't taken some of the more extreme logic changes because he said H. Peter Anvin had disagreed with Stephan about their value.

Sandy Harris replied here, pointing out some of the issues with the existing `/dev/random` design. He said, "you cannot generate good output without a good seed and just after boot, especially first boot on a new system, you may not have enough entropy. A user space process cannot do it soon enough, and all the in-kernel solutions (unless you have a hardware RNG) pose difficulties."

He added that Stephan's approach had a lot of good theory behind it, in terms of being able to get a proper seed as early as possible in the boot process. Nevertheless, he agreed that there wasn't enough justification to merge the whole patch as-is.

But Theodore replied that he was skeptical of the theoretical underpinnings of Stephan's approach. He said:

Most of them are done using the x86 as the CPU. This is true of the McGuire, Okech, and Schiesser paper you've cited above. But things are largely irrelevant on the x86, because we have RDRAND. And while I like to mix in environmental noise before generating personal long-term public keys. I'm actually mostly OK with relying on RDRAND for initializing the seeds for hash table to protect against network denial of service attacks. (Which is currently the first user of the not-yet-initialized CRNG on my laptop during kernel boot.)

The real problem is with the non-x86 systems that don't have a hardware RNG, and there depending timing events which don't depend on external devices is much more dodgy. Remember that on most embedded devices there is only a single oscillator driving the entire system. It's not like you even have multiple crystal oscillators beating against one another.

So if you are only depending on CPU timing loops, you basically have a very complex state machine, driven by a single

oscillator, and you're trying to kid yourself that you're getting entropy out the other end. How is that any different from using AES in counter mode and claiming because you don't know the seed, that it's "true randomness"? It certainly passes all of the statistical tests!

Hence, we have to rely on external events outside of the CPU, and so we need to depend on interrupt timing – and that's what we do in `drivers/char/random.c` *already! You can debate whether we are being too conservative when we judge that we've collective enough unpredictability to count it as a "bit" of randomness. So it's trivially easy to turn the knob and make sure the CRNG gets initialized more quickly using fewer interrupt timings, and boom! Problem solved.*

*Simply turning the knob to make our entropy estimator more lax makes people uncomfortable, and since they don't have access to the internal microarchitecture of the CPU, they take comfort in the fact that it's really, really complicated, and so something like the Jitter RNG *must* be a more secure way to do things. But that's really an illusion.*

If the real unpredictability is really coming from the interrupts changing the state of the CPU microarchitecture, the real question is how many interrupts do you need before you consider things "unpredictable" to an adequate level of security? Arguing that we should turn down the "interrupts per bit of entropy" in `drivers/char/random.c` *is a much more honest way of having that discussion.*

In a later email, Theodore also added, "Practically no one uses `/dev/random`. It's essentially a deprecated interface; the primary interfaces that have been recommended for well over a decade is `/dev/urandom` and, now, `getrandom(2)`. We only need 384 bits of randomness every 5 minutes to reseed the CRNG, and that's plenty, even given the very conservative entropy estimation currently being used. This was deliberate. I care a lot more that we get the initial boot-time CRNG initialization right on ARM32 and MIPS embedded devices, far, far, more than I care about making plenty of information-theoretic entropy available at `/dev/random` on an x86 system. Further, I haven't seen an argument for the use case where this would be valuable. If you don't think they count because ARM32 and MIPS don't have a high-res timer, then you have very different priorities than I do."

But Stephan pointed out that his concerns were not limited to `/dev/random` alone, but included `/dev/urandom` as well. Regarding Theodore's insistence that 384 bits of randomness every five minutes was enough, Stephan pointed out that on certain embedded systems, this would not actually be enough.

Jeffrey Walton also objected to Theodore's statement that `/dev/random` was a deprecated interface; he said if this were true, it should be better documented. He quoted the `random(4)` man page as saying, "`/dev/random` should be suitable for uses that need very high quality randomness such as one-time pad or key generation." He added, "If the generator is truly deprecated, then it may be prudent to remove it completely or remove it from userland. Otherwise, improve its robustness. At minimum, update the documentation."

The discussion petered out shortly afterward. The debate seems to focus on whether or not Stephan has identified a truly better source of random number seeds for all hardware setups, or not. At stake is the security of the system, particularly early in the boot process, when good sources of random number seed data can be scarce. For the moment, it doesn't seem as though Stephan has convinced Theodore or the other big-time hackers in that area, but at least one of Stephan's concerns has been answered – his smaller fixes are being considered and, in some cases, adopted into the kernel.

Checking When a Process Dumps Core

Roman Gushchin from Facebook wanted to be able to check to see if a process was in the midst of producing a core dump. The idea was, you don't want to kill a process while it's dumping core, because then you get an undebuggable dump. He posted a small patch to add a core-dump flag to `/proc/$PID/status`.

There was no significant dissent, but there were questions about the proper interface to use. Alexey Dobriyan felt that instead of adding the flag, it might be better to introduce a new "core-dumping" state that any process ID might be in. Roman didn't like this idea

because, even while dumping core, the process could be in other states, such as sleeping or running.

In that case, Alexey said, it might be faster to have a `/proc/$PID/coredump` file that was simply either `1` or `0`, instead of having to open and parse the status file each time. Although Roman felt that speed was not really an issue, there wasn't really a need for anyone to loop on checking for a core dump. They'd just check once and then kill the process.

Konstantin Khlebnikov also had a couple of suggestions. For one thing, he thought it might reduce clutter to have the line in the `/proc/$PID/status` file only if a core dump was actually taking place. But Roman didn't like this idea because no other data element appears and disappears like that.

Konstantin also suggested exposing the process ID of the core-dump helper, but Roman felt this would risk introducing race conditions, and he couldn't think of any valid use for it. At the most, Roman felt this would be a separate feature, not part of the current patch.

With no further suggestions and no objections, Roman asked Andrew Morton to accept the patch into his tree to feed up to Linus Torvalds at some point, but before doing so, Andrew wanted to know what real-world situations would actually use this feature.

Roman replied that, at Facebook, they were seeing "corrupted coredump files on machines in our fleet, just because processes are being killed by timeout in the middle of the core writing process. We do have a process health check, and some agent is responsible for restarting processes which are not responding for health check requests. Writing a large coredump to the disk can easily exceed the reasonable timeout (especially on an overloaded machine). This flag will allow the agent to distinguish processes which are being coredumped, extend the timeout for them, and let them produce a full coredump file."

Andrew also wanted the feature explained in documentation, and Roman updated the patch to do so.

In general, there's no guarantee that a patch like this would go into the kernel. The fact that this particular one had an easy path means nothing. A security hole can always emerge from an unexpected angle, and that would be that for the patch. Or, Andrew or Linus could decide that the use case envisioned by Roman was too specific to the company producing the patch and didn't justify the added cruft to the `proc` file. It could turn out that the "proper" place to perform a given check is in user space. Any number of things can put the kibosh on a patch of this sort. In this case, however, it looks like smooth sailing.

Fixing Filesystem Security Issues

Security fixes sometimes resemble a game of whack-a-mole. As long as new features keep being added to the kernel, it seems there will always be security holes to exploit, and new features must continue to be added, as long as there is new hardware to support.

Salvatore Mesoraca recently closed a security hole in which an attacker would create a file that would normally be created by a piece of user software, and then the user software would write its sensitive data to the attacker's file instead of one owned and readable only by itself.

The solution was simply to disallow this at the filesystem level and cause the user software to fail to access the file if it was unable to create the file itself. This way, in the worst case, the user would be unable to run their software; however, this is better than running the software and falsely believing it to be secure.

There was a general cheer among those watching Salvatore's work. In particular, Alexander Peslyak pointed out that there were a number of related security problems that could be better identified with this fix in the code. He had used a similar feature to help debug security issues in an old Postfix privsep implementation, and he expected others would find similar benefits.

It's interesting to note that in Linux – and in free software generally – security fixes take precedence over all other considerations, even to the point of extensive design changes and disabling significant features, and that's exactly the right ethic to have about it. Contrast this with proprietary operating systems that tend to spawn entire economies based on using simple scripts to crack millions of vulnerable systems, using them to deliver spam, launch DoS attacks, and so on.

Adding Build Dependencies to Clean the Source Tree

Linus Torvalds recently took the lead on adjusting the kernel build dependencies after Masahiro Yamada asked about the situation of compiled – or in some other way processed – files in the kernel source tree. For example, some files in the Git repository had been processed by Lex and Yacc before going into the tree. On one level, this was odd, because traditionally only true source files are included in source repositories. On another level, though, it was useful. If all those source files had to be compiled by the user, there would always be the risk that the user's system would contain the wrong versions of the various Lex, Yacc, GCC, and other utilities needed to compile those files properly.

In his post, Masahiro noted that the kbuild system had already added some code to recompile those processed blobs from source, so, he reasoned, why not make that the default?

In an unusual move, Linus decided to see about implementing this himself. Possibly he just felt like having a little fun with something that wasn't really very dangerous to play with. Whatever the reason, he started tinkering with the kbuild system to see how robust he could make it and still compile all the Lex, Yacc, and other source files from scratch.

For most tools, he found that it was easy to require and detect a minimal version number to be officially supported by a given kernel release, but in the case of gperf, he found that the tool didn't generate any version markers that could be used to detect which version was actually in use on a given system.

This was relevant because apparently gperf behavior had changed in version 3.1, causing it to generate kernel files improperly. Without a good way to test for that version of the tool, kbuild couldn't satisfy itself that it would produce a proper build on a given run.

The usual way to solve such issues would be to use `autoconf`, which Linus hated with a fiery passion. Instead, he felt that any tool lacking proper version markers should just be removed from the build requirements. In this case, he would remove any gperf dependencies from the Linux kernel.

So, Linux now depends on more tools than it did before, and yet it's highly unlikely that anyone will even notice the change, because all the tools it relies on are standard on every Linux distribution. Still, it's unusual for Linus to be so quick to allow new dependencies like that. Maybe he figured that the dependencies were there in any case; they were just hidden by the fact that the compiled versions of each file were included in the source tree. ■■■

Protecting your bitcoin with BitKey

Key to the Bitcoin Safe

Bitcoin is the king of cryptocurrencies, which makes it attractive prey for Internet predators. BitKey is a Live Linux distribution designed to ensure that your Bitcoin wallet stays protected. *By Rubén Llorente*

Bitcoin [1] is an innovative form of currency that exists only in cyberspace. The Bitcoin monetary system is designed to allow people to pay each other over the Internet, without the need for intermediaries such as banks. An easy way to understand how it works is to imagine that the Bitcoin network is a big, public accounting book that is maintained and closely watched by all of its members. The book contains information about who has how many bitcoin and who transfers bitcoin to whom. Each bitcoin owner has a set of public and private keys that are used to write operations to the book. Thus, somebody who owns bitcoin can send them to another person just by writing that he is doing so in the accounting book and signing the operation with a private key.

This model is an extreme simplification, but it helps to understand that each bitcoin is not a piece of code, or a file, or something with its own entity. In fact, bitcoin only exist as a reference in a public registry that states that a certain user has a certain amount of them. In reality, that user does not have the bitcoin stored anywhere. All the user has is a set of public and private keys that allow changes to the registry in order to spend or receive money.

The Security Problem

Most desktop Bitcoin wallets are programs that manage the user's keys, allowing the user to manage digital money in a user-friendly way. However, most users keep their wallet software running in computers that are connected to the Internet and are not necessarily very secure. If an online computer is infected by a piece of malware that allows an attacker to steal the private keys stored in that computer, the attacker would then gain access to the money.

Most popular Bitcoin wallets offer basic security features, such as password encryption. However, since many wallets are used to store very large amounts of money, those features are often not enough to stop a determined attacker. Security-minded users often keep their private key material in machines that are never connected to the Internet, or print the keys and store them as paper copies in order to protect them from black-hat crackers. The operational security procedures required to receive and spend bitcoin under these circumstances are extremely cumbersome.

Enter BitKey

BitKey [2] is an attempt to make it easier to follow proper operational security procedures when using bitcoin. BitKey is a Live Linux distribution that includes all the basic tools a Bitcoin power-user might need for such a purpose, and nothing more. At least, that is the theory.

As of July of 2017, the latest BitKey version is 14.2.0, and it is offered as a hybrid ISO download. After you burn the ISO to a CD or install it on a USB flash drive, you can boot BitKey as a live system. BitKey loads itself fully into the computer's RAM

AUTHOR

Rubén Llorente is a mechanical engineer whose job is to ensure that the security measures for the IT infrastructure of a small clinic are law compliant and safe. He is also an OpenBSD enthusiast and a weapon collector.

memory, allowing you to remove the boot media afterwards.

The boot menu prompts you for a boot mode. BitKey supports three modes of operation. The simplest, and also the least secure, is hot-online mode. In hot-online mode, which is not recommended for real-world scenarios, BitKey behaves much as any regular live operating system with a set of Bitcoin tools installed.

The remaining modes are the ones used for proper operational security procedures (Table 1). Cold-offline mode is used for creating and managing private keys in secure, air-gapped computers. Cold-online mode is used for performing operations with public keys using Internet-enabled computers. The recommended scenario for using BitKey is to have two computers; one offline for signing operations, running in cold-offline mode; and one for interfacing with the Bitcoin network, running in cold-online mode without private keys. I will call the first machine *Safe* and the second machine *Unsafe*.

The user can check the current account balance from the *Unsafe* machine. The *Unsafe* machine is also used for generating unsigned transactions. An unsigned transaction is just an instruction for the Bitcoin network to transfer funds belonging to you to somebody else, but such an instruction has not been signed with the masker keys yet or sent to the network. An unsigned transaction must be copied over to the *Safe* system and signed for it to be valid.

The *Safe* system is used to sign the unsigned transactions generated from the *Unsafe* machine. The signed transactions are copied over to the *Unsafe* system and then sent to the network,

TABLE 1: BitKey Boot Modes

BitKey Boot Mode	Uses	Security Level
Hot-online	Create wallets, watch wallets, perform transactions	Low
Cold-offline	Create wallets, sign transactions	High
Cold-online	Watch wallets, create unsigned transactions	High

making them effective. Since the *Safe* machine is air-gapped, compromising it is not trivial, and the effects of an actual infection are less dangerous. It would be harder for a malicious program to deliver information to an attacker from a computer that has no working Internet connection.

Setting Everything Up

If what you have read so far sounds too complex, don't worry: It is not just you. It sounds complex because it is. The question is: How does all of this work in practice?

BitKey includes **Electrum** [3] as its main Bitcoin client. Although the reference Bitcoin client requires the user to download the whole blockchain, which is actually all the data existing in the accounting book and takes up lots of gigabytes, Electrum delegates to third-party servers in order to avoid downloading so much information. The Electrum client only downloads the parts of the blockchain that are needed, which allows the user to get started more quickly than the reference Bitcoin program. Electrum has many other interesting features, such as its integrated cold storage support [4].

The first step to get started with BitKey and Electrum is to boot your *Safe* machine in cold-offline mode (Figure 1) and create a wallet, which will be stored in a safe flash drive. I will call this flash drive *Blue* from now on. The *Blue* storage will contain the private keys. Therefore, it must never be attached to a system connected to the Internet – or to any untrusted system. Putting the flash drive on a network would expose the wallet to the threat of malware. When not in use, the best place for it is in a safe.

In order to create the wallet, attach your empty *Blue* storage and run Electrum. Electrum will ask you for an encryption passphrase that will be used to protect the data contained in *Blue*. You will also be asked whether you want to import an existing wallet or create a new one. Hit *Create a new wallet* and move forward. You will be provided with a key generation seed, which is no more than a very long passphrase that can be used to recreate your whole wallet from scratch if you ever lose it. Either learn it by heart, or write it down. When you click on *Next*, Electrum will ask you for that very same passphrase (Figure 2). You will also be prompted for an encryption passphrase for your wallet, which is a bit strange because you just provided a passphrase for *Blue* a few clicks ago. Still, it can only help if your valuable money is protected with double encryption.

The second step is to get your *Unsafe* computer working. Your *Safe* machine is now up and ready, but since it is not intended to connect to the dangerous den of evil that we call the Internet, you cannot do much with it yet. To perform Internet tasks such as monitoring your account balance, you need to create an online wallet that will contain only your public key material. Such a wallet is called a *watch-only wallet*.

Attach another flash drive to your *Safe* machine. This drive will eventually contain your watch-only wallet, and I will call it *Black* from now on. In Electrum's toolbar, hit *Wallet* | *Master Public Keys*. You will see a string that you can save in plain text in the *Black* storage.

Extract the *Black* storage from the *Safe* computer and boot the *Unsafe* one in cold-online mode. Run Electrum just as you did before. This time, hit *Import wallet or import keys*, and

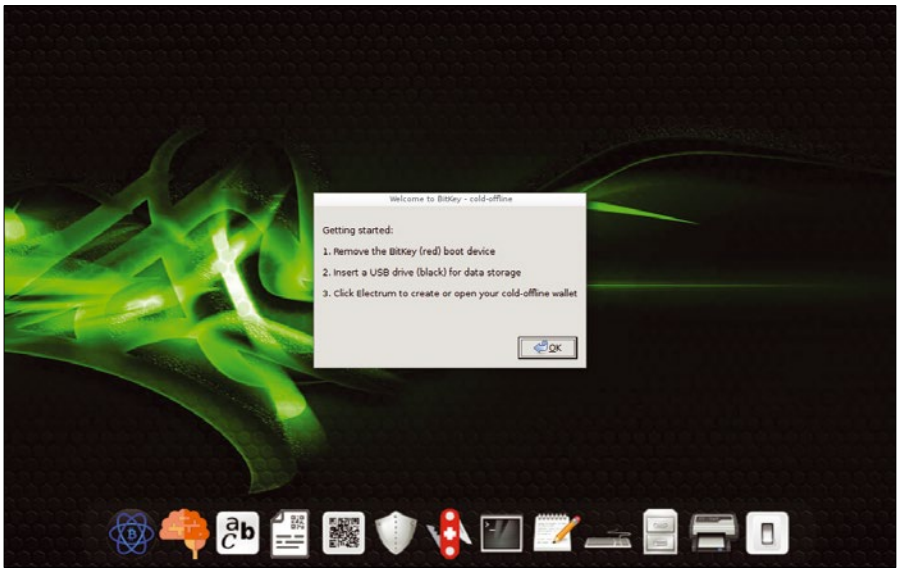


Figure 1: After you boot into cold-offline mode, you are greeted with a set of instructions that is not especially helpful without some prior knowledge of the BitKey environment.

then paste your master public key (which you saved in plain text in *Black*) in the *Import* box. Hit *Next* and then let Electrum auto-connect to the network. Congratulations, your BitKey configuration is now complete.

Transferring Funds to Other People

The wallet stored in *Blue* is the only one that contains the private key material, and thus it is the only one that can sign transactions. In other words: You need to use the *Blue* wallet for making payments. However, since you cannot attach *Blue* to Internet-enabled systems, you cannot use the wallet directly. The procedure for making a transaction is cumbersome, inconvenient, and very safe.

In order to make a transaction, you must generate an unsigned transaction in your *Unsafe* machine using the wallet in *Black* (Figure 3), open the *Send* tab, and fill in the payment form. Hit *Send* and then save the unsigned transaction to a file. You may transfer this file to the *Safe* machine that has access to the *Blue* device using any means you find convenient. A third disposable flash drive is often the most convenient option. Run Electrum in *Safe* and go to *Tools* | *Load Transaction* | *From File*.

Figure 2: Electrum will ask you to reintroduce the generation seed, so make sure that you either learned it or wrote it down.

Hit *Sign* and save your signed transaction to a file. Transfer this file to the *Unsafe* machine. Finally, on the *Unsafe* box, use Electrum, and select *Tools* | *Load Transaction* | *From File* in order to load your signed transaction. Hit *Broadcast*. Congratulations, your payment has just been sent to the Bitcoin network!

Other Tools

BitKey also includes two utilities for creating paper wallets [5]:

- BitAddress [6]
- Bitcoin Paper Wallet [7]

Paper wallets are pieces of paper that contain a primary private and public key. The idea is that you can print and store a paper wallet in a safe location far from malware. Papers wallets are an alternative to the cold-online and cold-offline computer combination. The use of paper wallets is usually discouraged because it is extremely easy to lose the access to your coins if you make any mistake. Please, don't use paper wallets unless you understand their ins and outs. BitAddress offers instructions for the proper use of paper wallets if you are really interested in using them. Please, read the instructions carefully, and see the box entitled "Mistakes that Destroy your Money."

To get started with BitAddress, launch it from the BitKey desktop. Move your pointer randomly over the screen in order to obtain enough entropy for generating your Bitcoin address (Figure 4). A public-private key pair will be created and displayed on the screen, along with QR codes, in a printer friendly format. You may wish to print this paper, note your Bitcoin address down, and lock the paper wallet in a safe location. Treat this wallet as a piggy-bank. You may tell people to send money to you by giving your Bitcoin address to them. When you need to access your funds, import the private key of the paper wallet into any conventional Bitcoin client, spend all the funds, and

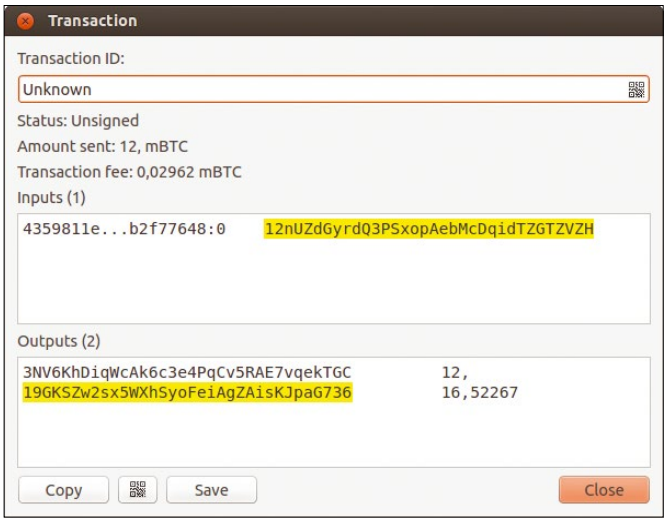


Figure 3: You need to create an unsigned transaction from your watch-only wallet and sign it using your private keys in order to make a payment.

MISTAKES THAT DESTROY YOUR MONEY

When you first create a Bitcoin wallet in any format and with any software, it creates a public-private key pair. This key pair is associated with a Bitcoin address that you use to receive funds.

The process followed when sending a payment, however, involves the creation of multiple sub-keys that have the main key pair as a master. Bitcoin clients track your account balance by tracking the input and output associated with your address.

For example, if you create an address and then Jack and Adam send you 1 bitcoin and 2 bitcoin each, your client will know that you have an input worth 1 bitcoin and another input worth 2 bitcoin. If you try to send 1.5 bitcoin to Adam one day later, your client will take the 2 bitcoin input, split it in two parts, send 1.5 bitcoin to Adam, and send the change (0.5 bitcoin) back to you. Your wallet software will send the change to a *phantom* address, that is, it will create a key pair attached to a new dynamically generated address and send the change to it.

The rationale for creating this change address is that it makes it harder for a hostile entity to track your transfers than just delivering the change to your main Bitcoin address. Each time you deliver a payment, a phantom address and a corresponding key pair are generated in the background. Since Bitcoin clients do this task automatically, the user is protected from the complexities of this method.

Problems occur when a user loses the key pairs that allow access to the money delivered to the user's own change addresses. If you create a paper wallet and money is delivered to it, all the inputs will be associated with your main address. If you later load this paper wallet in a software Bitcoin client and then make some payments, this program will generate multiple change addresses and key pairs that won't exist in the paper wallet at all. These change addresses will receive part of your funds. If you destroy the software wallet, you may lose all the funds associated with the change addresses.

Many people used to create a paper wallet in order to keep their main keys offline. They would then load the main key pair into a software wallet, make payments, and delete the software wallet in order to prevent it from being stolen by malware. The problem with this approach was that these users lost their access to the money associated with change addresses when they deleted their keys!

destroy the paper. In theory, once you have imported the paper wallet into a regular Bitcoin client, you could treat it as you would treat any regular software wallet. However, once the private keys are imported in an Internet-enabled computer, they are exposed to potential compromise, and the common recommendation is to use all the funds up at once and discard the wallet forever. Reusing a spent paper wallet is dangerous, and money loss is more likely than not if you try. Look online for more on the dangers of address reuse [8].

Also included with BitKey is WarpWallet, a utility for creating *brainwallets* [9]. A brainwallet is an easy-to-remember passphrase that can be fed to a brainwallet program in order to create a public and private Bitcoin key pair. The algorithm is deterministic – the same passphrase always generates the same key pair. The theory is that you can keep the passphrase in your head and avoid placing your Bitcoin keys on a computer until you really need to. When you need to access your money, you use the passphrase and WarpWallet to generate the keys and then import the private key into a regular Bitcoin program. Once imported, these wallets are similar to paper wallets and the same principles apply. As with paper wallets, brainwallets are dangerous, so use them with care.

Launch WarpWallet from the bar. Feed the program with a very secure passphrase, and provide it with your email address in order to generate the salt (Figure 5). The salt is extremely important, because unsalted brainwallets are considered extremely insecure (see the box entitled "Unsalted Brainwallets"). WarpWallet will generate and display a public-private key pair. Note the public address down and close the program. As with paper wallets, a brainwallet is a piggy bank. You can receive payments to your public address. When you are ready to spend the money, fire up WarpWallet, feed it with your passphrase and email address, and note down the private key that is generated. Import that private key into your Bitcoin client as before. Spend all the money at once! Brainwallets suffer drawbacks similar to paper wallets, so don't reuse your brainwallet address. (I know I have already mentioned the dangers of reusing addresses, but trust me, it is important.)

For maximum security, it is better to perform the wallet generation in cold-offline mode for both brainwallets and paper wallets and to load the private keys in hot-online mode just when you are going to spend them.

Figure 4: BitAddress requires the user to generate entropy by moving the pointer randomly around the screen.

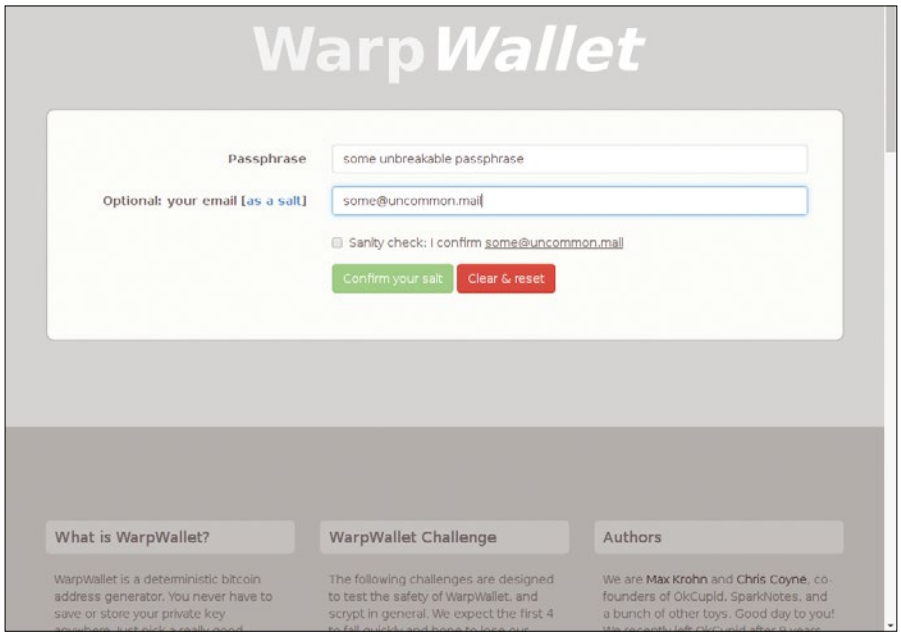


Figure 5: Brainwallets are like paper wallets, but instead of storing the keys on a piece of paper, the user remembers a passphrase and generates the keys by feeding the phrase to WarpWallet.

Finally, BitKey includes a password strength analyzer called zxcvbn (Figure 6). This tool uses advanced analysis to determine if a password or passphrase is safe.

What BitKey Lacks

BitKey is a useful solution for people who wish to be very proactive about protecting their Bitcoin wallets; however, it is not without shortcomings.

The main problem is that you won't find much official documentation about how to implement BitKey and integrate it into your secure procedures. The most useful instructions are in an article written by Liraz Siri on the TurnKey site [10]. (BitKey was created by core developers with the TurnKey Linux project.)

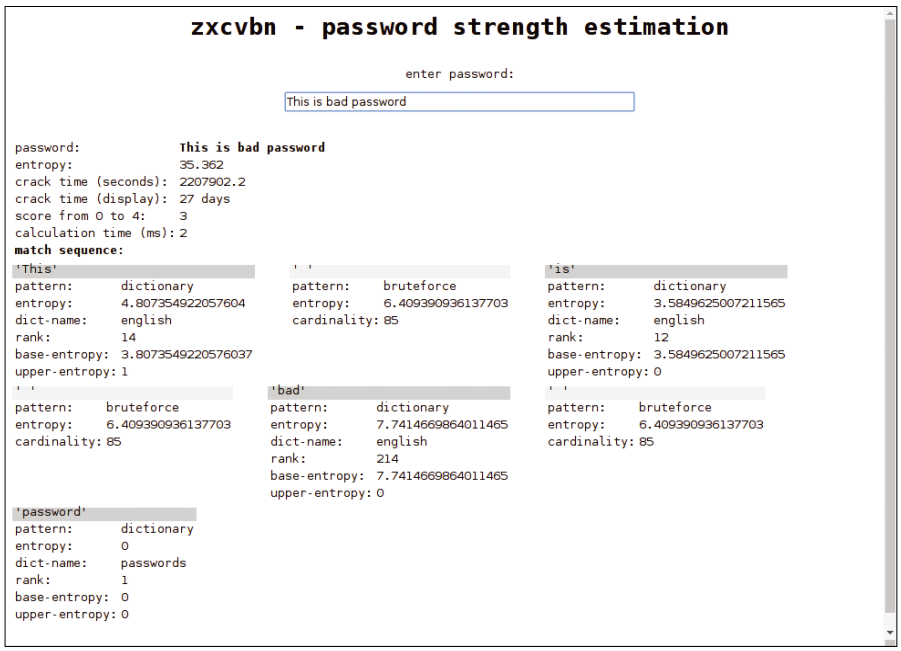


Figure 6: A tool for measuring password strength is included with BitKey.

UNSALTED BRAINWALLETS

Brainwallets that are generated without salt are vulnerable to cracking using rainbow tables and other advanced cracking methods. Bots are known to exist that patrol the blockchain, searching for vulnerable brainwallets, cracking them, and stealing all their funds. If you really want to use a brainwallet, make sure the brainwallet program you use is salting the hashes. Otherwise, you risk giving away your money to automated thieves.

The version of Electrum included with BitKey lacks a QR scanner. QR is a very useful way of importing unsigned transactions and keys into Electrum. The fact that the version included in BitKey lacks a QR scanner means the user must type the keys into Electrum, instead of letting the webcam do the task.

International users will notice the lack of a documented boot code that lets you select a keyboard layout at boot time, such as you will find in Knoppix. With BitKey, you boot straight into an English keyboard layout. If you need another layout, you will have to switch it manually with `setxkbmap` from a terminal emulator.

Conclusion

BitKey is a very handy tool for managing Bitcoin securely, but it is certainly cumbersome and documentation is difficult to find. If you follow the recommended rules and procedures, the BitKey environment provides some protection for Bitcoin beginners to get started with minimal risk. Experienced users might prefer to use cheap Bitcoin hardware wallets [11] to can accomplish similar goals. ■■■

INFO

[1] Bitcoin: *https://bitcoin.org/en/*
[2] BitKey: *https://bitkey.io*
[3] Electrum Bitcoin Wallet: *https://electrum.org/#home*
[4] Electrum cold storage: *http://docs.electrum.org/en/latest/coldstorage.html*
[5] Paper wallet: *https://en.bitcoin.it/wiki/Paper_wallet*
[6] BitAddress: *https://www.bitaddress.org*
[7] Bitcoin Paper Wallet: *https://bitcoinpaperwallet.com*
[8] Bitcoin address reuse: *https://en.bitcoin.it/wiki/Address_reuse*
[9] Brainwallet: *https://en.bitcoin.it/wiki/Brainwallet*
[10] BitKey tutorial: *https://www.turnkeylinux.org/blog/secure-bitcoin-transactions*
[11] Hardware wallet: *https://en.bitcoin.it/wiki/Hardware_wallet*

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

How the OpenStack community is shaping the new Pike release

Changes at OpenStack

The quintessential open source cloud platform unveils a new development model with its latest release. *By Swapnil Bhartiya*

OpenStack has become boring. Boring is good in enterprise. It's a sign of maturity, stability, and consistent growth. OpenStack is showing all of that. However, it has also had its share of problems. With great adoption, comes great bloat. As OpenStack started getting deployed in new use cases, it started to see new projects being created by those users to address their own niche.

To maintain quality, the OpenStack project came out with an integrated release model, where new projects had to go through a vigorous incubation process. Once matured, they had to go through a voting process to become part of the integrated release. It didn't work out well, as many projects failed to meet the standards set by the OpenStack project. It was a dead end.

OpenStack tried to solve that problem in 2015 by moving away from an integrated release model to a Big Tent model. Under the Big Tent model, community members were free to work on their projects without having to worry about going through the incubating and voting process.

The Big Tent model allowed projects to work independently of OpenStack. It was a self-service model, where OpenStack stopped managing those projects and started offering refined processes and tools so that communities could manage their own projects. These projects also had an option to coordinate their release with the six-month release cycle for OpenStack. It was seen as a better solution to the integrated release.

But the Big Tent model brought its own set of problems, as the ever-growing community started building more services around OpenStack, instead of focusing on the core of the project. It also lead to confusion. It was not clear anymore what was part of OpenStack and what was not. Many projects were started but then left unmaintained. It became a bigger problem than the integrated release.

Mark Shuttleworth, CEO of Canonical, criticized the Big Tent model and said OpenStack must move away from "BS as a Service" and focus on the core strength that's compute, storage, and networking.

Mirantis cofounder Boris Renski agreed with that assessment and told me in an interview that Mirantis itself is withdrawing from such projects and is going to focus on adding value to the OpenStack ecosystem. Mirantis ended up letting go of many people who were working on such projects.

The OpenStack Foundation was paying close attention to the changing dynamics. OpenStack takes a lot of beating for it, but people fail to see that there has never been such a massive open source project that enjoyed such growth. Making mistakes, learning from them, and fixing them is the natural course of action. That's exactly what OpenStack is doing.

Early this year at OpenStack Summit, Boston, the OpenStack Foundation gave clear signals of moving away from the Big Tent model. A lot of work is happening at the code level, community level, structuring level, and communication level.

"We are basically getting rid of the Big Tent name and doing some restructuring of the project," said Lauren Sell, vice president of marketing and community services for the OpenStack Foundation. She told us that they will have large board meetings in September and October to discuss this in more detail. We are expecting announcements at the upcoming OpenStack Summit.

So what would the new model look like?

Talk Is Easy; Show Me the Code

At the previous Summit, Mark Collier, COO of the OpenStack Foundation,

Images courtesy of Swapnil Bhartiya

talked about OpenStack as a "composable" infrastructure that allows the use of a mix of tools and features depending on the use cases. That's where OpenStack is heading. It's becoming composable or modular.

The latest release of OpenStack, code-named Pike, brings this Open Source Infrastructure as a Service (IaaS) platform closer to that composability.

Jonathan Bryce, the executive director of the OpenStack Foundation, told us that some of that composable work has gone into the Pike release. Depending on use case, services like Cinder, Ironic, or Swift can be used on their own or in combination with other services. Bryce gave the example of Cinder, the block storage service for OpenStack, which can now be set up as a standalone service. It can be integrated with other projects like Docker and Kubernetes. Cinder now comes with a snapshot feature that allows for reverting to an older working version if something goes wrong. Users can extend storage volumes without having to shutdown the virtual machines (VMs).

What if a customer doesn't want to run VMs? Pike has improved integration between Cinder and Ironic (a service that provisions bare metal machines instead of VMs).

"If you want bare metal servers instead of VMs, you still need a way to image the operating system on them. It can now be done with Cinder," said Bryce. Users can have clouds with Cinder and Ironic, without any virtualization, and they can run containers or workloads like machine learning directly on it. Irrespective of the use case, composability of OpenStack now enables users to do what they need.

With this release, they have also improved the network integration, specifically around network segmentation. Users can run Ironic (bare metal provisioning) and Neutron (networking) to get full network segmentation between physical services, meaning multitenant applications run in the cloud (i.e., scalability).

This release brings scalability to the next level. The community has been working on evolving the way Nova (a core service that provisions Compute) environment is deployed. Users can start off with one cell, add additional cells as they need, and scale out the environment to many thousands of servers. It enables users to achieve horizontal scaling. It's a major feature that will continue to see further iterations into the next release of OpenStack, code-named Queen.

Another major work going on in OpenStack is toward containerization of OpenStack Control Plane. A lot of customers are taking these services, putting them in containers, and then managing those containers using tooling like Ansible and Kubernetes. Now a new project called OpenStack Helm that uses Kubernetes helps users manage these services as containers.

That's some of the higher level stuff that users will see in Pike. There is a lot of under-the-hood work, too. It might sound trivial, but OpenStack is upgrading from Python 2.x, which will reach the end of its life in 2020. What may look like a small upgrade is actually a pretty huge effort. The work has been going on for more than a year now, because OpenStack has a very large codebase, and they need to ensure that everything will function as expected. The upgrade will definitely bring new features and improvements. What's more important is that it will future-proof OpenStack as customers run a cloud for 10 years or longer.

What's Next?

Bryce said that the teams have been more focused on user experience, especially around managing a cloud and keeping it running for years. It reflects the change in dynamics within the OpenStack ecosystem. Companies are increasing focus on operations and services and looking for tools that are oriented around the life cycle of the OpenStack cloud. They are looking at environments they can operate, manage, upgrade, scale, and more.

Some of these new challenges and solutions will be showcased by the OpenStack community at the upcoming OpenStack Summit in Sydney. AT&T, for example, will demonstrate the set of tooling they are using to run hundreds of OpenStack clouds.

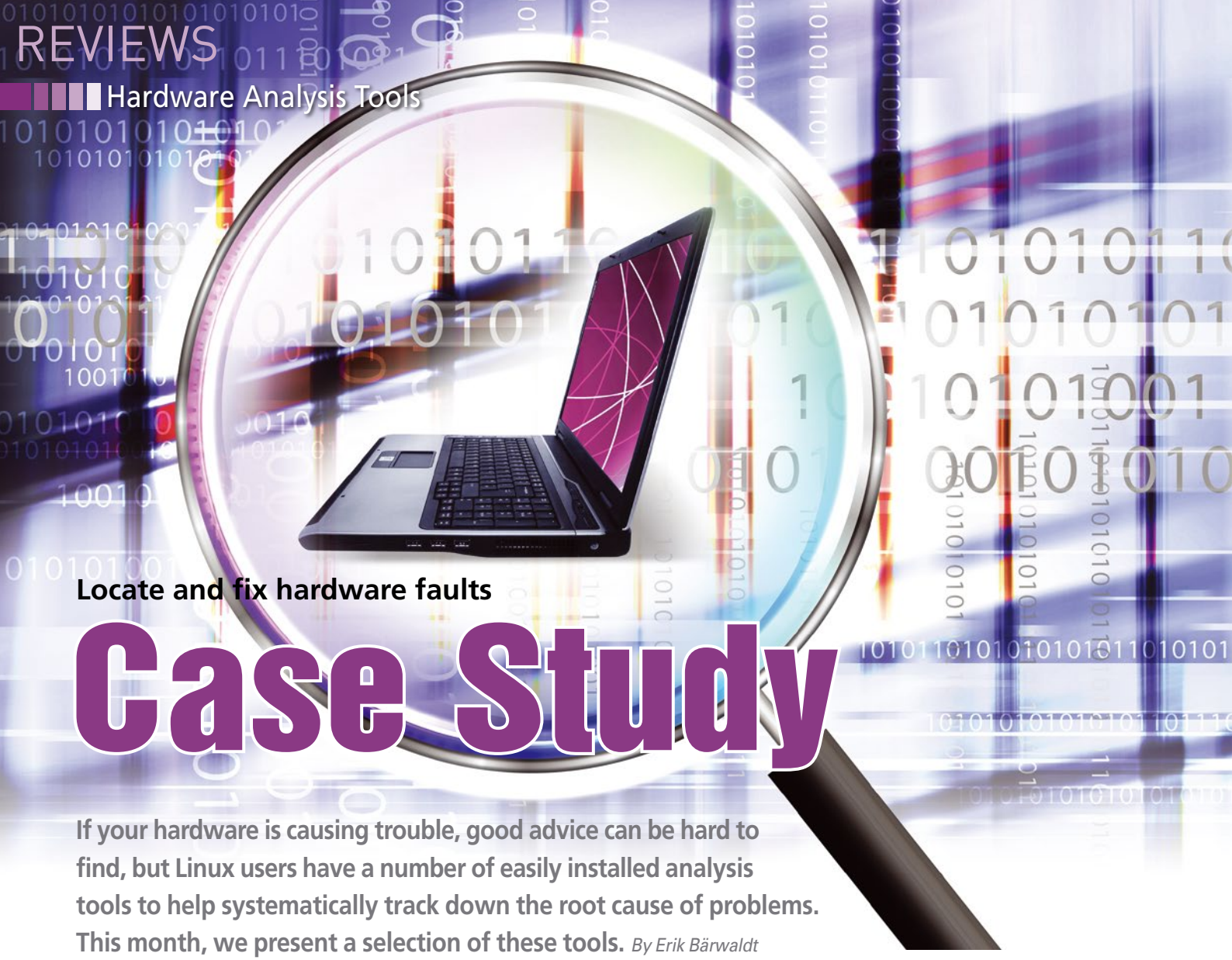
"That's what we like about the OpenStack community. We can have open discussions and put things out in the open and then the community responds and we continue to move forward," said Bryce.

To better serve this community, OpenStack has also adjusted its release cycle. Now a new version is released at least three months ahead of the flagship OpenStack Summit. The new release cycle gives developers enough time to discuss long-term goals at the Summit, which was not possible before.

OpenStack has come out with a new event called Forum, which takes place at the Summit. What's unique about Forum is that it brings together developers, vendors, and users who had been communicating on mailing lists into the same physical room. It creates a very dynamic environment for discussions and solving problems. Forum also brings together members of adjacent communities, like Ansible and Kubernetes, under the same roof.

"When you are at the Forum, you know that the next release is due in three months so you have three months of head start to look at pain points and actually have long term plans," said Bryce. Sell said that the Boston Forum was a huge success, and they are bringing it back to Sydney.

In a nutshell, the OpenStack bloat is gone, and it's becoming an "agile" and nimble IaaS platform. It's becoming boring again, but boring is good! ■■■



Locate and fix hardware faults

Case Study

If your hardware is causing trouble, good advice can be hard to find, but Linux users have a number of easily installed analysis tools to help systematically track down the root cause of problems. This month, we present a selection of these tools. *By Erik Bärwaldt*

Because computer systems are complex, problems are often difficult to identify when working on a PC or server. Besides software bugs and faulty configurations, hardware defects are the main cause of failures. An expensive and time-consuming replacement will obviously show which of the components is no longer working correctly; however, various Linux programs and kernel modules support troubleshooting and help you find problem configurations. In this article, I take a closer look at some of these tools.

System Components

Sometimes even identifying the hardware can be tricky. Laptops and convertibles whose components differ from current standards often make it difficult for users and administrators to identify the chipsets and GPUs or the status of individual assemblies. A hardware overview should determine the actual capacity of a laptop battery or the motherboard and BIOS revision.

Sometimes you can reach your target by replacing a flat battery or updating obsolete firmware. Especially with mobile systems, reading out the temperature values can point to error sources, such as a dried up thermal paste or a dusty fan that is consequently working inefficiently.

You can't normally expect software to repair your defective hardware, but targeted tests reveal error sources. The right choice of analysis software helps to isolate these errors quickly. Sometimes, proactive replacement of individual components makes sense, for example, for disks that show the first symptoms of impending faults. Hardware analysis tools and their test routines make a valuable contribution to data security.

inxi

While the features included in most graphical desktop environments often only support superficial diagnosis, the inxi [1] command-line tool, which is based on dmidecode [2], provides a detailed overview of the hardware installed on the system. Inxi is available in almost all software repositories and is therefore easily installed with your distribution's package manager.

The `inxi -help` command displays the script's very extensive parameter list. Inxi primarily provides detailed hardware information, but it also lists some operating system-specific data, including the kernel version, number of processes, memory consumption, init system with version number, run level, and desktop environment. The `inxi -v 7` command, which the admin runs with root privileges, provides a largely complete list, including the temperature values reported by sensors in the system (Figure 1).

Although inxi will run with normal user rights, you won't see all the data: Details about the installed memory, such as the manufacturer, clock speed, bus width, and slot assignments, are missing in user mode. In addition to hardware-specific information, the software dis-

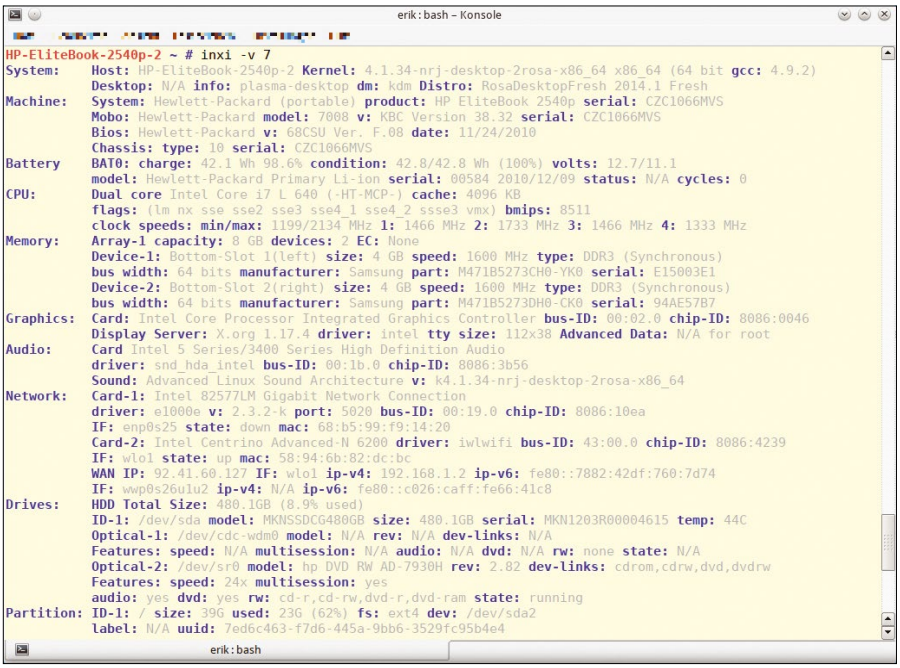


Figure 1: Inxi is visually very plain but technically provides very detailed information about installed hardware.

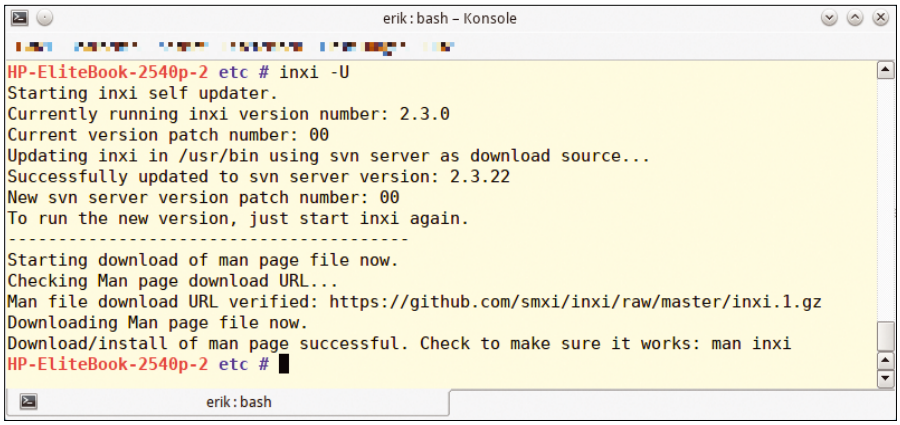


Figure 2: Inxi has its own update function to detect the latest hardware.

plays the BIOS firmware revision and the exact partitioning data for the mass storage media, and it identifies network cards, including wireless WAN hardware. If the server has a RAID array, it also appears in the output with all the relevant details. The software reads the laptop batteries' firmware and provides information on defective or exhausted cells based on the capacity data and – if available – the number of charge cycles.

If you don't need the full output, inxi lets you display individual, optionally combinable parameters in a short listing. Additionally, you can set the level of detail for the program output with the `-v <detail level>` parameter, which ranges from `0` to 7. The use of color combinations makes the output clearer and easier to read.

Updates

You should always keep inxi up to date, because new hardware is constantly coming onto the market. Thus, older versions may display components and their data inaccurately or incompletely. Inxi can be updated using the package manager, by entering `inxi -U` with root privileges. The software then checks the version status and, if necessary, loads a newer version off the Internet; this also updates the man page (Figure 2).

Because of its detailed detection function, inxi is very well suited to broadly isolate hardware problems: If certain components do not show up in the detail view or data is incorrect (e.g., the amount of RAM), you can draw conclusions about possible defects.

SMART Monitoring

Problems with mass storage are particularly critical, because, in the worst case, they result in data loss. Therefore, as early as the 1990s, several leading hard disk manufacturers collaborated with IBM to launch the Self-Monitoring, Analysis and Reporting Technology (SMART) standard [3], which, as a diagnostic tool, proactively warns you about mass storage medium failures. SMART technology works with threshold values and has long been integrated into virtually all mass storage devices. Even modern SSDs use parts of it.

On Linux, the Smartmontools package [4] takes care of testing and evaluating SMART data. Almost all distributions have this collection of command-line tools on board; therefore, you can easily install it with your distribution's package manager.

Smartmontools, a command-line program run in a terminal, is now supported by various graphical front ends, probably because of the extensive set of command-line parameters. Most of these GUIs are tailored to individual desktop environments, and they often support both reading the mass storage medium's various operating parameters and options for running bench-

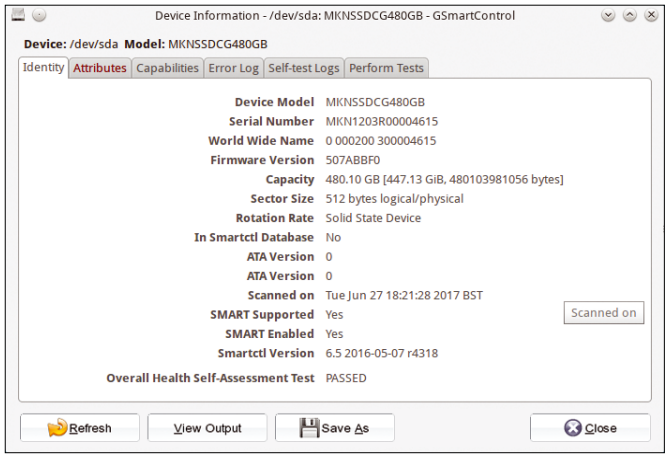


Figure 3: GSmartControl's summary page shows important drive parameters in abbreviated form.

marks or testing the hard disks or SSDs built into the system. The most well known are:

- GSmartControl (Gnome) [5]
- DisKMonitor (KDE) [6]
- Gnome Disk Utility (Gnome) [7]

All three front ends display the selected parameters of the mass storage medium in a table and additionally run test routines (Figure 3).

Poor Values

Smartmontools normally allows both a quick test and a more in-depth test (Figure 4), but even if the drives pass all tests, individual values sometimes indicate problems.

For example, with conventional hard drives in desktop systems, it is a good idea to take a look at the *load-cycle-count* value shown under ID 193 in the table. In some Linux distributions, the hard disk drive heads move to park position too frequently because of overly aggressive default ACPI settings. This not only means excessive mechanical wear and tear but also spoils any attempts to save energy: When repositioning from the park position, the heads need far more energy than they do in normal use.

If the table shows a six-digit value at this point, you will want to back up your data; all major hard drive manufacturers state a mean value between failure figures for mass storage media of between 300,000 and 600,000 load cycle counts, depending on the model.

If you then compare the load cycle counts with the values displayed in the *power-on-hours* or *power-on-time* section of the SMART table and determine that the hard drive heads have often parked despite a relatively short period of operation, it is advisable to adjust the ACPI values to stop this undesirable behavior [8].

Smartmontools and its graphical front ends do not monitor the state of the overall system; rather, they focus on mass storage media, which also includes optical drives. Problems can arise if the external hard disks you are monitoring use a USB or IEEE 1394 adapter (Firewire). The signal converters required for these mass storage media often fail to pass through the SMART values; in this case, the drives cannot be analyzed.

The `badblocks` command-line tool that is integrated into the *e2fsprogs* tool collection [9] can then step into the breach, if need be. Most Linux distributions come with the tool installed; the following command launches a non-destructive read-write test:

```
badblocks -n -s -v /dev/<drive label>
```

Here, the progress and results appear in the terminal, but be careful: If you use the application incorrectly, you risk losing data. It is strongly recommended that you read the corresponding man page before use or call the tool with the `--help` parameter to learn the command syntax.

hdparm

Hdparm [10] is a very powerful tool for modifying hard disk parameters. The software is provided by virtually all Linux distributions, it can be installed – if not already preinstalled – using the respective package manager. Since the program can permanently change the settings of mass storage media, you should also carefully study the documentation before using the tool's various options; this will help you avoid potential data loss or even damage to the hardware from incorrect settings.

For an initial overview of the hard disk parameters, enter

```
hdparm -I / dev/<drive label>
```

at the prompt with admin rights; this syntax will usually correspond to block device `sda` in current Linux distributions. The software then outputs all the relevant data in a list (Figure 5). While doing so, hdparm can hide information not relevant for the type of drive: For example, a rotational speed does not appear for a flash drive.

Constant Flow

Hdparm offers the option to test data throughput with the `-t`, `-T`, and `--direct` parameters. If you notice poor performance of a computer system, the throughput test helps you track the cause of the problem because the tool determines the throughput for the processor, system cache memory, and RAM using the `-t` and `-T` parameters. You need to launch the tests several times in succession to obtain a realistic mean value.

If you set the `--direct` parameter in conjunction with `-t`, hdparm bypasses the cache and reads data directly from the specified drive. Therefore, this test is well suited to testing the drive performance, depending on the interface used. Poor throughput rates often indicate problems with mass storage media. With conventional hard disk drives, in particular, hdparm helps to improve read rates, and thus throughput values, using many optimization parameters. However, incorrect application of certain options entails a risk of data loss.

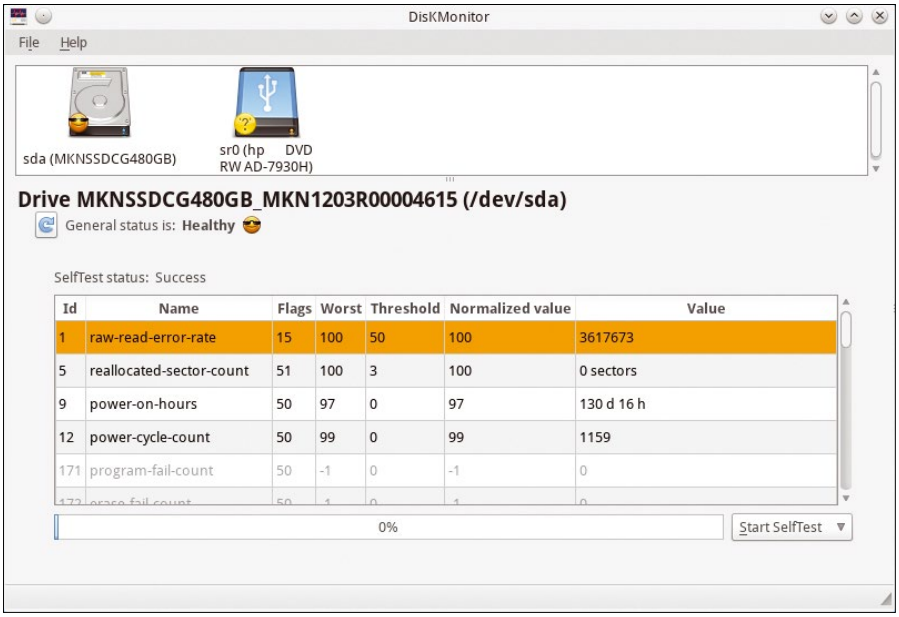


Figure 4: KDE DisKMonitor showing the operating values for the active drive in the input screen.

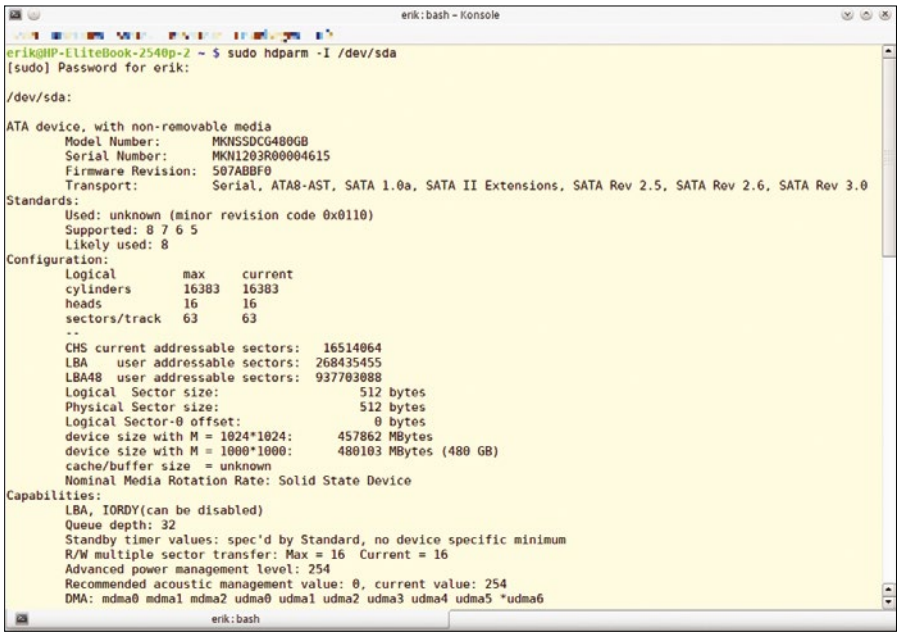


Figure 5: Hdparm provides data relating to mass storage devices and lets you change the parameters.

Power Tool

The software can also help reduce energy consumption, mainly in the case of hard drives, and thus reduce mechanical wear. For example, in the case of aggressive power management, the `-B` parameter reduces the number of park operations for the hard disk heads, which also prevents premature spin-down of the hard disk, which, in turn, can reduce energy consumption in many cases because constantly starting up the disk drive requires significantly more energy after a spin-down than continuous operation at idle speed without switching off the drive motors.

The values for power management that follow the `-B` parameter range between `1` and `255` for most models; the lowest value generally triggers a complete spin-down, whereas the highest value completely disables power management. However, because the individual values vary depending on the hard disk vendor, and partly on the model, several test runs are usually required to achieve the best value.

The `--idle-immediate`, `--idle-unload`, and `-S` parameters can be used in combination to reduce energy requirements and help reduce mechanical wear. With Hitachi hard drives, hdparm uses the `-H` parameter to read the current operating temperatures; this is useful for older Microdrive hard disks, in particular.

Security

Hdparm also assigns drive passwords on the basis of the ATA standard by using various parameters. The result is that third parties can only access the device in question after authentication. Using the `--security-erase` and `--security-erase-enhanced` options, you can securely delete drives so that the deleted data cannot be reconstructed. In the case of most drives, an overview of technical features in the *Security* section will tell how long the deletion process will take.

MemTest86

If you suspect that a computer system's memory is defective, the MemTest86 tool [11] can provide valuable assistance to help locate the damage. MemTest86 is included in the standard scope of all major Linux distributions and is available as a dedicated Live distribution. Depending on the image, you can either burn it on an optical disc, or use a USB memory stick.

Keep in mind that the images provide two versions of MemTest86: New systems with UEFI BIOS need to launch version 7, whereas version 4 helps manage older computers with a conventional BIOS. Both will perform repair actions automatically, if so desired.

The current version 7 is available in a Pro version for $39 including support options and updates for six months. This version also includes a memory benchmark, as well as a reporting function and several improvements for automated memory testing.

For very old computer systems, such as industrial computers with long operating periods, the tool is also available on a floppy disk [12]. Additionally, many distributions offer to run MemTest86 as a boot entry in the GRUB 2 bootloader. If the computer system is causing trouble, you can perform a memory test by rebooting and selecting the MemTest routine.

I recommend that you always use the latest version, because the developers of current, specific CPU families regularly integrate advanced functions into the software. MemTest not only detects memory errors, but also identifies most memory controllers and reads the memory chip vendor information.

Dual

While the older versions 4 and 5 still rely on an ncurses interface, which every computer system can display, the developers have upgraded and functionally changed the visual appearance of version 7. The new version no longer runs on computers with a conventional BIOS. If you want to use it, you need to change the boot process to a UEFI-only boot in the BIOS settings. The software does not accept combined BIOS settings that also use a legacy BIOS. In such cases, it launches the old MemTest version.

In the old versions, MemTest86 provides different boot options in the boot manager; the software also supports automated multiple runs. At top left on the screen, the tool displays CPU and memory data, including the data throughput for different cache storage levels. The individual test routines and their progress indicators appear at top right in the form of a buttonbar. The lower area of the screen displays memory errors. When a test run ends, the software displays a message at bottom center and indicates the number of memory errors detected (Figure 6).

Because MemTest86 restarts the tests after a successful run, you can see how many iterations the software has already carried out in the *Pass* section in the list at the bottom of the screen. As a

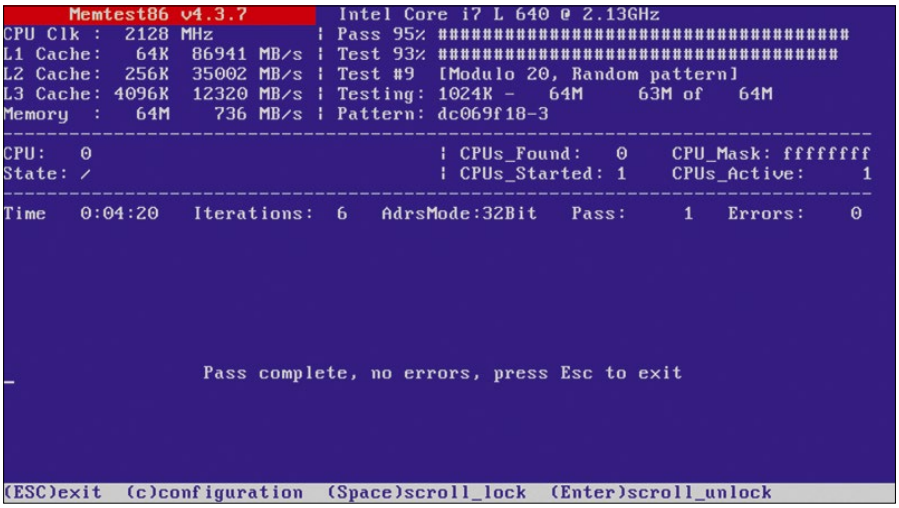


Figure 6: MemTest86 is visually very simple but has in-depth storage tests.

rule of thumb, the more test runs MemTest86 completes without error, the more certain you can be that the memory is not faulty.

Using UEFI

The UEFI version of MemTest86 provides a visually enhanced interface. After launching the program, click on the *Config* button on the right of the screen to open a two-pane display: On the left, you will see a list of program options, and on the right, the much larger screen segment visualizes the software's activities and the results of the memory test.

After enabling the segmented display, various information about the processor, memory, and cache memory appears in the *System Info* tab on the left. You will also see the *Test Selection* tab, where you can choose from 14 different test routines and define the number of test passes. In the *Address Range* tab, you can define optional address ranges if you don't want to test the entire memory.

Then determine the number of CPUs that the test has to include in the *CPU Selection* tab, which does not refer to physically existing processors but refers instead to cores. The selection lets you use either one or all CPU cores and define which algorithm to use when testing all CPUs.

Initiate the test run by pressing the *Start Test* button. MemTest86 then shows an ncurses screen, like the old interfaces, that tracks the progress of the tests. After a successful run of routines, pressing any button generates a report and displays it on the screen, revealing any memory errors found so that you can replace defective components. The report can also be saved in the form of a specially generated file for documentation purposes.

HardInfo

One of the most popular information tools with built-in test routines for Linux is HardInfo [13]. The program is available in almost all current package managers. Depending on the distribution, the installation routine creates a *System Profiler and Benchmark* or *System Information* launcher in the System or System Tools submenus. When clicked, HardInfo comes up with a two-pane, easily manageable window.

In the left segment of the window, the software sorts the hardware detected in your system into categories. Clicking on one of the components – they are also represented by symbols – displays detailed data on the respective component or assembly to the right (Figure 7). If several devices show up in a category (e.g., for drives or PCI devices), clicking on one of the devices at top right in the window splits the right-hand pane; then, detailed information is displayed at bottom right.

Since the respective device information does not yet provide any conclusions on the functional capability of the component, HardInfo uses benchmark tests to trace irregularities. A *Benchmarks* group in the left part of the screen contains multiple CPU and FPU tests. Using this, you can measure the performance of the current system and determine weaknesses in the computer's cooling system from a high CPU or FPU load in the respective tests.

HardInfo is particularly useful on laptops, where you can use the benchmarks to check whether the CPU fan needs cleaning or whether you need to reapply thermal compound. If network access problems arise, HardInfo provides information on the data throughput in the *Network* | *Statistics* entry in the left pane.

For the Record

The *Generate Report* button in the small buttonbar below the menubar, lets you generate a test results summary. If necessary, you can restrict the

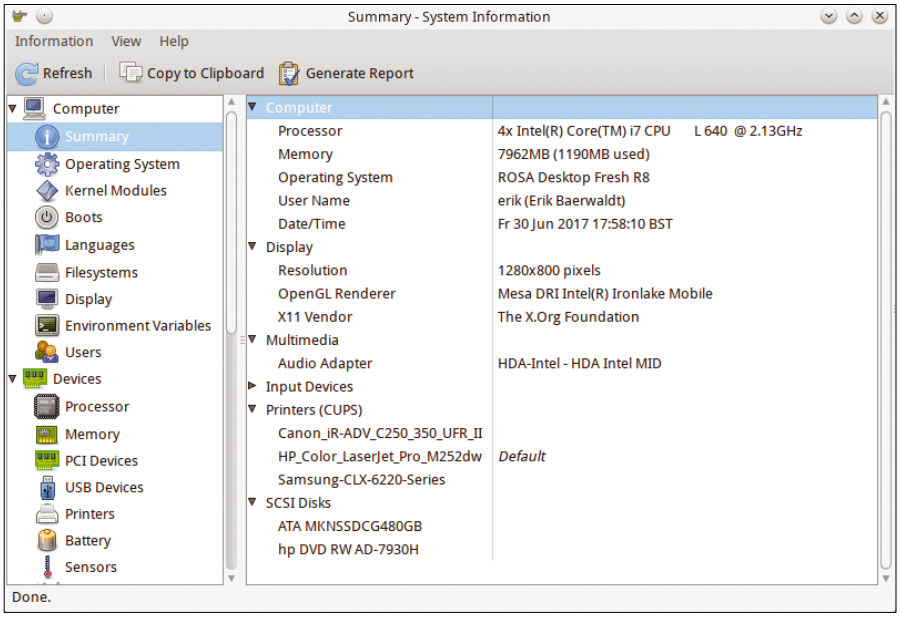


Figure 7: HardInfo displays the individual system components in a clear-cut format.

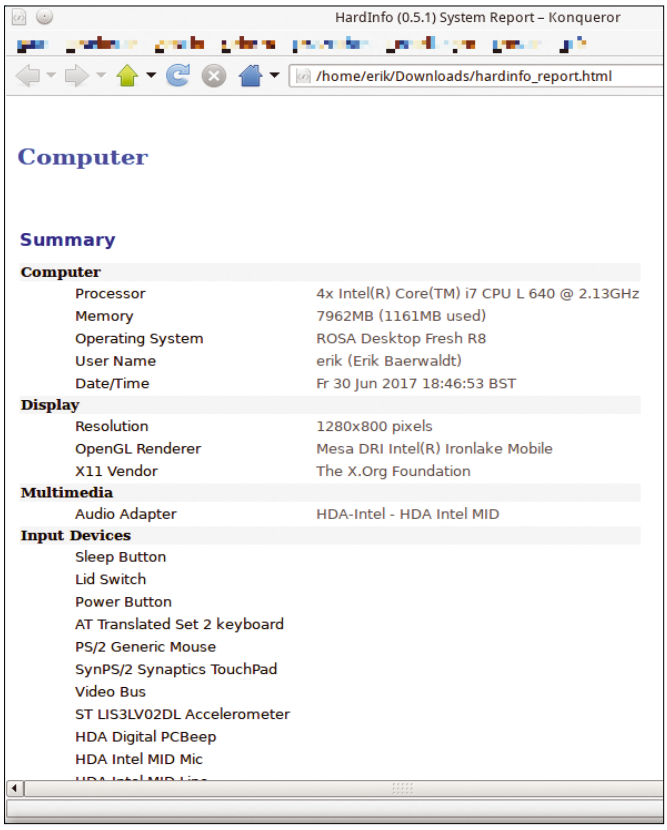


Figure 8: The HardInfo report provides a good system summary in the form of a simple HTML file.

report content to certain component groups in a separate window in order to view only the relevant information. HardInfo then stores the report as a simple HTML file in a directory of your choice (Figure 8).

Checkbox

Checkbox [14] is a piece of software for testing hardware components that Canonical uses for Ubuntu certification. The application requires an installed Python environment and is available in several versions. In addition to a command-line version, Checkbox with a Qt-based graphical user interface is available.

The tool collection partly uses internal Linux commands to detect the hardware. Ubuntu and Debian users install Checkbox directly from their repositories; no precompiled packages exist for other distributions.

After you call the application, Checkbox opens a very simple window without controls in the graphical version and first scans the hardware internally. You then click on the *Start Testing* button. A list with various test categories appears (Figure 9). You can disable it on the right side of the window by removing a check mark.

After clicking n *Continue*, the software lists the individual tests sorted by main category. The tests can also be disabled if necessary.

The test is divided into passive and interactive routines: For example, while CPU and storage tests run without the user's help, the functional test for multimedia components such as speakers, headphones, microphones, and cameras, as well as memory card slots, requires user interaction. After a run, you actively need to tell the system whether or not it has passed the test by clicking on *Pass* or *Fail*. For those routines for which Checkbox runs a third-party program with administrative privileges, the software requires authentication.

Because Checkbox also runs a number of benchmarks, the total number of categories can be as high as 100 or more. Thus, a complete test of the entire system can take quite a long time if all the categories are enabled. It is thus a good idea to deactivate any routines you do not need.

Jump to the next dialog window by clicking the *Start Testing* button at the bottom of the window. For interactive tests, the program window will first display a brief note regarding the purpose of the test; then click on *Start the Test*. After the run, you need to confirm whether or not the system has passed the test. If a routine fails, a corresponding display shows you this in the program window.

In the following section, you then decide whether you want to repeat the test by clicking on the *Back* icon top left in the window or jump to the next routine by clicking on *Continue* at the bottom of the window.

Reporter

Once Checkbox has run all the required tests, the software displays a pie chart in the program window that represents the passed, failed, and skipped routines in absolute terms. A report, which is generated by clicking on *Save Detailed Report* at the bottom of the window, supplements this less than useful display. The report lists the completed test routines in neat groups and is stored in the `Documents` folder in your home directory.

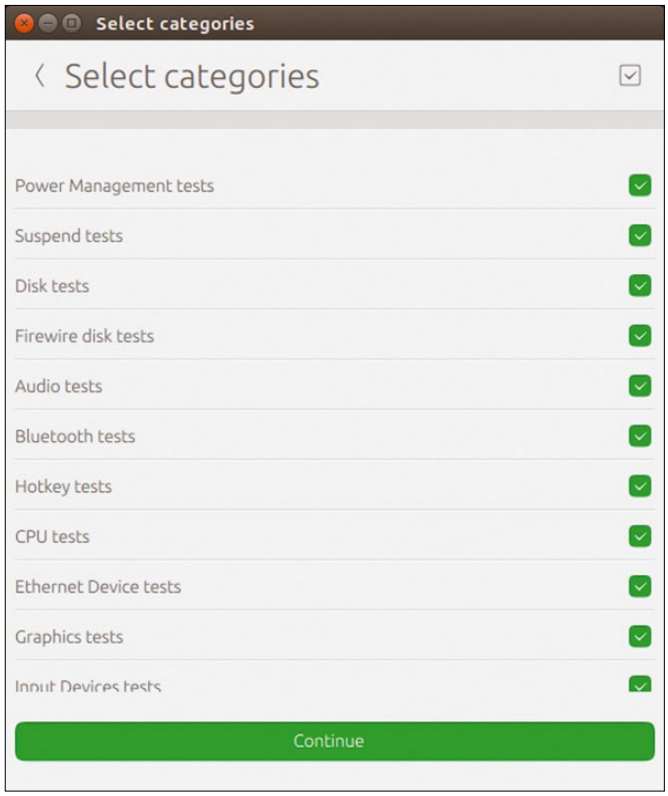


Figure 9: Select the tests to be completed in Checkbox by removing the corresponding check mark.

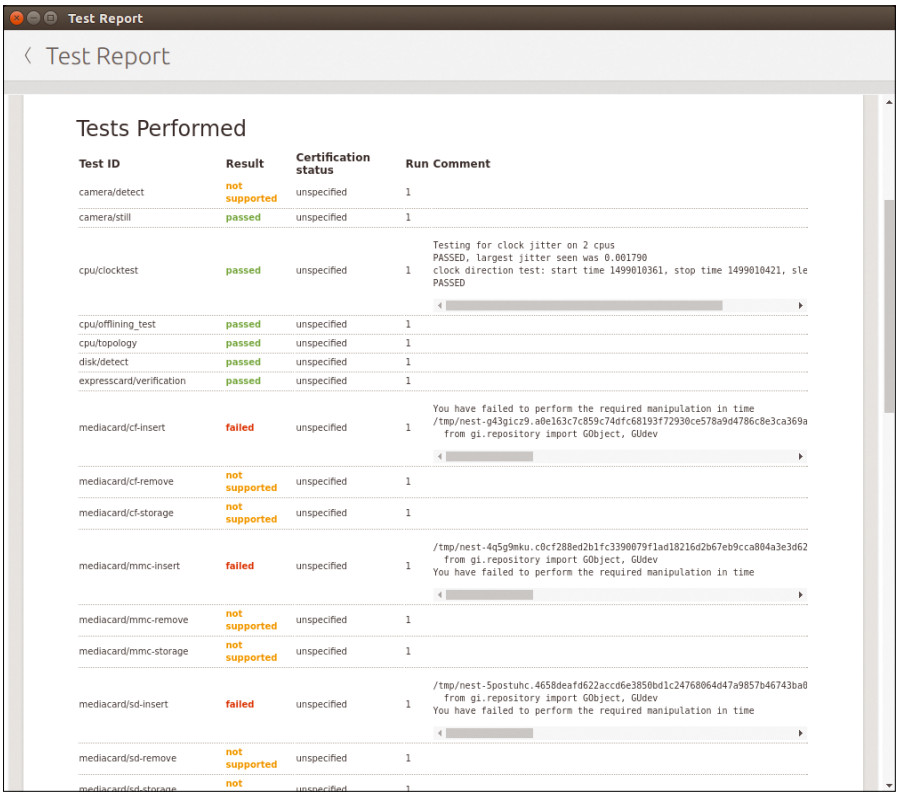


Figure 10: The Checkbox report reveals many details and is also visually appealing.

In the Comment section, the software provides some notes on the outcome of individual routines, so you can find out which test runs have caused problems. This enables detailed conclusions on the support for, and functional capability of, individual tested components (Figure 10).

Stress

The Stress program lives up to its name [15]. This small command-line tool tests the limits of hardware components by artificially generating load. The software focuses on the CPU, as well as the mass storage and RAM of a system. Stress is available from the repositories of virtually all major Linux distributions and can be installed in the package manager.

The man page or output from the `stress -?` command informs you of the software's command syntax. The output also shows examples.

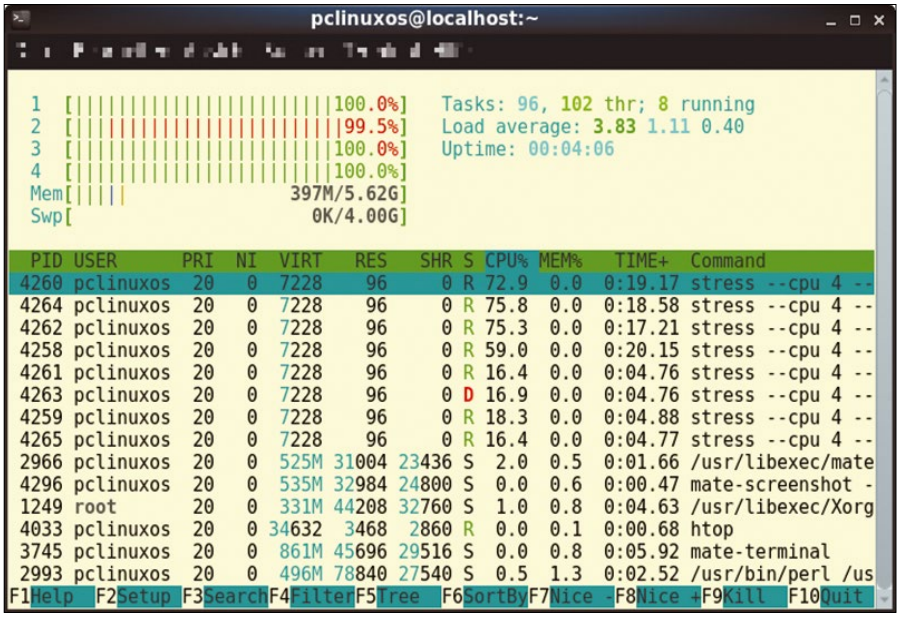


Figure 11: With `top` or `htop`, you can detect the extent to which Stress is stressing your computer.

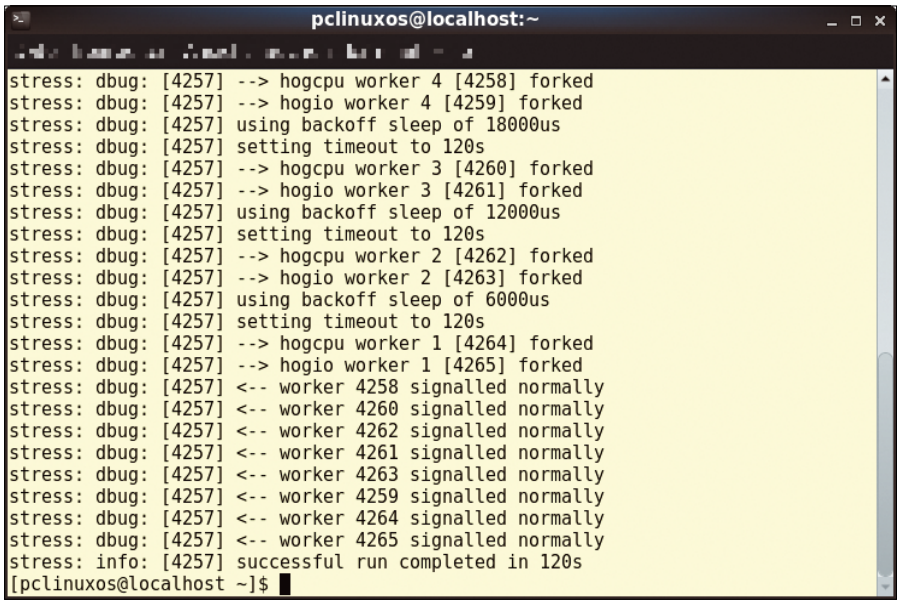


Figure 12: Stress succinctly informs the user as to whether or not the tests were successful.

Stress is a very simple tool; it doesn't generate any reports and is quite reserved when displaying information in the terminal. It is always advisable to specify the `-v` parameter and observe the load on the system in a second window via `top` or `htop`. For example, the command

```
stress -v --cpu 4 ➥
          --io 4 ➥
          --timeout 20s
```

generates a full load on the CPU and the I/O subsystem for 20 seconds (Figure 11).

Stress succinctly informs you in the terminal when the stress test is complete, with no further details on the generated workload (Figure 12). Because of its relatively simple routines and optionally configurable time out, Stress is especially suitable for intensive long-term CPU tests that simulate the stability of a computer system under full load.

Conclusions

The test and analysis programs presented here are very much heterogeneous in terms of their functionality and appearance. Other than Checkbox, you will almost always find the tools in the repositories of the major distributions, and the tools are all fairly easy to install. Thus, you can put them in place quickly on machines already showing signs of faulty behavior. (For additional analysis tools not tested in this article, see the "Untested" box.)

The software tested for this article ranged from predictive analysis for mass storage to test routines that detect hardware defects. The components included benchmark tests that exposed a wide range of components to full load and thus, in part, helped to locate thermally induced or sporadically occurring problems. Regardless of the individual user scenarios, these analysis tools also ensured a more complete understanding of hardware interactions and functional capability. In this respect, you will not want them missing from any of your systems. ■■■

UNTESTED

In addition to the tools discussed here, there are other tools that were not considered in this article, including the well-known benchmark Bonnie++ [16] and the equally widespread Phoronix Test Suite [17], among others. We did not consider Bonnie++ in our lab because it is biased toward mass storage benchmarking.

The Phoronix Test Suite provides many practice-oriented benchmarks for comparison purposes with other computer systems. These are based on application-specific performance measurement and are therefore less suitable for identifying problematic hardware components.

One of the benchmarks offered by Russia's UNIGINE LLC [18] is limited to graphics cards and is also application-specific. LuxMark [19], an OpenCL benchmark also primarily focuses on applications with a rating system and only targets graphics hardware.

INFO

[1] inxi: *http://smxi.org/docs/inxi.htm*
[2] dmidecode: *http://www.nongnu.org/dmidecode/*
[3] SMART standard: *https://en.wikipedia.org/wiki/Self-Monitoring,_Analysis_and_Reporting_Technology*
[4] Smartmontools: *https://www.smartmontools.org*
[5] GSmartControl: *http://gsmartcontrol.sourceforge.net*
[6] DisKMonitor: *https://github.com/papylhomme/diskmonitor*
[7] Gnome Disk Utility: *https://git.gnome.org/browse/gnome-disk-utility/*
[8] "Linux and Laptops" by Erik Bärwaldt, *Linux Magazine*, February 2012, pg. 34
[9] e2fsprogs: *http://e2fsprogs.sourceforge.net*
[10] hdparm: *https://sourceforge.net/projects/hdparm/*
[11] MemTest86: *http://www.memtest86.com*
[12] Floppy disk version of MemTest86 v5.01: *http://www.memtest.org/#downiso*
[13] HardInfo: *https://github.com/lpereira/hardinfo*
[14] Checkbox: *https://launchpad.net/checkbox-project*
[15] Stress: *https://people.seas.harvard.edu/~apw/stress/*
[16] Bonnie++: *http://www.coker.com.au/bonnie++/*
[17] Phoronix Test Suite: *http://phoronix-test-suite.com*
[18] UNIGINE: *http://unigine.com/en/company/profile*
[19] LuxMark: *http://www.luxrender.net/wiki/LuxMark*

Create a select menu with smenu

Not Spoiled for Choice

The smenu tool reduces the effort of creating shell menus to one line, with numerous options for a wide range of design alternatives. *By Harald Zisler*

If you program for the shell frequently, sooner or later you will have to create a select menu, which usually requires several lines of code – unless you use smenu. Smenu reduces the menu's script to a single line. A number of parameters allow you to adapt the design and to simplify your work a little.

The source code for smenu is available on its project page [1], which contains installation instructions, as well as some useful tips. The smenu wiki [2] contains links to two YouTube videos that demonstrate the practical use of the program. The README file provides a structured overview of smenu's options [1].

AUTHOR

Harald Zisler has worked with Linux and FreeBSD for many years. He writes magazine articles and books on the topics of technology and computing. The fourth edition of his book *Computer-Netzwerke* was recently released by Rheinwerk Publishing.

Basic Function

In smenu's simplest form, a command's output is passed via pipe to the `smenu` command. For example, `<command> | smenu` displays a selection on a single line. To create a line-by-line menu, use `<command> | smenu -c` (Figure 1).

LISTING 1: Command Substitution

```
VARIABLE=$([COMMAND] | smenu)

VARIABLE=`[COMMAND] | smenu`
```

```
$ # Selection in one line
$
$ ls -1 | smenu
1.txt 2.txt 3.txt 4.txt 5.txt
$
$ # Selection line by line
$
$ ls -1 | smenu -c
1.txt
2.txt
3.txt
4.txt
5.txt
$
```

Figure 1: Pipe command output to smenu and leave it to the program to prepare the output in the form of a single-line menu.

```
$ # Selection in one line
$
$ echo "A B C" | smenu
A B C
$
$ # Selection line by line
$ echo "1 menu item A
2 menu item B
" | smenu -c
1 menu item A
2 menu item B
$
```

Figure 2: If you want to enter the selection options with your own text, use the `echo` command.

```
dd@dd-kubu1704x64-vm:~/smenu$ ./menu1.sh
1 first
2 second
3 third
input value: 2
dd@dd-kubu1704x64-vm:~/smenu$
dd@dd-kubu1704x64-vm:~/smenu$ ./menu2.sh
1.txt
2.txt
3.txt
4.txt
5.txt
Selected file: 4.txt
dd@dd-kubu1704x64-vm:~/smenu$
```

Figure 3: Listings 2 and 3 show simple scripts that demonstrate how you can easily create a menu for your software with smenu.

TABLE 1: smenu Options

Option	Action	Tip/Example
-C	Column mode	
-m "<Text>"	Heading	Listing 4
-n<Lines>	Limit lines in selection	Listing 4
-M	Center menu	
-d	Delete menu according to selection on the screen	—
-t<Column>	Selection with several columns	Listing 5
-w	Selection with multiple columns at full terminal width	Together with -t
-T <Field Separator>	Allow multiple selections	Listing 6
-e <Expression>	Exclude the transferred string from the selection	Listing 7
-i <Expression>	Only allow designated strings to select	Listing 8

LISTING 2: A Selection with Explanations

```
#!/bin/sh
a=$(echo "1 first \n 2 second \n 3 third \n" | smenu -c)
echo "input value: $a"
```

LISTING 3: Selecting a File

```
#!/bin/sh
b=$(ls *.txt | smenu -c)
echo "Selected file: $b"
```

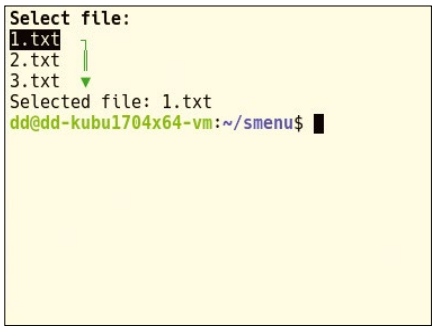


Figure 4: The green scrollbar shows that there are more items from which to choose.

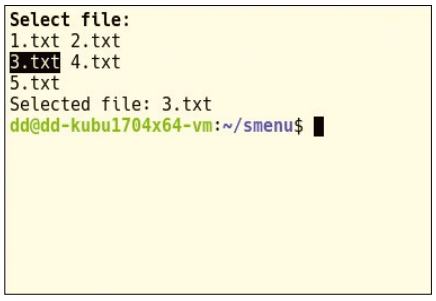


Figure 5: If necessary, distribute the data for the selection to multiple columns.

LISTING 4: Specifying the Heading

```
#!/bin/sh
clear
a=$(ls -1 *.txt | smenu -n3 -c -m "Select file:")
echo "Selected file: $a"
```

For a freely defined menu, transfer the selection options using the `echo` command. However, the examples in Figure 2 only show the most rudimentary form: It doesn't make sense to use the tool without variables.

Table 1 lists the most important call options for smenu. Press Enter to select a menu option, and press *Q* to exit the menu without selecting. To search within the selection, use the forward slash (*/*) in the open menu and then enter the search term.

When the search completes, the cursor stops at the first hit, and the background appears blue instead of black. If you wait seven seconds, the software reverts back and waits for your next input. Typically, the selection results from the data obtained via the pipe.

Script Use

To use the result of the selection, apply command substitution, as shown in Listing 1, line 1 (current syntax) and line 2 (obsolete syntax). Listing 2 and Listing 3 show two functional examples, and Figure 3 shows their usage.

Listing 2 demonstrates how to establish a selection with explanations. The end-of-line symbol (newline), which you represent with an escape sequence (`\n`), serves as a separator between the entries. Within an entry, the space serves as a separator between fields. The result adopted in the variable is the value from the first column. Smenu only inverts the current value of the variable on the screen, not the full line.

Listing 3 shows just how easily users can select a file interactively.

Improvements

Figure 4 shows a small menu, which already has some additional features. If the list of entries reaches a certain number of lines, a green scrollbar appears.

If you limit the number of visible lines to three with `-n3`, the bar appears as soon as the length of the list exceeds this value. Use the `-m "<text>"` option to specify the heading. Listing 4 shows the corresponding code.

Look at the scrollbar in Figure 4 more closely: If you reach the start or the end of the selection, the arrow symbol appears. The arrows indicate more entries in the respective directions.

If you need a selection with multiple columns, type the number of columns using the `-t` option (Listing 5). Figure 5

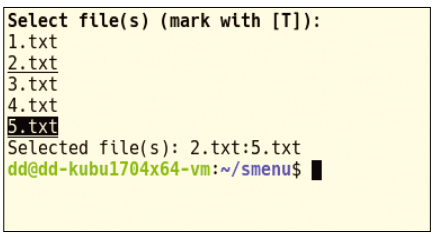


Figure 6: Some applications require the selection of multiple entries.

LISTING 5: The -t Option

```
#!/bin/sh
clear
a=$(ls -1 *.txt | smenu -n3 -c -m "Select file:" -t2)
echo "Selected file: $a"
```

LISTING 6: Using a Colon as a Field Separator

```
#!/bin/sh
clear
a=$(ls -1 *.txt | smenu -n5 -T: -c -m "Select file(s)
  (mark with [T]):")
echo "Selected file(s): $a"
```

LISTING 7: The -e Option

```
#!/bin/sh
clear
a=$(ls -1 | smenu -n5 -c  -e [m][4] -m "Select file: ")
echo "Selected file: $a"
```

LISTING 8: The -i Option

```
#!/bin/sh
clear
a=$(ls -1 | smenu -n5 -c  -i [t][x][t] -m "Select file: ")
echo "Selected file: $a"
```

shows the result on the terminal screen. If you want to expand the line content to the terminal's full width, use the `-w` option.

Multiple Choice

The `-T` option allows for the selection of multiple points by pressing the *T* or Insert keys in the running script for each relevant point. Complete the selection by pressing Enter. A marked entry appears underlined. Remove the marker by repeatedly pressing *T* or Delete.

The software uses the space as a field separator, unless you specify otherwise. However, the example shown in Listing 6 (Figure 6) uses a colon, which allows you to use tools (e.g., `cut`) during further processing of the variable's contents, to address a column.

Menu Preselection

If you have no way to limit the list of menu options from the command in the pipe, filter the data when you call `smenu`. If necessary, use regular expressions (`-e` option, Listing 7). Alternatively, work with the positive hits (`-i` option, Listing 8).

The effect is generally the same: The unwanted entries appear in a color that offer less contrast and cannot be selected (Figure 7). When using both options simultaneously, keep in mind that exclusion has a higher priority. As a result, you might not get any items to select.

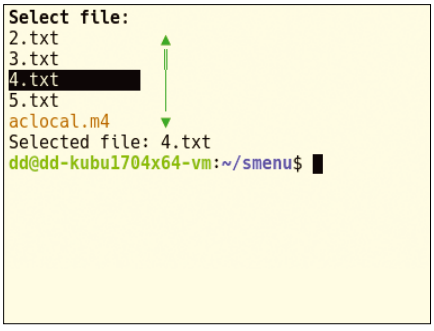


Figure 7: If the software supplying the data offers no way to filter it, use smenu functions to include or exclude points.

PDF Search

To browse a (quite large) portfolio of PDF files, use the `pdfsearch.sh` shell script (Listing 9). Using smenu, the small program displays the selection of hits and allows you to take further actions.

The script only runs under Bash because it uses the `read` command. Either assign the appropriate rights to run it directly, or call it with:

```
bash pdfsearch.sh
```

(For PDFs, check out `qpdfview` or Evince: Both accept a search term at launch.)

When launching, enter the search term (Figure 8), which the script then transfers to `pdfgrep`. It shows only the number of hits, ignoring case sensitivity. Any hits are provided for selection via smenu.

If you have selected a file to display, the script browses the search term in the PDF viewer. It then highlights it in the document (Figure 9). After closing the display program, either reuse the selection or enter a new search term (Figure 10).

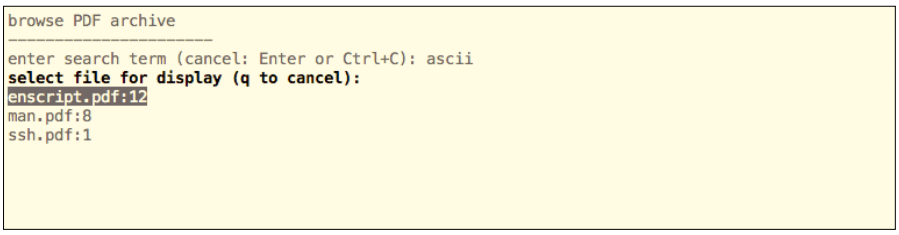


Figure 8: After entering a search term, the search launches and provides a selection of hits.

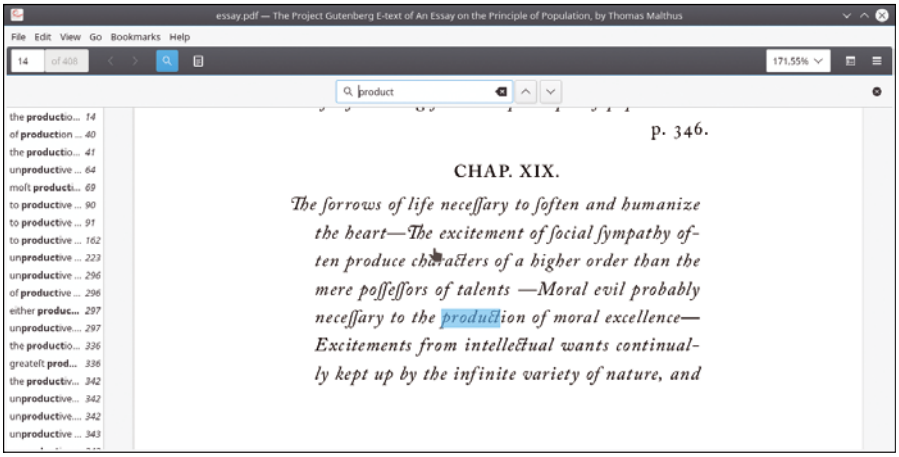


Figure 9: Certain programs allow you to transfer a term when launched, which the software then highlights in the document.

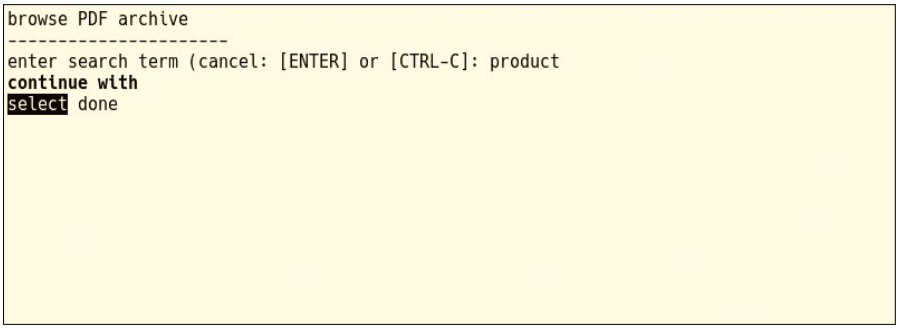


Figure 10: After viewing a PDF document, the script gives you the option of displaying another file or launching a new search.

Conclusions

Setting up a custom selection menu with shell functions is quite a lot of work. Smenu provides a practical alternative. Using the flexible tool, you can quickly and easily bring up a selection menu on the screen, with relatively simple syntax and with the use of very little code. Many options allow for a versatile design. ■■■

INFO

[1] smenu project page: *https://github.com/p-gen/smenu*

[2] smenu wiki: *https://github.com/p-gen/smenu/wiki*

LISTING 9: pdfsearch.sh

```
#!/bin/bash
# pdfsearch.sh
while true; do
  clear
  # Enter search term
  echo "browse PDF archive "
  echo "----------------------"
  read -p "enter search term (cancel: Enter or Ctrl+C): " sube
  if [ -z $sube ]; then
    exit
  fi
  # Evaluation list structure, case sensitivity is ignored.
  # only the number of hits is issued
  for i in $(pdfgrep -H -c -i "$sube" *.pdf); do
    number of hits = $(echo $i | cut-d\: -f2)
    if [ $number of hits-gt 0 ]; then
      hitlist=$(echo $hitlist $i)
    fi
  done
  # if hit is list empty, new loop run
  if [ -z "$hit list" ]; then
    echo "no hits"
    sleep 3
    continue
  fi
 # here comes smenu
  while true; do
    file=$(echo $hitlist | tr "\ " "\n" | smenu -d -n20 -c -m "select file for
           display (q to cancel):" | cut -d\: -f1)
    # If cancel with [q], delete hit list and
    # new loop run
    if [ -z $file ]; then
      unset hit list
      break
    fi
    # Evince and Qpdfview support the transfer
    # a search term when launching on Shell
    evince --find "$sube" $file
    # continue or finished?
    Continue= $(echo "select done" | smenu -d -m "continue with" -s /A)
    if [ "$continue" = "done" ]; then
      break
    fi
  done
done
```

HDF5 for efficient I/O

Fast Containers

HDF5 is a flexible, self-describing, and portable hierarchical filesystem supported by a number of languages and tools, with the ability to run processes in parallel. *By Jeff Layton*

Input/output operations are a very important part of many applications, sometimes involving a huge amount of data and a large number of reads and writes. Therefore, applications can use a very significant portion of their total run time to perform I/O, which becomes critical in Big Data, machine learning, and high-performance computing (HPC).

In a previous article [1], I discussed options for improving I/O performance, focusing on parallel I/O. One of the options mentioned was to use a high-level library to perform the I/O. A great example of such a library is the Hierarchical Data Format (HDF) [2], a standard library used primarily for scientific computing.

In this article, I introduce HDF5 and focus on the concepts and its strengths in performing I/O; then, I look at some simple Python and Fortran code examples, before ending with an example of parallel I/O with HDF5 and Fortran.

What Is HDF5?

HDF5 is a freely available file format standard and set of tools for storing and organizing large amounts of data. It uses a filesystem-like data format familiar to anyone who has used a modern operating system – thus, the "hierarchical" portion of the name. You can store almost any data you want in an HDF5 file, including user-defined data types; integer, floating point, and string data; and binary data such as images, PDFs, and Excel spreadsheets. Files written in the HDF5 format are portable across operating systems and hardware (little endian and big endian).

HDF5 also allows metadata (attributes) to be associated with virtually any object in the data file. Metadata is the key to useful data files, and attributes make HDF5 files self-describing (e.g., like XML).

An example of how you could structure data within an HDF5 file is described in an online tutorial [3] that shows how to use HDFView [4] (Figure 1) to view an HDF5 file of hyperspectral remote sensing data. Notice how the temperature data falls under a hierarchy of directories. At the bottom of the viewer, the metadata associated with that data displays when you click on a data value (the temperature).

A number of tools and libraries use HDF5 in your favorite language. For example, C, C++, Fortran, and Java are officially supported with HDF5 tools, but some third-party bindings (i.e., outside the official distribution) are also available for Python, Matlab, Octave, Scilab, Mathematica, R, Julia, Perl, Lua, Node.js, Erlang, Haskell, and others [5]-[9].

The HDF5 format can also accommodate data in row-major (C/C++, Mathematica) or column-major (Fortran, Matlab, Octave, Scilab, R, Julia, NumPy) order. The libraries from the HDF5 group are capable of compressing data within the file and even "chunking" the data into sub-blocks for storage. Chunking can result in faster access times for subsets of the data. Moreover, you can create lots of metadata to associate with data inside an HDF5 file.

Data in an HDF5 file can be accessed randomly, as in a database, so you don't have to read the entire file to access the data you want (unlike XML).

One of the most interesting capabilities of HDF5 is parallel I/O [10]. However, you might have to build HDF5 with an MPI library that supports MPI-IO, a low-level interface for carrying out parallel I/O that gives you a great deal of flexibility but also requires a fair amount of coding. Parallel HDF5 is built on top of MPI-IO to remove most of the pain of parallel I/O.

Storing Data in HDF5

HDF5 comprises a *file format* for storing HDF data, a *data model* for organizing and accessing HDF5 data, and the *software*, comprising libraries, language interfaces, and tools.

The file format is defined and published by the HDF Group. The HDF data model is fairly straightforward. Fundamentally, an HDF5 file is a container that holds data objects. Currently, eight objects either store data or help organize it:

- dataset
- group
- attribute
- file
- link
- datatype
- dataspace
- property list

Of these objects, datasets (multidimensional homogeneous arrays) and groups (container structures that can hold datasets and other groups) hold data, whereas the other objects are used for data organization.

Groups and Datasets

HDF5 groups are key to organizing data and are very similar to directories in a

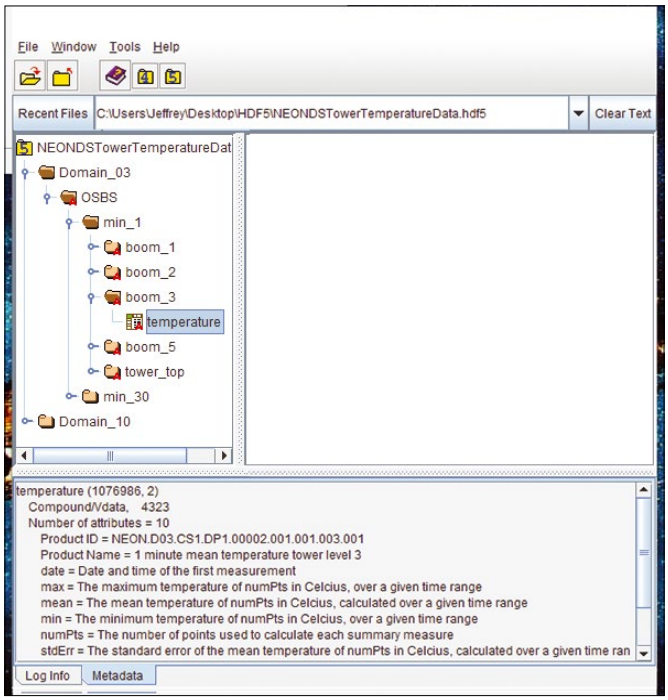


Figure 1: Example of the HDF5 data hierarchy in HDFView.

filesystem. Just like directories, you can organize data hierarchically so that the data layout is much easier to understand. With attributes (metadata), you can make groups even more useful than directories by adding descriptions.

HDF5 datasets are very similar to files in a filesystem. They hold data in the form of multidimensional arrays of elements. Data can be almost anything (e.g., images, tables, graphics, documents). As with groups, they also have metadata.

Every HDF5 file contains a "root" group that can contain other groups or datasets (files) or can be linked to other dataset objects in other portions of the HDF5 file. A root group is something like the root directory in a filesystem. In HDF5, it is referred to as /. If you write `/foo`, then `foo` is a member of the root group (it could be another group or a dataset or a link to other files) and looks something like a "path" in a filesystem.

Figure 2 show a theoretical HDF5 file layout. The root group (/) has three subgroups: A, B, and C. Groups `/A/temp` and `/C/temp` point to different datasets. However, Figure 2 also shows that datasets can be shared (like a symbolic link in filesystems): `/A/k` and `/B/m` point to the same object (a dataset).

Groups and datasets are the two most fundamental object types in HDF5. If you can remember them and use them, then you can start writing and reading HDF5 files in your applications.

Dataset Details

A dataset object comprises data values and the metadata that describes it. A dataset has two fundamental parts: a header and a data array. The header contains information about the data array portion of the dataset and the associated metadata. Typical header information includes the name of the object, dimensionality, number type, information about how the data is stored on disk, and other information that HDF5 can use to speed up data access or improve data integrity.

The header has four essential classes of information: name, datatype, dataspace, and storage layout. A name in HDF5 is just a set of ASCII characters, but you should use a name that is meaningful to the dataset. A datatype in HDF5 describes the individual data elements in a dataset and comprises two categories or types: atomic and compound. Datatypes can be quite complicated to define, so I focus only on the basics. Some predefined datatypes [11] can be used for the data you typically might encounter.

The atomic datatype includes integers, floating-point numbers, and strings. Each datatype has a set of properties. For example the integer datatype properties are size, order (endianness), and sign (signed/unsigned). The float datatype properties are size, location of the exponent and mantissa, and location of the sign bit.

Compound datatypes refer to collections of several datatypes that are presented as a single unit. In C, this is similar to a `struct`. The various parts of a compound datatype are called members and may be of any datatype, including another compound datatype. One of the fancy features of HDF5 is that it is possible to read members from a compound datatype without reading the whole type.

The layout of a dataset's data elements can consist of non-elements (`NULL`), a single element (a scalar), or a simple array. The dataspace can be fixed or unlimited, which allows it to be extensible (i.e., it can grow larger).

Dataspace properties include rank (number of dimensions), size (dimensions), and maximum size (size to which an array may grow). The dimensionality (rank) of the dataspace is fixed when the array is created and can include a maximum size that each dimension can grow during the lifetime of the dataspace.

If you are not sure what dimensions your dataspace might become, you can always use the HDF5 predefined variable `H5P_UNLIMITED`.

Attributes

One of the fundamental objects in HDF5 is an attribute, which is how you store metadata inside an HDF5 file. Optionally, attributes can be associated with other HDF5 objects, such as groups, datasets, or named datatypes if they are not independent objects. As such, attributes are accessed by opening the object to which they are attached.

As the user, you define the attributes (make it meaningful), and you can delete them and overwrite them as you see fit.

Attributes have two parts. The first is a name, and the second is a value. Classically, the value is a string that describes the data to which it is attached. They can be *extremely* useful in a data file. Using

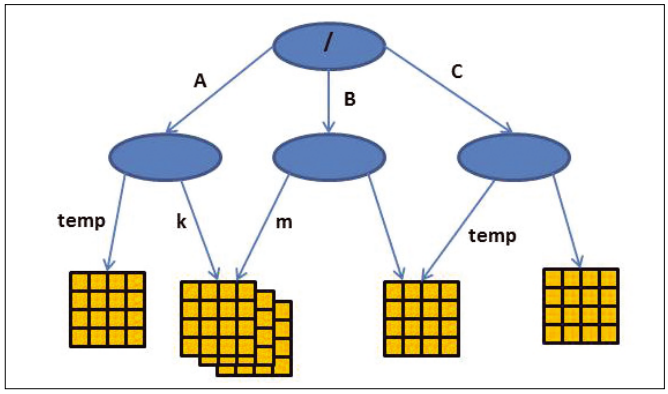


Figure 2: Theoretical HDF5 layout.

attributes, you can describe the data, including information such as when the data was collected, who collected it, what applications or sensors were used in its creation, a description (with as much information as you can include), and so on. A lack of useful metadata is one of the biggest problems in HPC data today, and attributes can be used to help alleviate the problem. You just have to use them.

HDF5 Basics

In this section, I want to present a quick introduction to HDF5 through some simple code examples. The goal is not to dive deep into HDF5 but to illustrate the basics in practice. I'll start with Python because it is a widely used language, and the HDF5 Python library *h5py* [12] is very easy to use and very easy to understand.

h5py

The dominant Python interface to HDF5 is h5py. It is included with many Python and most Linux distributions. For the examples here, I use the Anaconda Python [13] distribution for Python 2.7.

The examples I use here are fairly simple and are derived from the Quick Start page [14] on the h5py website. The first example simply illustrates a few concepts, such as opening an HDF5 file for writing, creating datasets, and creating groups. The simple Python script in Listing 1 incorporates these concepts.

The first `h5py` command in line 10 opens a file for writing. If the file exists, it is overwritten; if it doesn't exist, it is created. Remember that HDF5 is really a *container* for data objects. When you create a file, the library creates a number of defaults, such as the root group (`/`), so the file is non-zero in size, even if no data or attributes are written into it.

After the file is opened and created, line 11 creates a dataset (`mydataset`) with 100 integers. At this point, only the object for the dataset is created in the file (dataspace). Line 12 puts data into the data object using NumPy [15]. When you put data into the object or modify existing data, the *h5py* library takes care of updating the HDF5 file.

Recall that in Python almost everything is an object that has properties. Listing 1 prints the properties of the HDF5 file (line 16) and the first dataset (lines 13-15). Because HDF5 is object based, it fits well with the object nature of Python.

Line 17 creates a subgroup to the root group (`subgroup`); then, in line 18, a method of the group object creates a new dataset that resides in this subgroup using a *float* data type that starts with 50 elements. Line 20 creates a new dataset in a new subgroup named `subgroup2`. H5py will create the subgroup automatically if it doesn't exist. The output from this example Python script is show in Listing 2.

Notice the size of the integers. The NumPy integer type represents integers with 32 bits (`int32`).

Another short Python script reads the HDF5 file and outputs some of the attributes. This can be done fairly easily using the h5py function `visit` (Listing 3), which walks the HDF5 file recursively, so you can discover the objects in the file, including groups and datasets. With this function, you can print the "names" of the objects. The output from the script is shown in Listing 4.

You can find more information in the HDF5 documentation [16], and the Quick Start guide [14] has more examples of accessing HDF5 files from Python.

Fortran and HDF5

H5py is a very Python-centric library allowing HDF5 to be used in a very flexible manner, but compiled languages are a little different, so I also want to illustrate how to use HDF5 with a compiled language – in particular, Fortran. Using HDF5 with compiled languages is not quite as easy as with Python, but it is still not difficult. The developers of HDF5 have created a number of functions and

LISTING 1: test.py

```
01 #!/home/laytonjb/anaconda2/bin/python
02
03 import h5py
04 import numpy as np
05 # ====================
06 # Main Python section
07 # ====================
08 #
09  if __name__ == '__main__':
10  f = h5py.File("mytestfile.hdf5", "w")
11  dset = f.create_dataset("mydataset", (100,), dtype='i')
12  dset[...] = np.arange(100)
13  print "dset.shape = ",dset.shape
14  print "dset.dtype = ",dset.dtype
15  print "dset.name = ",dset.name
16  print "f.name = ",f.name
17  grp = f.create_group("subgroup");
18  dset2 = grp.create_dataset("another_dataset", (50,),
    dtype='f');
19  print "dset2.name = ",dset2.name
20  dset3 = f.create_dataset('subgroup2/dataset_three',
                             (10,), dtype='i')
21  print "dset3.name = ",dset3.name
22  # end if
```

LISTING 2: Output of test.py

```
$ ./test.py
dset.hsape = (100,)
dset.dtype = int32
dset.name = /mydataset
f.name = /
dset2.name = /subgroup/another_dataset
dset3.name = /subgroup2/dataset_three
```

LISTING 3: test2.py

```
01 #!/home/laytonjb/anaconda2/bin/python
02
03 import h5py
04 import numpy as np
05 def printname(name):
06   print name
07 # ====================
08  # Main Python section
09  # ====================
10  if __name__ == '__main__':
11  f = h5py.File("mytestfile.hdf5", "r")
12  for name in f:
13    print name
14  # end for
15  f.visit(printname);
16  # end if
```

LISTING 4: Output of test2.py

```
$ ./test2.py
mydataset
subgroup
subgroup2
mydataset
subgroup
subgroup/another_dataset
subgroup2
subgroup2/dataset_three
```

subroutines to be used for manipulating data and objects in an HDF5 file that make programming straightforward.

For this example, I use a CentOS 7.3 system with the default Fortran compiler (gfortran) and the HDF5 library that is part of the distribution. It's not difficult to build a Fortran executable with gfortran and the HDF5 library. The generic command line below illustrates how,

```
$ gfortran code.f90 ➥
  -fintrinsic-modules-path ➥
  /usr/lib64/gfortran/modules ➥
  -lhdf5_fortran -o exe
```

where `code.f90` is the source file and `exe` is the resultant binary.

The HDF Group has provided some sample Fortran 90 code to get started, as well as more complex examples [17]. Listing 5 shows a Fortran 90 version of the first sample Python code from these examples.

Notice that the code uses some predefined HDF5 variables that are necessary to use the library. Also note that this isn't "good" coding, in that the error variable is not checked when returning from a subroutine call. This code is just an example, and I wanted to keep it short in the interest of space.

The basic process of using HDF5 in Fortran is pretty logical. To begin, you initialize or enable the Fortran interface (line 57); then, you open a file (line 59) and start creating objects.

The first object to create is a dataset in the root (/) group, but first, you have to create the dataspace (line 62) then the dataset (line 64). Line 66 writes the data to the dataset. To reverse the process, first close the dataset (line 68) and then the dataspace (line 70).

The general approach for writing a dataset to an HDF5 file using Fortran is (1) open a dataspace, (2) open a dataset within the dataspace, (3) write the data to the dataset, (4) close the dataset, and (5) close the dataspace. You could easily write a function in Fortran 90 for all these steps if you desired.

Interestingly, that when using these subroutines, you have to use the full path to the group in which you are going to write the dataset. With the h5py Python module, you can write to a group by using the method associated with the specific group.

LISTING 5: Sample Fortran 90 Code

```
001 PROGRAM TEST
002
003  USE HDF5 ! This module contains all necessary HDF5 modules
004
005  IMPLICIT NONE
006
007  ! Names (file and HDF5 objects)
008  CHARACTER(LEN=15), PARAMETER :: filename = "mytestfile.hdf5" ! File name
009  CHARACTER(LEN=9), PARAMETER :: dsetname1 = "mydataset" ! Dataset name
010  CHARACTER(LEN=8), PARAMETER :: groupname = "subgroup" ! Sub-Group 1 name
011  CHARACTER(LEN=9), PARAMETER :: groupname3 = "subgroup2" ! Sub-Group 3 name
012  ! Dataset 2 name
013  CHARACTER(LEN=24), PARAMETER :: dsetname2 = "subgroup/another_dataset"
014  ! Dataset 3 name
015  CHARACTER(LEN=23), PARAMETER :: dsetname3 = "subgroup2/dataset_three"
016
017  ! Identifiers
018  INTEGER(HID_T) :: file_id   ! File identifier
019  INTEGER(HID_T) :: group_id  ! Group identifier
020  INTEGER(HID_T) :: group3_id   ! Group 3 identifier
021  INTEGER(HID_T) :: dset1_id  ! Dataset 1 identifier
022  INTEGER(HID_T) :: dset2_id  ! Dataset 2 identifier
023  INTEGER(HID_T) :: dset3_id  ! Dataset 3 identifier
024  INTEGER(HID_T) :: dspace1_id  ! Dataspace 1 identifier
025  INTEGER(HID_T) :: dspace2_id  ! Dataspace 2 identifier
026  INTEGER(HID_T) :: dspace3_id  ! Dataspace 3 identifier
027
028  ! Integer array
029  INTEGER :: rank               ! Dataset rank
030  INTEGER(HSIZE_T), DIMENSION(1) :: dims1 = (/100/) ! Dataset dimensions
031  INTEGER(HSIZE_T), DIMENSION(1) :: data_dims1
032  INTEGER, DIMENSION(100) :: dset_data1 ! Data buffers
033
034  ! FP array
035  INTEGER(HSIZE_T), DIMENSION(1) :: dims2 = (/50/)
036  INTEGER(HSIZE_T), DIMENSION(1) :: data_dims2
037  REAL, DIMENSION(50) :: dset_data2
038
039  ! Integer array for dataset_three
040  INTEGER(HSIZE_T), DIMENSION(1) :: dims3 = (/10/) ! Dataset dimensions
041  INTEGER(HSIZE_T), DIMENSION(1) :: data_dims3   ! Dataset rank
042  INTEGER, DIMENSION(10) :: dset_data3
043
044  ! Misc variables (e.g. loop counters)
045  INTEGER :: error ! Error flag
046  INTEGER :: i,j
047  ! ================================================================
048
049  ! Initialize the dset_data array
050  data_dims1(1) = 100
051  rank = 1
052  DO i = 1, 100
053    dset_data1(i) = i
054  END DO
055
056  ! Initialize Fortran interface
057  CALL h5open_f(error)
058  ! Create a new file
059  CALL h5fcreate_f(filename, H5F_ACC_TRUNC_F, file_id, error)
060
061  ! Create dataspace 1 (the dataset is next) "dspace_id" is returned
```

LISTING 5: Sample Fortran 90 Code (continued)

```
062  CALL h5screate_simple_f(rank, dims1, dspace1_id, error)
063  ! Create dataset 1 with default properties "dset_id" is returned
064  CALL h5dcreate_f(file_id, dsetname1, H5T_NATIVE_INTEGER, dspace1_id, &
                      dset1_id, error)
065  ! Write dataset 1
066  CALL h5dwrite_f(dset1_id, H5T_NATIVE_INTEGER, dset_data1, data_dims1, &
                     error)
067  ! Close access to dataset 1
068  CALL h5dclose_f(dset1_id, error)
069  ! Close access to data space 1
070  CALL h5sclose_f(dspace1_id, error)
071
072  ! Create a group in the HDF5 file
073  CALL h5gcreate_f(file_id, groupname, group_id, error)
074  ! Close the group
075  CALL h5gclose_f(group_id, error)
076
077  ! Create dataspace 2 (the dataset is next)
078  data_dims2(1) = 50
079  DO i = 1, 50
080    dset_data2(i) = 1.0
081  END DO
082  ! Create dataspace 2
083  CALL h5screate_simple_f(rank, dims2, dspace2_id, error)
084  ! Create dataset 2 with default properties
085  CALL h5dcreate_f(file_id, dsetname2, H5T_NATIVE_REAL, dspace2_id, & dset2_id,
                      error)
086  ! Write dataset 2
087  CALL h5dwrite_f(dset2_id, H5T_NATIVE_REAL, dset_data2, data_dims2, & error)
088  ! Close access to dataset 2
089  CALL h5dclose_f(dset2_id, error)
090  ! Close access to data space 2
091  CALL h5sclose_f(dspace2_id, error)
092
093  ! Create a group in the HDF5 file
094  CALL h5gcreate_f(file_id, groupname3, group3_id, error)
095  ! Close the group
096   CALL h5gclose_f(group3_id, error)
097
098   ! Create dataspace 3
099   data_dims3(1) = 10
100   DO i = 1, 10
101   dset_data3(i) = i + 3
102   END DO
103   ! Create dataspace 3
104   CALL h5screate_simple_f(rank, dims3, dspace3_id, error)
105   ! Create dataset 3 with default properties
106   CALL h5dcreate_f(file_id, dsetname3, H5T_NATIVE_INTEGER, & dspace3_id,
                       dset3_id, error)
107   ! Write dataset 3
108   CALL h5dwrite_f(dset3_id, H5T_NATIVE_INTEGER, dset_data3, data_dims3, &
                      error)
109   ! Close access to dataset 3
110   CALL h5dclose_f(dset3_id, error)
111   ! Close access to data space 3
112   CALL h5sclose_f(dspace3_id, error)
113
114   ! Close the file
115   CALL h5fclose_f(file_id, error)
116   ! Close FORTRAN interface
117   CALL h5close_f(error)
118  END PROGRAM TEST
```

After running the Fortran code, which has no output, run the `test2.py` script from the Python section against the Fortran output:

```
$ ./test2.py
mydataset
subgroup
subgroup2
mydataset
subgroup
subgroup/another_dataset
subgroup2
subgroup2/dataset_three
```

If you compare this with the output from the Python code, you will see that they are the same.

HDF5 and Parallel I/O

A key attribute of HDF5 is Parallel HDF5, which is included in the source code and available through a configure option. Several processes, either on the same node or on different nodes, can write to the same file at the same time. This capability can reduce the time an application spends on I/O, because all of the processes are performing a portion of the I/O.

Amdahl's law says that your application will only go as fast as its serial portion. As an application is run over more processors, run time decreases. As the number of processors, *N*, goes to infinity, the wall clock time approaches a constant [1]. In general, this constant can be thought of as the "serial time" of the application (i.e., the amount of time the application needs to run regardless of the number of processes used).

Many times I/O is a significant portion of this serial time. If the I/O could be parallelized, the application could scale even further, which could improve application performance and the ability to run larger problems. The great thing about HDF5 is that, behind the scenes, it uses MPI-IO [18]. A great deal of time has been spent designing and tuning MPI-IO for various filesystems and applications, which has resulted in very good parallel I/O performance from MPI applications.

Here, I explore using HDF5 for parallel I/O. The intent is to reduce the serial portion of I/O to improve scalability and performance. Before jumping knee-deep into HDF5 parallel I/O, I'll explore MPI processes writing to separate datasets in the same HDF5 file.

Single Thread Process I/O

Among the previously mentioned ways that applications can perform I/O, Listing 6 shows how MPI tasks can each write a specific set of data to the same file. The example is fairly simple and illustrates the basics parallel I/O work with HDF5. The example is written in Fortran for two reasons: (1) It's easy to read, and (2) it's the scientific language of scientists.

For each MPI process to write an individual dataset to the same HDF5 file, the rank 0 MPI process initializes the HDF5 file and the dataspace with the appropriate properties and then closes the file. Next, each MPI process reopens the file and writes its data to and closes the HDF5 file. For the sake of brevity, the error codes returned from functions are not checked.

The following tasks are performed by the rank 0 process, which defines the HDF5 file and its attributes: Line 53 initializes the HDF5 file and properties, line 56 creates the HDF5 file, line 59 creates the dataspace, line 62 creates the data properties, line 65 is the call required when each process writes to a single file, lines 67-73 loop over all processes and create the dataset name and dataset for each process, line 76 closes the dataspace, line 79 closes the properties list, and line 82 closes the HDF5 file.

The following tasks are then performed by all of the processes: Line 89 creates a new properties list, line 90 sets the MPI-IO property, line 93 opens the file, line 96 closes the properties, line 103 opens the dataset, line 107 sets the MPI-IO property, lines 110-111 create a pointer to and write data to the dataset (a specific dataset for each process), and lines 114-125 close everything and finish up.

A quick run of the example with two processes creates the file `test1.hdf5`. Listing 7 shows the content of this file using the HDF5 `h5dump` tool. Both datasets are in the file, so each MPI process wrote its respective dataset.

Parallel I/O and HDF5

The previous discussion demonstrates a good way to get started with parallel I/O and HDF5, but it has some limitations. For example, each MPI process writes its portion of an array to a different dataset in the HDF5 file, so if you want to restart the application from the beginning, you would have to use the same number of MPI processes as used originally. A better

LISTING 6: Basic Parallel I/O with HDF5

```
001       PROGRAM TEST1
002
003       USE iso_c_binding
004       USE HDF5 ! This module contains all necessary modules
005       IMPLICIT NONE
006
007       INCLUDE 'mpif.h'
008
009      INTEGER :: NUMTASKS, MPI_RANK, LEN, IERR , MPIERROR, INFO, C_LEN
010      INTEGER :: I, J
011      INTEGER(HID_T) :: PLIST_ID      ! Property list identifier
012      INTEGER(HID_T) :: DCPL
013      INTEGER(HID_T) :: FILE_ID       ! File identifier
014      INTEGER(HID_T) :: DSET_ID       ! Dataset identifier
015      INTEGER(HID_T) :: FILESPACE     ! Data space identifier in file
016      INTEGER(HSIZE_T), DIMENSION(2) :: DIMSF = (/5,8/) ! Dataset dimensions
017      INTEGER, ALLOCATABLE, TARGET :: DATA(:,:)   ! Data to write
018      INTEGER :: RANK = 2             ! Dataset rank
019
020      CHARACTER(MPI_MAX_PROCESSOR_NAME) HOSTNAME
021      CHARACTER(LEN=100) :: FILENAME  ! File name
022      CHARACTER(LEN=3) :: C           ! Dataset name for specific rank
023      CHARACTER(LEN=10) :: DATASET_NAME
024
025      TYPE(C_PTR) :: F_PTR
026    ! ----------------------------------------------------------------------
027      ! Initialization
028      INFO = MPI_INFO_NULL
029
030      FILENAME = "test1.hdf5"   ! Define File name
031
032      ! Initialize MPI
033      CALL MPI_INIT(IERR)
034
035      ! Get number of tasks
036      CALL MPI_COMM_SIZE(MPI_COMM_WORLD, NUMTASKS, MPIERROR)
037
038      ! Get my rank
039      CALL MPI_COMM_RANK(MPI_COMM_WORLD, MPI_RANK, MPIERROR)
040
041      ! Initialize FORTRAN interface
042      CALL H5OPEN_F(IERR)
043
044      ! Initialize data buffer with trivial data
045      ALLOCATE ( DATA(DIMSF(1),DIMSF(2)) )
046      DO I = 1, DIMSF(2)
047         DO J = 1, DIMSF(1)
048            DATA(J,I) = J - 1 + (I-1)*DIMSF(1)*(MPI_RANK+1)
049         ENDDO
050      ENDDO
051
052      ! Have the rank 0 process create the HDF5 data layout
053      IF (MPI_RANK == 0) THEN
054
055         ! Create the HDF5 file
056         CALL h5fcreate_f(FILENAME, H5F_ACC_TRUNC_F, FILE_ID, IERR)
057
058         ! Create the dataspace for the  dataset
059         CALL h5screate_simple_f(RANK, DIMSF, FILESPACE, IERR)
060
061         ! Create the properties for the data
062         CALL H5Pcreate_f(H5P_DATASET_CREATE_F, DCPL, IERR)
063
064         ! This is required for this data pattern
065         CALL H5Pset_alloc_time_f(DCPL, H5D_ALLOC_TIME_EARLY_F, IERR)
```

LISTING 6: Basic Parallel I/O with HDF5 (continued)

```
066         ! Create each dataset with default properties
067         DO I=1,NUMTASKS
068            WRITE(C,"(i0)") i
069            dataset_name = "dataset" // TRIM(C)
070            WRITE(*,*) 'C = ',C,' data_name = ', DATASET_NAME
071            CALL h5dcreate_f(FILE_ID, DATASET_NAME, H5T_NATIVE_INTEGER, &
                                FILESPACE, DSET_ID, IERR, DCPL_ID=DCPL)
072            CALL h5dclose_f(DSET_ID, IERR)
073         END DO
074
075         ! Close the dataspace
076         CALL h5sclose_f(FILESPACE, IERR)
077
078         ! Close the properties
079         CALL h5pclose_f(DCPL, IERR)
080
081         ! Close the file
082         CALL h5fclose_f(FILE_ID, IERR)
083      END IF
084
085      ! Use an MPI barrier to make sure every thing is synched
086      CALL mpi_barrier(MPI_COMM_WORLD, IERR)
087
088      ! Setup file access property list with parallel i/o access.
089      CALL h5pcreate_f(H5P_FILE_ACCESS_F, PLIST_ID, IERR)
090      CALL h5pset_fapl_mpio_f(PLIST_ID, MPI_COMM_WORLD, INFO, IERR)
091
092      ! Open HDF5 to write data
093      CALL h5fopen_f(FILENAME, H5F_ACC_RDWR_F, FILE_ID, IERR, PLIST_ID)
094
095      ! Close the property list
096      CALL h5pclose_f(PLIST_ID, IERR)
097
098     ! Create the dataset names based on MPI rank
099     WRITE(C,"(i0)") MPI_RANK + 1
100     DATASET_NAME = "dataset" // TRIM(C)
101
102     ! Open dataset (each rank opens its own dataset)
103     CALL h5dopen_f(FILE_ID, DATASET_NAME, DSET_ID, IERR)
104
105     ! Open properties and definee MPIO model (independent)
106     CALL h5pcreate_f(H5P_DATASET_XFER_F, PLIST_ID, IERR)
107     CALL h5pset_dxpl_mpio_f(PLIST_ID, H5FD_MPIO_INDEPENDENT_F, IERR)
108
108     ! Write data to dataset (use Fortran pointer)
110     F_PTR = c_loc(DATA(1,1))
111     CALL h5dwrite_f(DSET_ID, H5T_NATIVE_INTEGER, F_PTR, IERR)
112
113     ! Close the property list, the data set and the file
114     CALL h5pclose_f(PLIST_ID, IERR)
115     CALL h5dclose_f(DSET_ID,IERR)
116     CALL h5fclose_f(FILE_ID, IERR)
117
118     ! Close fortran interface
119     CALL h5close_f(IERR)
120
121     ! Deallocate Fortran data
122     DEALLOCATE(DATA)
123
124     ! Finalize MPI
125     CALL mpi_finalize(IERR)
126
127     END
```

solution would be for each MPI process to read and write its data to the same dataset, which allows the number of MPI processes to change without worrying about how the data is written in the HDF5 file.

The best way to achieve this is to use hyperslabs, or portions of datasets. It can be a contiguous section of a dataset, such as a block, a regular pattern of individual data values, or a block within a dataset. The 2x2 blocks in Table 1 are separated by a column and a row. Each 2x2 block or each row or column of a 2x2 block can be a hyperslab, depending on the design.

To describe a hyperslab completely, you need four parameters:

- `start` – a starting location
- `stride` – the number of elements that separate each element or block to be selected
- `count` – the number of elements or blocks to select along each dimension
- `block` – the size of the block selected from the dataspace

LISTING 7: Content of HDF5 File

```
$ h5dump test1.hdf5
HDF5 "test1.hdf5" {
GROUP "/" {
   DATASET "dataset1" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SIMPLE { ( 8, 5 ) / ( 8, 5 ) }
      DATA {
      (0,0): 0, 1, 2, 3, 4,
      (1,0): 5, 6, 7, 8, 9,
      (2,0): 10, 11, 12, 13, 14,
      (3,0): 15, 16, 17, 18, 19,
      (4,0): 20, 21, 22, 23, 24,
      (5,0): 25, 26, 27, 28, 29,
      (6,0): 30, 31, 32, 33, 34,
      (7,0): 35, 36, 37, 38, 39
      }
   }
   DATASET "dataset2" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SIMPLE { ( 8, 5 ) / ( 8, 5 ) }
      DATA {
      (0,0): 0, 1, 2, 3, 4,
      (1,0): 10, 11, 12, 13, 14,
      (2,0): 20, 21, 22, 23, 24,
      (3,0): 30, 31, 32, 33, 34,
      (4,0): 40, 41, 42, 43, 44,
      (5,0): 50, 51, 52, 53, 54,
      (6,0): 60, 61, 62, 63, 64,
      (7,0): 70, 71, 72, 73, 74
      }
   }
}
}
```

Each of the parameters is an array with a rank that is the same as the dataspace.

The HDF group has created several parallel examples. The simplest, ph5example [19], illustrates how to get started. In the code, the HDF5 calls are accomplished with MPI processes (all with the same data). The program in Listing 6, in which each MPI process wrote its own dataset, started with this example.

A number of HDF5 examples use hyperslab concepts for parallel I/O. Each MPI process writes a part of the data to the common dataset. The main page for these tutorials [20] has four hyperslab examples: writing datasets by contiguous hyperslab, by regularly spaced data, by pattern, and by chunk. Here, I go through the last example, which writes data to a common dataset by chunks.

In the Fortran example [21] (see an excerpt of the code in Listing 8), the number of processes is fixed at four to illustrate how each MPI process writes to a common dataset (Figure 3). The dataset is 4x8 (rows by columns), and each chunk is 2x4 (rows by columns).

Recall that four parameters describe a hyperslab: `start`, `stride`, `count`, and `block`. The `start` parameter is also referred to as `offset` in the example (and other) code.

The dimensions of each chunk are given by the array `chunk_dims`. The first element is the width of the chunk (number of columns), and the second element is the height of the chunk (number of rows). For all MPI processes, `block(1) = chunk_dims(1)` and `block(2) = chunk_dims(2)`.

Because the data from each process is a chunk, the `stride` array for each chunk is 1 (i.e., `stride(1) = 1`, `stride(2) = 1`, or a contiguous chunk). Some of the other examples have arrays for which `stride` is not 1. The array `count` for each chunk is also 1; that is, each MPI process is only writing a single chunk (`count(1) = 1`, and `count(2) = 1`).

What differs is the `offset` or `start` of each chunk. For clarity, the `rank = 0` process writes the chunk in the bottom left portion of the dataset, so both elements of its offset arrays are `0`. The `rank = 1` process writes the bottom right chunk of the dataset. The `offset` array is `offset(1) = chunk_dims(1)` and `offset(2) = 0`.

MPI process 2 writes the top left-hand chunk. Its offset array is `offset(1) = 0` and

TABLE 1: Hyperslab Pattern

|	|	|	|	|	|	|
|	X	X	|	X	X	|
|	X	X	|	X	X	|
|	|	|	|	|	|	|
|	X	X	|	X	X	|
|	X	X	|	X	X	|
|	|	|	|	|	|	|

LISTING 8: Write by Chunk (Excerpt [21])

```
01 !
02 ! Number of processes is assumed to be 4
03 !
04       PROGRAM DATASET_BY_CHUNK
05       USE HDF5 ! This module contains all necessary modules
06 !     USE MPI
07 [...]
08       INTEGER(HSIZE_T), DIMENSION(2) :: chunk_dims = (/2,4/) ! Chunks dimensions
09       INTEGER(HSIZE_T),  DIMENSION(2) :: count
10       INTEGER(HSSIZE_T), DIMENSION(2) :: offset
11       INTEGER(HSIZE_T),  DIMENSION(2) :: stride
12       INTEGER(HSIZE_T),  DIMENSION(2) :: block
13       INTEGER, ALLOCATABLE :: data (:,:)  ! Data to write
14       INTEGER :: rank = 2 ! Dataset rank
15 [...]
16       !
17       ! Each process defines dataset in memory and writes it to the hyperslab in
           the file.
18       !
19       stride(1) = 1
20       stride(2) = 1
21       count(1) =  1
22       count(2) =  1
23       block(1) = chunk_dims(1)
24       block(2) = chunk_dims(2)
25       if (mpi_rank .EQ. 0) then
26           offset(1) = 0
27           offset(2) = 0
28       endif
29       if (mpi_rank .EQ. 1) then
30           offset(1) = chunk_dims(1)
31           offset(2) = 0
32       endif
33       if (mpi_rank .EQ. 2) then
34           offset(1) = 0
35           offset(2) = chunk_dims(2)
36       endif
37       if (mpi_rank .EQ. 3) then
38           offset(1) = chunk_dims(1)
39           offset(2) = chunk_dims(2)
40       endif
41 [...]
42       ! Select hyperslab in the file.
43       !
44       CALL h5dget_space_f(dset_id, filespace, error)
45       CALL h5sselect_hyperslab_f (filespace, H5S_SELECT_SET_F, offset, count,
                                     error, & stride, block)
46 [...]
47       ! Write the dataset collectively.
48       !
49       CALL h5dwrite_f(dset_id, H5T_NATIVE_INTEGER, data, dimsfi, error, &
         file_space_id = filespace, mem_space_id = memspace, xfer_prp = plist_id)
50       !
51       ! Write the dataset independently.
52       !
53 !     CALL h5dwrite_f(dset_id, H5T_NATIVE_INTEGER, data, dimsfi,error, &
         file_space_id = filespace, mem_space_id = memspace)
54 [...]
```

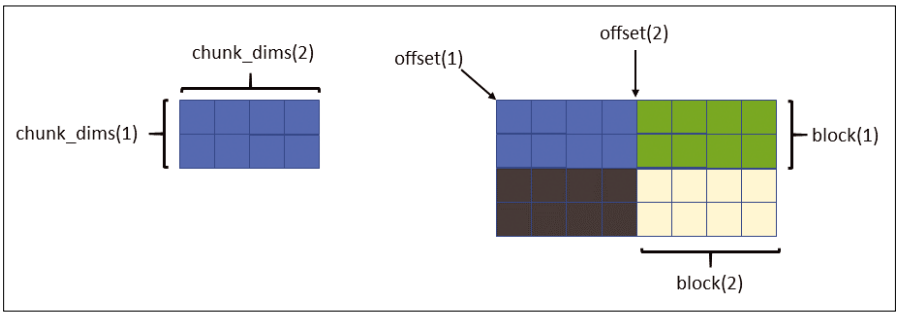


Figure 3: Data layout for contiguous "chunk" approach.

`offset(2) = chunk_dims(2)`. Finally, MPI process 3 writes the top right-hand chunk. Its offset array is `offset(1) = chunk_dims(1)` and `offset(2) = chunk_dims(2)`.

For each chunk, the data is an array of integers that correspond to rank (MPI process + 1). The data in the rank 0 process has values of 1, the data in the rank 1 process has values of 2, and so on.

The process for writing hyperslab data to a single dataset is a little different from the first example, but for the most part, it is the same. When creating the dataspace, you have to call some extra functions to configure the hyperslabs.

To write the hyperslab to the dataset, each MPI process calls `h5sselect_hyperslab_f` to select the appropriate hyperslab using the four parameters mentioned before. All MPI processes then do a collective write, which puts the information into the header of the dataset. The second call writes the dataset to the location defined by the hyperslab for that process.

The code produces `sds_chnk.h5`, which contains the data shown in Listing 9.

Because `h5dump` is written in C, the data writes to stdout in row-major [22] format (the opposite of column-major Fortran). With some transposing, you can see that the dataset is as expected.

It takes some work to write hyperslabs to the same dataset using MPI-IO. Read through the other examples in the parallel topics tutorial [20], particularly the writing to a dataset by pattern option, to understand how hyperslabs and MPI-IO can be used to reduce the I/O time of your application.

Summary

HDF5 has many features that make it probably the most used standard file format in HPC today. It's flexible, multiplatform, has a large number of language interfaces, and is easy to use.

In an effort to improve I/O performance and data organization, HDF5 uses a hierarchical approach to storing data.

HDF5 also allows you to associate metadata (attributes) with virtually any object in the data file. Taking advantage of attributes is the key to usable data files down the road. Attributes make HDF5 files self-describing. As with a database, you can access data randomly within a file.

Parallel HDF5 is included in the source code, allowing several processes, either on the same node or on different nodes, to write to the same file at the same time. This capability can reduce the time an application spends on I/O, because each process performs only a portion of the I/O.

HDF5 has a large number of wonderful features in addition to parallel performance, so it is definitely worth taking the time to experiment and to understand what it can do to improve application scalability and performance. ■■■

LISTING 9: Hyperslab Data

```
$ h5dump sds_chnk.h5
HDF5 "sds_chnk.h5" {
GROUP "/" {
   DATASET "IntArray" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SIMPLE { ( 8, 4 ) /
                          ( 8, 4 ) }
      DATA {
      (0,0): 1, 1, 2, 2,
      (1,0): 1, 1, 2, 2,
      (2,0): 1, 1, 2, 2,
      (3,0): 1, 1, 2, 2,
      (4,0): 3, 3, 4, 4,
      (5,0): 3, 3, 4, 4,
      (6,0): 3, 3, 4, 4,
      (7,0): 3, 3, 4, 4

   }
}
}
```

INFO

[1] Parallel I/O: *http://www.admin-magazine.com/HPC/Articles/Improved-Performance-with-Parallel-I-O*

[2] HDF: *https://en.wikipedia.org/wiki/Hierarchical_Data_Format*

[3] HDF5 tutorial: *http://neondataskills.org/HDF5/Exploring-Data-HDFView*

[4] HDFView: *https://support.hdfgroup.org/products/java/hdfview/*

[5] Perl support: *http://search.cpan.org/~chm/PDL-IO-HDF5-0.6501/hdf5.pd*

[6] Lua support: *https://colberg.org/lua-hdf5/*

[7] Node.js support: *https://github.com/HDF-NI/hdf5.node*

[8] Erlang support: *https://github.com/RomanShestakov/erlhdf5*

[9] Haskell support: *https://hackage.haskell.org/package/bindings-hdf5*

[10] Parallel HDF5: *https://support.hdfgroup.org/HDF5/PHDF5/*

[11] Predefined datatypes: *https://support.hdfgroup.org/HDF5/doc/UG/HDF5_Users_Guide-Responsive%20HTML5/index.html#t=HDF5_Users_Guide%2FDatatypes%2FHDF5_Datatypes.htm%23TOC_6_2_2_Predefinedbc-4&rhtocid=6.1.0_2*

[12] h5py Python library: *http://www.h5py.org/*

[13] Anaconda Python: *https://www.continuum.io/downloads*

[14] h5py Quick Start guide: *http://docs.h5py.org/en/latest/quick.html*

[15] NumPy: *http://www.numpy.org/*

[16] HDF5 Python docs: *http://docs.h5py.org/en/latest/*

[17] Fortran 90 examples in HDF5: *https://support.hdfgroup.org/ftp/HDF5/examples/src-html/f90.html*

[18] MPI-IO: *http://beige.ucs.indiana.edu/I590/node86.html*

[19] ph5example: *https://support.hdfgroup.org/ftp/HDF5/current/src/unpacked/fortran/examples/ph5example.f90*

[20] HDF5 parallel topics tutorial: *https://support.hdfgroup.org/HDF5/Tutor/parallel.html*

[21] Writing to a dataset by chunk: *https://support.hdfgroup.org/ftp/HDF5/examples/parallel/hyperslab_by_chunk.f90*

[22] Row-major order: *https://en.wikipedia.org/wiki/Row-_and_column-major_order*

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!

Developing an AI system with Markov chains

Sure-Fire Success

Markov chains model systems that jump from state to state with predetermined probabilities, but can they help write new columns like this one after learning from previously written articles? *By Mike Schilli*

Hard to believe, but true: This edition of my programming column marks its 20-year anniversary. Lately, I've been diving into artificial intelligence topics, a field that is increasingly replacing traditional jobs, but I wonder if a machine could possibly one day replace authors like me? An AI system would certainly have a number of advantages over a human writer, because a robot would no doubt deliver the manuscript on time every time, to the amazement of my editors, who are not fond of my bad habit of watching deadlines swoosh by. Also, I could afford to work less hard and have more time to kick back, relax, and spend my days pursuing my surfing hobby at various Pacific beaches (Figure 1).

Moody like the Weather

An algorithm that writes columns automatically could be implemented as a so-called Markov chain, named after the Russian mathematician Andrei Markov. Markov chains are stochastic processes with different states that change according to predetermined probabilities.

The weather forecasting model shown in Figure 2, for example, says the chance of a sunny day following a "Sun" state is 65 percent. At 30 percent, the weather will change to the "Clouds" state, and at 5 percent, it goes directly

Figure 1: Target: The author surfing in Hawaii whilst an AI system writes his column.

to the "Rain" state. The probabilities apply only to the current state; a change to the next state does not require knowledge of any event in the past, so the probability of a transition is very easy to calculate.

In a transition matrix such as that shown in Figure 2, the computer only needs to select the row with the current state, such as the third row for the Clouds state (C), to find out that there is a 25 percent chance of the weather being sunny, a 25 percent chance of rain, and a 50 percent chance of skies remaining cloudy. Equipped with corresponding random generators, the model jumps back and forth between states, and a later analysis of where this Monte Carlo method leads eventually allows conclusions to be drawn about the most likely weather in the future.

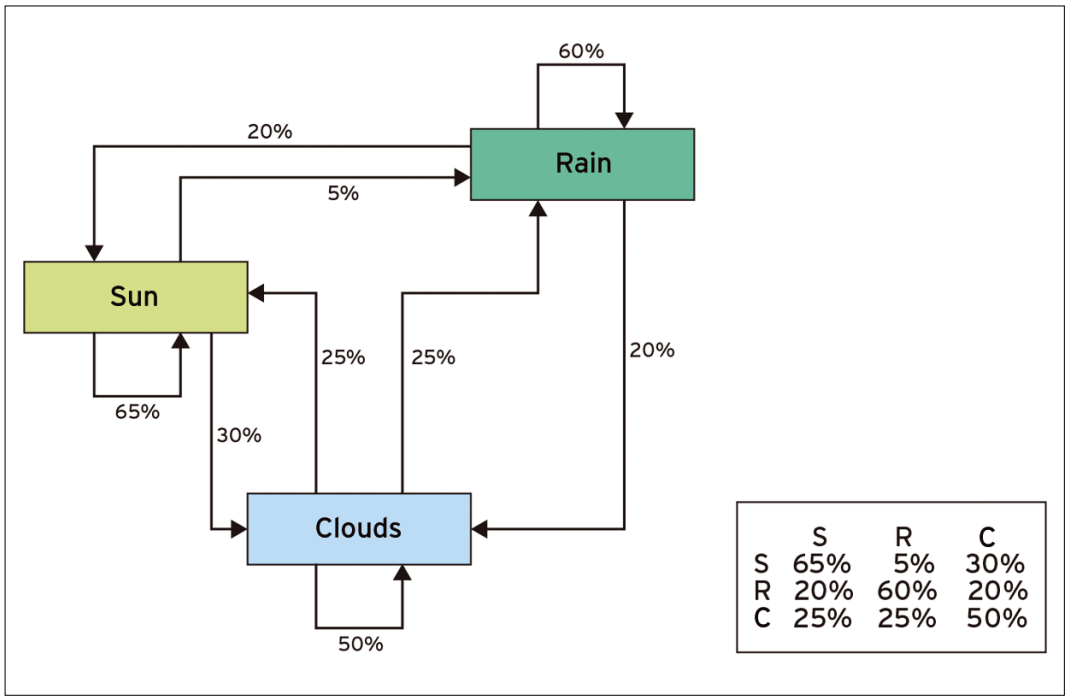


Figure 2: Markov diagram for the weather forecast, with the transition matrix in the lower right corner.

Sentences from the Machine

Such "random walks" can predict not only the weather, they can generate sequences of words that sound like they were written by a human author. For this purpose, a program first analyzes the word sequences used in an extensive body of text. It notes which words follow one another in a sentence, and pieces together a new text from learned phrases. For example, it might realize that, in corpus, three given words are sometimes followed by word A and sometimes by

LISTING 1: randomtext.py

```
01 #!/usr/bin/python3
02 from nltk.tokenize import word_tokenize
03 from collections import deque
04 import re
05 import random
06
07 re_special=re.compile("^[,.:]$")
08 re_word=re.compile("^\w+$")
09
10 def token_feed(file):
11     string  = open(file, 'r').read()
12
13     for i in word_tokenize(string):
14         if re_special.match(i) or \
15            re_word.match(i):
16             yield(i)
17
18 def tokens_to_text(tokens):
19     out=""
20     for word in tokens:
21         if(not re_special.match(word)):
22             out += " "
23         out += word
24     return out
25
26 def window(seq, n):
27     it = iter(seq)
28     win = deque(
29       (next(it, None) for _ in range(n)),
30       maxlen=n)
31     yield win
32     append = win.append
33     for e in it:
34         append(e)
35         yield win
36
37 def statemap_gen(file):
38     statemap={}
39     tokens=token_feed(file)
40     for state in window(tokens,4):
41         key   = list(state)
42         value = key.pop()
43         key   = tuple(key)
44
45         if key in statemap:
46             statemap[key].append(value)
47         else:
48             statemap[key] = [value]
49     return statemap
50
51 def generate_from(file):
52     statemap=statemap_gen(file)
53     key=list(
54       random.choice(list(statemap.keys())))
55     words_new=list(key)
56
57     for i in range(200):
58         if not tuple(key) in statemap:
59             continue
60         next_token = random.choice(
61           statemap[tuple(key)])
62         words_new.append(next_token)
63         key.append(next_token)
64         key.pop(0)
65
66     return tokens_to_text(words_new)
67
68 print(generate_from('all.txt'))
```

word B. When generating new prose, the algorithm will roll an imaginary die each time to determine whether the three initial words are followed by A or B, according to the probabilities in which they occur in the original text.

To start off, the algorithm separates the body of text into words (tokens) and then analyzes it step by step with a rolling window of words. The first $(n - 1)$ words of the window determine the state of the machine at the current time, and the last word in the window is the value the machine assumes during the transition to the next state.

Rolling Window

For example, for a window size of $n = 3$, from the text

Humpty Dumpty sat on a wall,
Humpty Dumpty had a big fall.

it extracts the sequences "Humpty Dumpty sat," "Dumpty sat on," "sat on a," and so on. The first two words (Humpty Dumpty) respectively determine the current state of the machine; the following word ("sat") is the next state value. In the second part of the nursery rhyme, the algorithm stumbles again on the "Humpty Dumpty" state. However, it sees not the word "sat," but rather the word "had" following the current state. When it later creates random text, the algorithm now has two options: It can either follow up with "sat," or it can choose "had," and it will do so according to calculated random values. The result is text that shows slight human qualities because it copies word sequences, but occasionally performs crazy jumps between contexts, which can sound either confusing or funny.

The oldest Programming Snapshot in *Linux Pro Magazine* dates back to 2003 (although the German edition started in 1997), and Listing 1 [1] reads all raw manuscripts cobbled together of every single issue since then in the file `all.txt`. The file name is first passed to the `generate_from()` function in line 68. The `statemap_gen` function starting in line 37 then splits the file into tokens with `token_feed()` and passes on the resulting list of the `window()` function from line 26, which accepts $n = 4$ as the window width. As explained previously, the status of the Markov chain in this case is made up of the three last words; the fourth is part of the result state.

To prepare the tokenizer of the `nltk` package, it needs to be installed with `pip3 install nltk`. Moreover, the tokenizer dataset *punkt* still needs to be downloaded separately in a Python shell (Figure 3). The tokenizer interprets punctuation marks, such as commas or colons (the regular expression in line 7 defines them all), as individual tokens. This approach later increases the readability and entertainment value of the random text.

Figure 4 shows how the algorithm generates a state file of the Markov chain from the nursery rhyme and the window length $n = 3$. Note how the `statemap` generated offers both "sat" and "had" as a follow-up state for "Humpty Dumpty":

```
('Humpty', 'Dumpty'): ['sat', 'had']
```

The Python code shows some of the spice of the language, such as the `yield()` operator in the `token_feed()` function in line 16: So that Python can iterate over the components of an object using a `for` loop, it must be of type `Iterable`. These objects do not spill out their elements in one go, but they pass them through one piece at a time with each call to `yield()`.

The advantage of this process is that the script does not need to store all of the data in a construct in memory; rather, it can get away with only calculating the data when needed. Therefore, an object can contain an infinite number of elements; as long as they are never needed all at once, the illusion stays alive despite limited memory.

Python also provides a variety of data structures. Double-ended queues, called `deque`, as used in the `window()` function

```
$ python3
Python 3.4.3 (default, Nov 17 2016, 01:08:31)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> nltk.download()
NLTK Downloader
---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> d

Download which package (l=list; x=cancel)?
  Identifier> punkt
    Downloading package punkt to /home/mschilli/nltk_data...
      Unzipping tokenizers/punkt.zip.
```

Figure 3: Downloading data for the NLTK tokenizer.

```
$ python3
Python 3.5.2 (default, Sep 14 2017, 22:51:06)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information
.
>>> from randomtext import statemap_gen

>>> print(open('test.txt', 'r').read())
Humpty Dumpty sat on a wall,
Humpty Dumpty had a big fall.
All the king's horses and all the king's men
Couldn't put Humpty together again.

>>> print(statemap_gen('test.txt'))
{('Could', 'put'): ['Humpty'], ('king', 'men'): ['Could'], ('king', '
horses'): ['and'], ('on', 'a'): ['wall'], ('Humpty', 'Dumpty'): ['sat
', 'had'], ('wall', ','): ['Humpty'], ('Dumpty', 'sat'): ['on'], ('Du
mpty', 'had'): ['a'], ('All', 'the'): ['king'], ('big', 'fall'): ['.'
], ('and', 'all'): ['the'], ('.', 'All'): ['the'], ('had', 'a'): ['bi
g'], (',', 'Humpty'): ['Dumpty'], ('all', 'the'): ['king'], ('togethe
r', 'again'): ['.'], ('the', 'king'): ['horses', 'men'], ('Humpty', '
together'): ['again'], ('horses', 'and'): ['all'], ('sat', 'on'): ['a
'], ('fall', '.'): ['All'], ('a', 'big'): ['fall'], ('a', 'wall'): ['
,'], ('put', 'Humpty'): ['together'], ('men', 'Could'): ['put']}
```

Figure 4: Dividing text into tokens and creating the Markov model.

in lines 26-35, can be modified really fast if the code limits itself to adding or removing elements to or from the head or tail of the queue. The only flaw is that the caller of the `window()` function cannot manipulate the `deque` object returned, or the algorithm gets confused.

For this reason, line 41 uses `key=list(state)` to copy the elements of the `deque` object into a Python list, which the `statemap_gen()` function can modify to its heart's content. It interprets the first $(n - 1)$ elements as a Markov state, and the last item as a possible next state.

In line 43, `key` is a Python tuple that can no longer be modified. The dictionary `statemap` maps such tuples to one or more follow-up state values. In this way, the algorithm can later look up possible follow-up state values quickly and select one randomly with `random.choice()` in line 60, leading the text on unpredictable paths and imbuing the algorithm with human-like traits.

The result can be seen in Figure 5. However, it demonstrates that the applied Markov chain procedure does not automatically provide exciting new magazine articles. The grammar is still lacking enormously, and the content doesn't really make any sense. This column will probably have to be created manually for the foreseeable future. ■■■

```
$ ./randomtext.py
 files should I only I be published on the screen, the Meta Ctrl J. I
nstead of dozens of separate entries just two lines, and the POE kern
el can reassume control, leaving everything else. No need to make sen
se of disassembled assembler code. Formatting ms Controlling program
flow returns to the currently processed element and then runs C execu
te C to handle cases where the reference hash contains optional eleme
nts. On code code directory and restores the data fed to the bottom o
f my water tank from time to complete all those painstaking entries,
you can add to the preset, and needs to use a tool is required to han
dle any errors that occur immediately after the program you are monit
oring. If a conflict occurs, it passes the command is running. You ca
n quit an interactive interpreter session using the Logical Volume Yo
u can assign only one energize and schmoop factors. The scripts given
 in this table, making the restore much easier later on can easily di
stinguish them from data lines. In other words, do try to be an Extre
me Programming Enthusiast to see it turn out
```

Figure 5: The AI system writes the programming column of all programming columns.

INFO

[1] Listing for this article:

ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/205/

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 go to *mschilli@perlmeister.com*, he will gladly answer any questions.

Lead Image © lassedesignen - Fotolia

OpenType and HarfBuzz

Hacking OpenType Fonts

Thanks to OpenType and HarfBuzz, you can now modify fonts like a designer.

By Bruce Byfield

Until now, digital fonts have been a static resource. From an application, users can select a font, as well as its size and weight, but no other options have been available to the average user. However, with OpenType [1] established as the dominant format for font files on all operating systems, a new relationship is opening up in which users can modify the display of fonts with the strategic placement of code tags. It's a command-line solution applied to desktop technology.

The OpenType format specifications were first drawn up by Microsoft and Adobe in 1996. However, in 2007, it became an ISO standard and soon became the file format of choice for font designers, including those developing free-licensed fonts. OpenType has a distinct advantage over both Type 1 (Postscript) and TrueType fonts (the main formats in the 1990s), because its Unicode support potentially gives it a full range of accents and other diacritical marks, as well as all the major writing systems – or scripts, as OpenType refers to them. As computing moved away from the English-dominated ASCII standard, increasingly, neither Type 1 nor TrueType could measure up. At the same time, because OpenType is backwardly compatible with both Type 1 and TrueType fonts, the formats can still be used by batch converting them with a script in FontForge, the open source font designer. You can tell an OpenType font by its file extension, usually `.otf`.

As OpenType became popular, its support in free software was aided by the spread of HarfBuzz [2], which renders a Unicode character into the correspond-

BRUCE BYFIELD

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

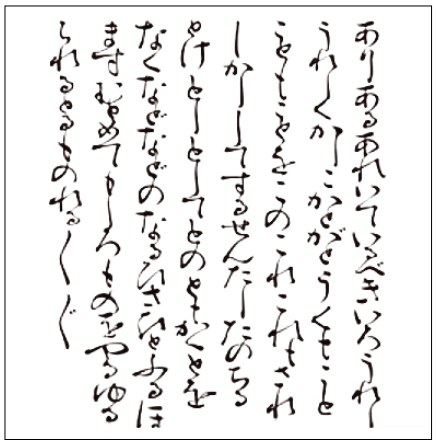


Figure 1: A simple script like Japanese or English consists of discrete characters in a standard order.

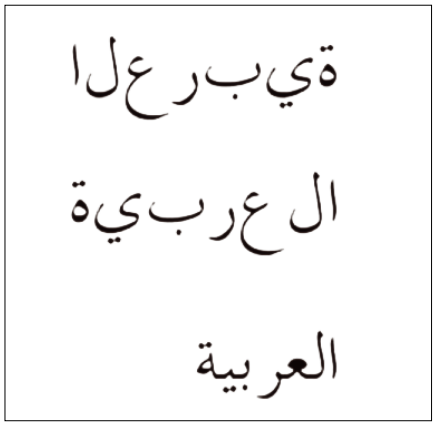


Figure 2: A complex script like Arabic or Persian uses a combination of joined and separate letters, whose positioning can change with context.

ing glyph in a font. In the early years of the millennium, HarfBuzz became necessary, because font rendering was done by three separate applications; FreeType, Pango, and Qt. This meant that the same font could be displayed differently in different applications. At first, lead developer Behdad Esfahbod released an early version of HarfBuzz – now called Old HarfBuzz – that tried to draw on all three font renderers, but in the end, the complications proved so great that an independent version was written from scratch, becoming the HarfBuzz now used today in Chrome OS, Chrome, Firefox, and KDE. Other projects, such a Gimp, Inkscape, Krita, and Scribus are now in the process of switching to HarfBuzz as well.

HarfBuzz receives attention mostly for its support of all Unicode-supported scripts. Together with a font like Noto, which supports the full array of Unicode characters up to 6.1, HarfBuzz today can consistently represent not only simple scripts such as English, Greek, or Japanese, which use sequences of standalone glyphs (Figure 1), but also more complex scripts like Persian, Arabic, or the Indic languages (Figure 2). In these scripts, some glyphs are joined, while others stand alone and can change position according to context. In fact, the name "HarfBuzz" is a transliteration of the Persian for "OpenType," reflecting both the format it was designed for and the fact that its 15-year-long development began with Esfahbod's efforts to render Persian correctly in web browsers.

That is a time-consuming accomplishment, because each language requires its own shaping rules. Yet what HarfBuzz does for Western European languages like English is almost as impressive. Just as word processors were an improvement over typewriters because they could print italics, rather than indicating them by underlining, so HarfBuzz is an improvement over earlier font renderers because it automatically adds available advanced features. With the spread of OpenType and HarfBuzz, computer users now have the potential to produce copy as polished as anything coming from a professional print shop. What is more, users can select the features they want to use with a bit of simple hacking.

Code Tags

With OpenType and HarfBuzz, features are edited by an extensive series of code tags. Most tags can be enabled simply with a four-letter abbreviation. For example, to enable the use of italics, you would use the tag `ital`. Similarly, `-ital` turns off italics. Both of these notations also have a more complex form that adds a value. Usually, the value simply toggles the feature – for example, `ital=1` is the same as `ital`, whereas `ital=0` is the same as `-ital`. However, in a few cases, the ability to add a value offers a selection of several choices.

Many OpenType tags are mostly relevant to a particular language, although experimenting with tags in another language can sometimes lead to interesting results. Here, I am concerned only with those useful in English and to a lesser extent with other Western European languages that use a Latin-based alphabet, because those are the typographical standards I know and that most readers are likely to find most relevant.

TABLE 1: Capital Tags

smcp	Replaces lowercase letters with small capitals
c2sc	Replaces uppercase letters with small capitals
cpsp	Improves spacing between all-capital text

Figure 3: Small capitals improve the look of strings of uppercase letters.

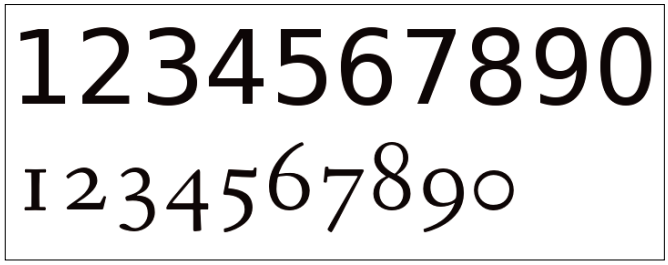


Figure 4: Lining or ranging figures are most common today, but old-style figures are more readable in blocks of text.

TABLE 2: Numeral Tags

inum	Replaces numerals with lining figures
onum	Replaces numerals with old-style figures
tnum	Replaces numerals with tabular figures
frac	Reduces the size of any two numbers separated by a forward slash
zero	Replaces a regular zero with a zero with a slash through it

Figure 5: Ligatures are single glyphs that replace awkward combinations of two or more glyphs.

Wikipedia gives a complete list of OpenType tags [3]. Some enable features that are available in applications like word processors, but may not be available in applications like web browsers. Next, I discuss some of the most useful.

Case Tags

English uses uppercase and lowercase characters. For layout, some of the most useful are small capitals, a modified set of glyphs that make a string of uppercase letters fit better into a body of text. Most word processors already support small capitals, but many do so by manufacturing their own versions, which are clumsy makeshifts that look nothing like the small capitals the font designers intended (Figure 3). Table 1 lists the tags used for capitals.

Numeral Tags

English generally uses lining, aka ranging figures, whose glyphs all sit on the same baseline. However, professional designers often prefer old-style figures, in which each glyph has its own baseline (Figure 4). Additionally, applications like spreadsheets can benefit from tabular figures, in which each glyph has a uniform width, although their availability is rare.

Typically, users write fractions with full-sized characters, making them difficult to read. Special character dialogs can replace a few common fractions, but OpenType fonts can be set to reduce the size of any two numbers separated by a forward slash, making them more readable. Some users might also appreciate the use of a slashed zero to distinguish zero from an upper case O. Table 2 shows the tags for modifying numerals.

Letter Tags

Even the best-designed font usually has glyphs that do not fit side by side. Often, they involve combinations starting with "f" such ff or fi. To improve legibility, professional typesetting replaces these combinations with ligatures, a single glyph that represents both of these letters, but is better designed (Figure 5). Until now, users have had to pick ligatures out of a special character dialog, which is enough of a distraction that many users never bother. With OpenType and HarfBuzz, ligatures can be added automatically (Table 3).

Similarly, word processors and layout applications have long had the ability to write subscript and superscript characters. However, as with small capitals, their position has often been set by the application. With OpenType, font designers can now specify a position for subscript and superscript (Table 3).

Of course, not every tag is available in all OpenType fonts. Some designers may prefer that their fonts not have certain features, and some probably will not add certain tags until their fonts support the scripts that use them. Yet, even with these limitations, OpenType and HarfBuzz make uniformity across applications greater than ever was possible in the past. No longer do users have to depend on an application to implement features; they can be enabled within the font itself.

TABLE 3: Letter Tags

liga	Replaces awkward letter combinations with a redesigned single character
subs	Replaces characters with subscript glyphs
sup	Replaces characters with superscript glyphs
title	Replaces characters with glyphs suitable for large type, such as titles

Figure 6: OpenType tags are stored in tables in the font file.

Applying OpenType Tags

In an OpenType font, tags are set in a series of layout tables. In free software, these tables can be accessed from within FontForge [4] (Figure 6). FontForge fully supports OpenType, so to enable a tag, designers only need to design all the glyphs and then right-click on each glyph separately and select the tag from the drop-down list. So long as the font is free-licensed, the designer or any other user can enable or disable tags at will – although backups of the OpenType file should be made first in case a mistake is made.

Users can also apply tags in specific applications. In any application that uses HarfBuzz, tags can be added in any field for font selection. Each tag must be preceded by a hyphen, and tags can be chained together, with no spaces between them and the font name or each other. For example, in Figure 7, the Fanwood font is selected in the paragraph style dialog, and tags are added to use small capitals and old style figures.

In LibreOffice, tags can be added to paragraph styles, or with even more flexibility, to character styles, where they can be applied only when needed. For convenience, you can add tags to a template – including your default – so that they don't have to be added to each new document.

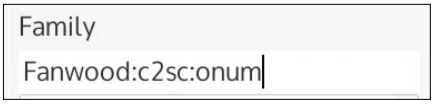


Figure 7: Tags can be added in a field for font selection, so long as the application uses HarfBuzz.

New Controls

The combination of the OpenType format and the HarfBuzz "text shaping library" gives users unprecedented control over fonts. If anything, some designers may be uncomfortable with how much control users now have over their creations. However, in effect, users can now modify fonts in the same way they can modify source code, although fonts are even more accessible than code.

Moreover, the availability of tags is only the beginning. Currently, OpenType Font Variations [5] are being developed as a modern version of Adobe's multiple master fonts [6]. With OpenType Font Variations, different weights of a font can be selected as easily as dragging a slider bar. Since making a glyph thinner or thicker without distorting it often requires design changes, an OpenType Font Variation can include nearly 100 changes (Figure 8). As I write, OpenType Font Variations are still in development, and whether a weight will be selectable through tags is apparently still uncertain. Yet, one way or the other, the new technology promises to give users even more control over their fonts.

Figure 8: Font variations promise to make different weights easier to select, even allowing for changes in design as the weight changes.

Clearly, free fonts have come a long way since their first appearance in the early years of the millennium. At the time, many people questioned whether designers would ever be interested in designing free fonts. Many even wondered if fonts, as artistic works, should be an exception to free software. Yet today, not only are hundreds of free fonts available, but, thanks to OpenType and HarfBuzz, fonts are now as hackable as any other part of the Linux stack. ■■■

INFO

[1] OpenType: *https://en.wikipedia.org/wiki/OpenType*

[2] HarfBuzz: *https://freedesktop.org/wiki/Software/HarfBuzz*

[3] OpenType tags: *https://en.wikipedia.org/wiki/List_of_typographic_features#OpenType_typographic_features*

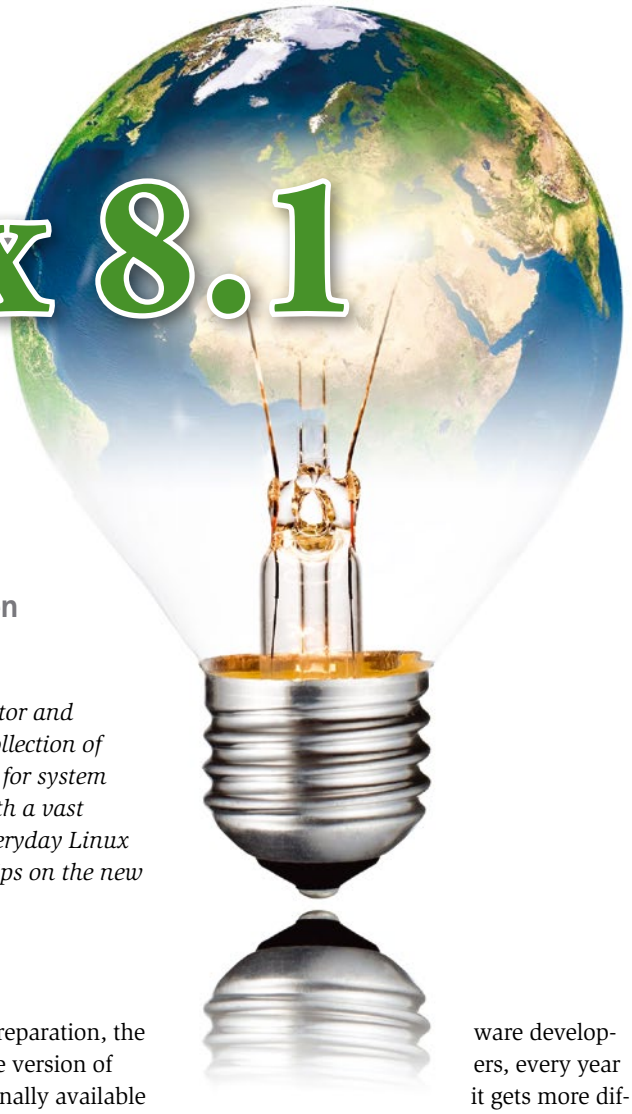
[4] FontForge: *https://fontforge.github.io/en-US/*

[5] OpenType Font Variations: *https://fontbureau.typenetwork.com/news/article/opentype-font-variations-open-up-a-world-of-possibilities*

[6] Multiple masters: *https://en.wikipedia.org/wiki/Multiple_master_fonts*

Klaus talks about some new features in the latest Knoppix

Knoppix 8.1

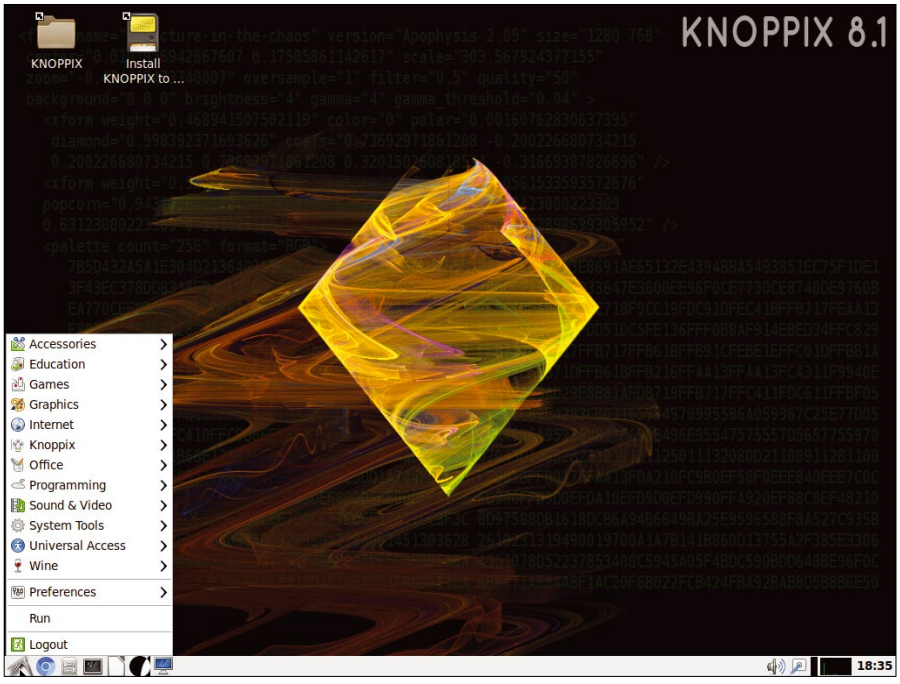


The latest Knoppix comes with a new I/O scheduler, and the new hybrid ISO image format allows you to boot from either a DVD or USB stick. Klaus talks about the changes with the latest edition of Knoppix, and offers a glimpse at some of the problems he faces when producing a new Knoppix version. *By Klaus Knopper*

Editor's Note: *Our own Professor Knopper has another life as the creator and maintainer of the popular Knoppix Live Linux distro. With its large collection of troubleshooting and diagnostic tools, Knoppix is a trusted companion for system administrators and other IT professionals, but Knoppix also comes with a vast collection of desktop tools, which means it is fully functional as an everyday Linux system on a live disk. In this article, Klaus offers some thoughts and tips on the new Knoppix 8.1, which is included on this month's DVD.*

After months of preparation, the English language version of Knoppix 8.1 is finally available [1]. Due to the increasing complexity of applications and dependencies, as well as the strategic decisions of software developers, every year it gets more difficult to make everything work together smoothly. All the changes and redesigns in recent Debian releases mean I now

Klaus Knopper is an engineer, creator of Knoppix, and cofounder of LinuxTag expo. He works as a regular professor at the University of Applied Sciences, Kaiserslautern, Germany.



Lead Image © Jan Treger, 123RF.com

Figure 1: The English edition of Knoppix 8.1 is now out in the world.

LABOR-INTENSIVE FIXES FOR SMALL BUGS

Especially annoying are programs that used to work perfectly in older versions but terminate with error messages or show unexpected behavior with updates. Just keeping the old version is not always a good solution, because the new version is often created to address security problems or software incompatibilities.

Often the only solution is to use the new version but to spend some time developing a workaround. For example, TeXmacs (Figure 2) is a graphical editor for the TeX typesetting system that lets you include live sessions of mathematical software, such as the popular Maxima computer algebra system. Internally, TeXmacs checks for the availability of a Maxima plugin by calling Maxima with a `version` option. In the latest version of Maxima, that `version` option produces an empty result, so TeXmacs thinks Maxima is not working and disables the live algebraic session option. As a workaround, I added some code in the Maxima front-end command to output Maxima's version number (like in the old version), so TeXmacs is now happily creating Maxima sessions in documents again. As you can imagine, it is very difficult to even *find* such problems, not to mention repairing them in software you aren't too familiar with. Regression tests that involve interaction with each and every installed program are not feasible.

spend more time on testing and workarounds than I do on development and programming. (For an example of the kinds of problems I face, see the box entitled "Labor-Intensive Workarounds for Small Bugs.")

Knoppix 8.1 (Figure 1) contains 4,100 software packages (including libraries and supporting shared data), which results in around 4,500 executable files. Only a small fraction of these programs appear in the desktop menus; many of the latest and greatest tools for data rescue or advanced networking are only found at the command line.

The base system is installed inside a cloop-compressed image called `KNOPPIX` inside the `KNOPPIX` folder, and additional programs for 3D construction and video editing appear in the `KNOPPIX1` add-on file. Splitting the contents is necessary because the FAT32 filesystem, as well as the traditional ISO 9660 filesystem used in DVDs, only support file sizes smaller than 4GB.

Although it is compressed to around 4.2GB, the total amount of data installed on Knoppix 8.1 is 11GB. The data is transparently decompressed by the cloop kernel module.

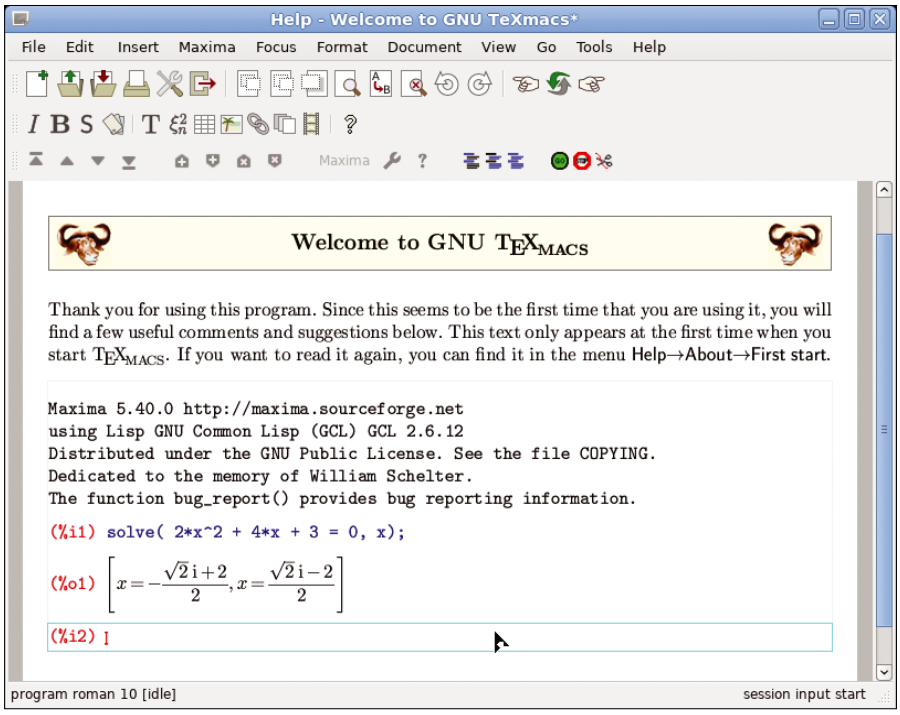


Figure 2: The TeXmacs TeX editor required some extra attention.

Knoppix uses its own, script-based startup system and only starts services needed for operation, so the memory footprint is around 150MB after booting into the graphical user interface.

Although the kernel is auto-selected as 32- or 64-bit on startup, installed programs are 32-bit, so all 32- and 64-bit Intel/AMD platforms are supported. Thus Knoppix can still be used on old computers with few CPU and memory resources.

Because the distributed ISO image is now in ISO hybrid format, it is no longer necessary to burn Knoppix to a DVD first; you can write it directly to a USB flash disk using `dd` or `cp` on Linux or using imaging tools like Win32 Disk Imager in Windows.

If you feel like experimenting, you can also create your own ISO hybrid image. (See the box entitled "Hybrid Image for DVD and Flash Disk.")

HYBRID IMAGE FOR DVD AND FLASH DISK

If you wish to create your own new hybrid ISO image, make sure you are using the development snapshot 1.4.9 from the xorriso project page [2].

The new hybrid image format adds a partition table for booting from flash to the ISO image [3]. Knoppix 8.1 includes some optimizations related to the hybrid generation procedure – to make the image more compatible with BIOS systems and UEFI firmware. I used an additional GUID partitioning scheme in version 8.0, but this approach turned out to confuse other operating systems after automatic expansion of the last partition.

The development snapshot for xorriso is not capable of creating a pure DOS/MBR-based partition table, and the partition containing the EFI boot files is now outside of the first ISO partition. If you would like to remaster the ISO image file, the command should be similar to Listing 1, where `$SRC` is the directory with a copy of the first Knoppix partition's content, `efi.img` is an image of the second, and `REISERFS.IMG` an image of the third (empty) partition, copied from the original downloaded file (not the `flash-knoppix`-generated version). `/usr/lib/ISOLINUX/isohdpfx.bin` is a hybrid MBR template included in the syslinux package.

Thanks to Thomas Schmitt from the xorriso project for all the useful tips!

LISTING 1: Creating a Hybrid Image

```
01 xorriso -compliance iso_9660_level=3 -as mkisofs -l -r -J \
02         -iso_mbr_part_type 0x83 \
03         -isohybrid-mbr /usr/lib/ISOLINUX/isohdpfx.bin \
04         -V "KNOPPIX_8" -A "KNOPPIX V8" \
05         -boot-load-size 4 -boot-info-table \
06         -b boot/isolinux/isolinux.bin -c boot/isolinux/boot.cat \
07         -no-emul-boot \
08         -hide-rr-moved --sort-weight-list "$sortlist" \
09         -eltorito-alt-boot \
10         -e --interval:appended_partition_2:all:: -no-emul-boot \
11         -append_partition 2 0xef efi.img \
12         -append_partition 3 0x83 REISERFS.IMG \
13         -partition_offset 16 \
14         -o Knoppix-New.iso "$SRC"
```

Rebootless Auto-Resize of Data Partition on Flash

Knoppix traditionally uses an additional partition with a native Linux filesystem in order to overlay one's own changes and installations on top of the read-only system data mounted from the compressed `KNOPPIX*` files. With the flash-knoppix tool, this partition can be sized by the user as needed. However, when booting from a USB flash disk that was installed by the image method directly from the ISO file, the last partition is only 4MB in size at first sight in order to save space in the distributed image. The initial ramdisk of Knoppix now contains code to expand this last partition on-the-fly up to the last available unused sector on the medium. Other distros, like Raspbian on Raspberry Pi, can do this as well, but when even just one partition is mounted from disk while repartitioning, the kernel won't accept the new partition table until a reboot occurs. For this reason, the Knoppix initial script unmounts and rescans the disk after repartitioning and before attempting to resize the filesystem – from the ramdisk that contains all necessary tools. This way, no reboot is needed and the user doesn't have to change the boot order sequence again in order to use the overlay partition.

The `resizedata` function is, of course, only called after a few safety checks – if it's running from a Knoppix-image-installed USB flash disk, enough empty space must be available after the last partition, and the current filesystem must be labeled `KNOPPIX-DATA`. If any of these is not the case, the data partition will be ignored, and the system will continue running with only a ramdisk overlay.

New I/O Scheduler

One of the most interesting features of the newer Linux kernels is the introduction of multi-queue schedulers, which allow parallelizing of read/write requests depending on selectable strategies. This feature is most useful for devices with low latency, as well as rotational disks with high seek latency. The Budget Fair Queueing (BFQ) scheduler has been introduced in the mainstream kernel for multi-queue mode only. Previous Knoppix versions had BFQ included as a nonstandard kernel patch, so this is a most welcome change for Knoppix that eliminates the need for an additional patch. BFQ attempts to distribute I/O bandwidth based on a "budget" for each newly started task, so the system should feel faster interactively, and programs should load quicker, even under massive parallel disk usage. Of course, BFQ can't circumvent physical bandwidth limits.

Where's Firefox?

One very controversial design decision of the Mozilla developer team is the removal of direct support for ALSA – the standard Linux sound system – in Firefox. Firefox now requires PulseAudio, but not only as a library, which is, of course, present in Knoppix, but as a sound server system running on the computer. Knoppix does not use PulseAudio in server mode because it can indefinitely block sound output for other programs, which is not acceptable for accessibility tools, such as speech assistance systems. Also, no other sound-enabled program requires a PulseAudio server – only Firefox. For this reason, Firefox, starting from version 52, remains silent and displays a confusing message telling the user to install additional software when entering YouTube and other sound-enabled websites. This problem is not just in Knoppix but in all Linux distros that refrain from starting an additional sound server. According to Mozilla [4], the removed ALSA sound support in the official Firefox releases will never be fixed.

An apparent workaround is using a different web browser for sites that require sound. In order to avoid too many questions about "sound not working," I replaced the Firefox icon with Chromium (the open source variant of Chrome) in the LXDE quick-start bar. But Firefox is still present as an alternative under the Internet desktop menu.

Remaster your Own Knoppix

When you copy Knoppix to a new USB flash disk using `flash-knoppix`, it is possible to include personal settings, additional programs, and configuration changes (Figure 3). All changes are compressed to a new overlay image inside

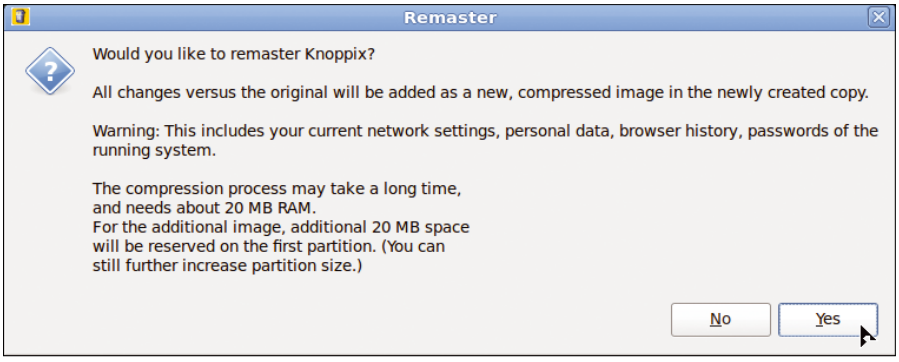


Figure 3: Remastering a custom Knoppix.

LISTING 2: Creating an Overlay

```
01 cd /KNOPPIX-DATA           # or /ramdisk in case of running from DVD
02 mkisofs -input-charset default -no-split-symlink-components \
03  -no-split-symlink-fields -V KNOPPIX_EXTRAS -hide-rr-moved \
04  -cache-inodes -pad -l -R -U -v \
05  -m tmp/\* -m var/cache/apt\* -m var/cache/man\* -m var/log/\* \
06  -m var/run/\* -m var/lib/dpkg/lock\* -m var/lib/NetworkManager/\* \
07  . | create_compressed_fs -L -2 -B 131072 -m - "$destination_image"
```

OTHER UPDATES IN KNOPPIX 8.1

- LibreOffice 5.4.1, Gimp 2.8.20
- Chromium 60.0.3112.78 and Firefox (Iceweasel) 55 web browser with uBlock Origin and NoScript security plugin
- New programs: EtherApe (graphical network monitor), archivemount, Terminator (terminal emulator with many features)
- New version of 3D window manager Compiz 0.9.13.1
- LXDE (Default) with file manager PCManFM 1.2.5, KDE 5.8 (boot option `knoppix desktop=kde`), Gnome 3.24 (boot option `knoppix desktop=gnome`, DVD version only)
- Wine v2.0.0 (git) for integration of Windows™-based programs
- qemu-kvm 2.8 for (para-)virtualization
- Electrum 2.7.9 for managing Bitcoin wallets
- Tiny boot-only CD image inside the `KNOPPIX` directory for computers that can only boot from CD but not from DVD or USB flash drive

the destination's `KNOPPIX` folder, thus minimizing the required space.

A teacher could, for example, install documents and exercises and make a new version containing these changes for the students. It's still easy to reset the modified Knoppix version to the original state by simply deleting the additional overlay file.

The code necessary for creating a new overlay on-the-fly from the running Knoppix system, excluding some unwanted files and cloop-compressing the image to the destination file, is shown in Listing 2.

I'm somewhat surprised that the code in Listing 2 really worked right away. Apparently, it's sufficient to ignore temporary files and sockets in order to create the ISO filesystem stream for compression, and the cloop tool `create_compressed_fs` will pad missing data with zeros in order to always write full blocks.

UEFI Secure Boot Support

Knoppix has been coming with a signed UEFI secure bootloader for a while now, but this does not mean it will boot right away on computers with *UEFI Secure Boot* enabled in the firmware. On first boot with UEFI, a blue screen will be displayed (Figure 4), telling the user to enroll the file `loader.efi` into the secure boot configuration. This is not an error message, though it might look like one; don't give up yet!

Just follow the instructions, which I describe in more detail online [5] . After this "enrollment" has been successfully conducted (it's only needed once to allow booting the Linux loader permanently), Knoppix will display its UEFI boot screen (without the usual penguin logo), and, in most cases, will continue to start up normally. In some cases, the boot procedure will be silent, because of missing text screen support in UEFI, until the graphical interface starts up.

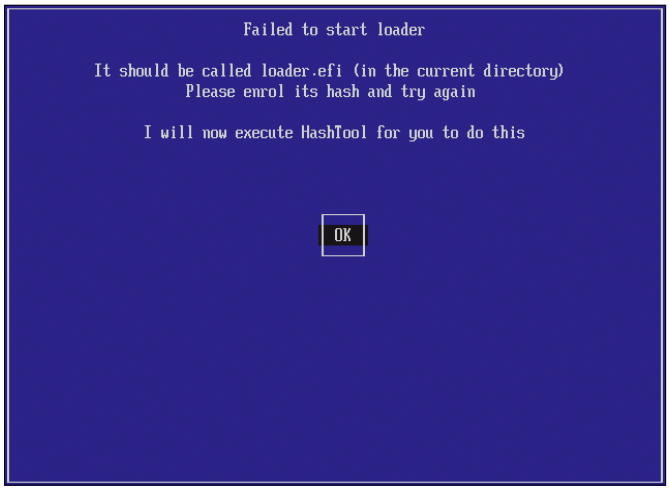


Figure 4: Don't be discouraged by the blue screen: configuring a UEFI boot requires an extra step.

Conclusion

Knoppix 8.1 is the latest update of the Debian-based Knoppix live distribution, which now comes as a single image for DVD and direct USB flash disk installation. For more information about the new release, see the Knoppix website. ■■■

DID YOU KNOW?

1. If you place a shell script called `knoppix.sh` inside the `KNOPPIX` folder on your `flash-knoppix`-generated Knoppix USB flash disk, it will be run automatically during startup.
2. You can start a single program instead of running the system initialization procedure, like the Bash shell or an editor, by adding

   ```
   init=/bin/bash
   ```

   or

   ```
   init=/bin/vim
   ```

   to the boot options after `knoppix` or `knoppix64`.

INFO

[1] Knoppix 8.1: *http://knopper.net/knoppix/knoppix810-en.html*
[2] Xorriso Project Page: *https://www.gnu.org/software/xorriso/#download*
[3] "Professor Knopper's Lab: Hybrid Image Mode" by Klaus Knopper, *Linux Magazine*, issue198, May 2017, *http://www.linux-magazine.com/Issues/2017/198/Professor-Knopper-s-Lab-Hybrid-Image-Mode*
[4] No Firefox ALSA Support: *https://bugzilla.mozilla.org/show_bug.cgi?id=1345661*
[5] Knoppix UEFI Support: *http://knopper.net/knoppix/knoppix-uefi-en.html*

The sys admin's daily grind: Reddit Terminal Viewer

Social Aid

Fortunately, Facebook is not the only place you can chat: When sys admin columnist Charly Kühnast visits the web chat center Reddit, he often goes without a web browser and uses the RTV tool, which can even display photos and videos on plain text consoles. *By Charly Kühnast*

Reddit is a web portal where all sorts of people post all kinds of news from all possible topics and lead impossible discussions about them elsewhere. The allocation in various topic areas (subreddits) ensures a certain degree of structure. Instead of bothering with a browser, I often just use Reddit Terminal Viewer (RTV) [1].

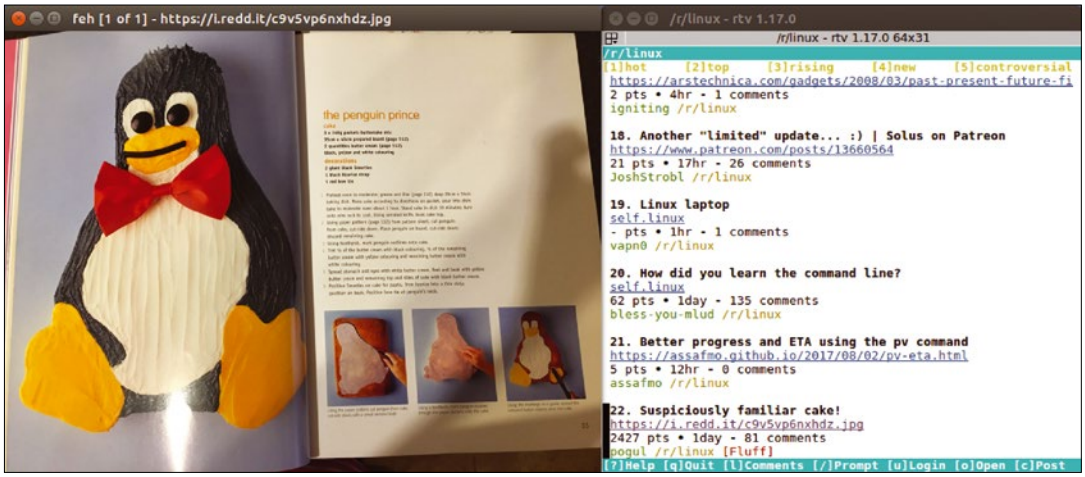


Figure 1: Pixel fineness – as a console program, RTV gets help from the image viewer feh.

The tool is written in Python, which is why the *python* and *python-setuptools* packages have cast their anchors in the system. Using

```
pip install rtv
```

I then run the actual RTV installation. After opening (`rtv`), you are greeted by the Reddit start page.

I either navigate using the arrow keys or Vi style: I can get to the login at the touch of a button (*U*). For me, Reddit is predominately a read-only medium, I rarely post and comment. If I did, then I'd need to log in, and the same goes for evaluating contributions. Using *A*, I can rate a post as positive (upvote); using *Z*, I can express my disapproval (downvote).

The Reddit comment system is implemented well in RTV. Indentations indicate who replied to whom. Using the spacebar, I can hide comment branches that I'm not interested in – for popular or controversial issues in particular, this is the only way to stay on top of things, because the length of comment threads on Reddit is legendary.

If I don't like a key assignment, I change it then and there. To do this, I enter

```
rtv --copy-config
```

to create a configuration file where the default settings are initially stored and which I now can adjust to my taste. As well as the key assignment, it is possible to define the character set (usefully preset to UTF-8) or allow automatic login. I can also set a home page for myself – for me, the subreddit `/r/linux`.

When the Images Learned to Run

Of course, showing media such as pictures or videos is an issue in a shell. However, if I've opened the terminal on a graphical user interface, RTV can call several configurable help programs to display media content. (Or I launch a browser immediately, but that can't be helped.) In preparation, I call

```
rtv --copy-mailcap
```

once. After that, in the home directory, I find the `.mailcap` file, which defines which viewer should play which medium (MIME type). The slim viewer feh is the default image displayer, which I have to install first (Figure 1).

Later in the configuration file, I find links especially for text processors: This is where converters that convert images and videos into ASCII appear and show the content of the media on a text-only interface. ■■■

CHARLY KÜHNAST

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

INFO

[1] RTV: *https://github.com/michael-lazar/rtv*

Help your friends and colleagues make the switch to Linux!

This single issue shows beginners how to:

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

HELP OTHERS
JOIN THE LINUX REVOLUTION!

ORDER ONLINE:
shop.linuxnewmedia.com/specials

Developing concurrent programs with Pony

Horse Power

Pony, an object-oriented programming language with static typecasting, trots down well-mapped paths to deliver secure, high-performance code for concurrent applications.

By Andreas Möller

The still young Pony [1] programming language uses the actor model [2] and capabilities [3] to make deadlocks and data races things of the past. In this article, I take Pony for a test ride with an example application that, once it has compiled successfully, logs the consumption of paint and reports the results in a single line.

Figure 1 revisits the problems of concurrent programming in C and C++. To improve performance, the hypothetical program outsources tasks to concurrent threads that access the shared memory area. Locks manage access to prevent different threads editing data simultaneously, falsifying each other's results in the process, and generating race conditions. Nothing good results when programming errors interact and create deadlocks.

Table 1 summarizes the problems associated with implementation across various languages. The comparison with

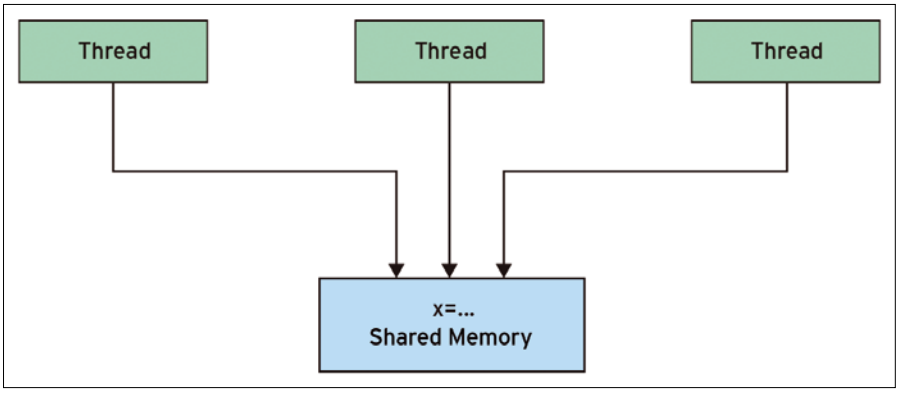


Figure 1: In C and C++, a lock is necessary to allow a thread exclusive write access.

TABLE 1: Feature Comparison

Language Feature	C/C++	Java	Rust	Pony
AoT Compilation	Yes	No*	Yes	Yes
Memory Secure	No	Yes	Yes	Yes
Type Secure	No	Yes	Yes	Yes
Avoids Race Conditions	No	No	Yes	Yes
Avoids Deadlocks	No	No	No	Yes
Actor-Based	No	No	No	Yes

*Officially as of Java 9 [5].
Except for dynamic loading of classes [6].

LISTING 1: Installation on Debian 8

```
echo "deb https://dl.bintray.com/pony-language/ponyc-debian pony-language main" | tee -a /etc/apt/sources.list
apt-get update
apt-get install ponyc-release
```

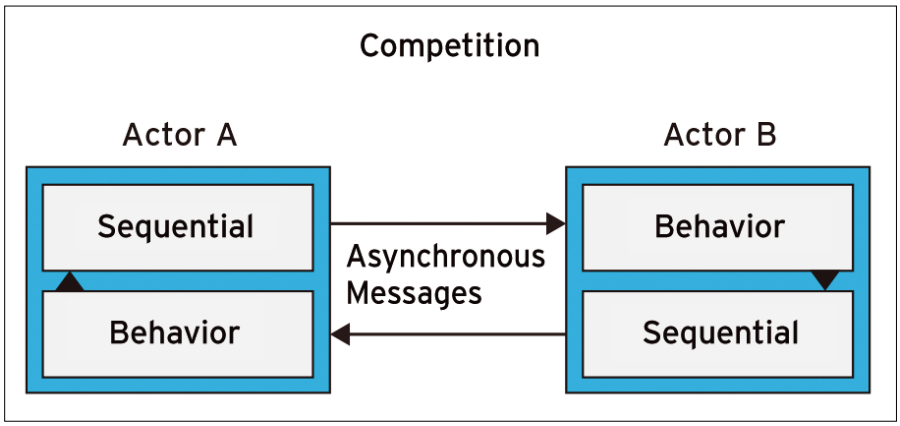


Figure 2: Behaviors accept asynchronous messages and feed them to the actor's sequential operating mode.

Java is inevitable, because the language offers one of the most successful enhancements of the C and C++ method. A comparison with Rust [4] is also appropriate, because the new kid in town also tries to generate secure and fast code for concurrent applications.

Both Pony and Rust compile programs ahead of time (AoT) to create executable binary code. The method saves time because the languages identify and rule out type conflicts in advance. Java, on the other hand, usually first converts source code to non-executable bytecode, which a virtual machine (VM) executes and monitors. Pony and Java both leave memory management to a garbage collector, which Pony integrates into the runtime environment of the compiler results because it lacks a VM.

Actor Model

Pony programs are always free of data races and deadlocks through the agency of actors. As Figure 2 shows, the model does without a shared memory area and therefore does not need to use locks.

LISTING 2: The .pony Files

```
|- blue-horses
  |- blue-horses
  |- main.pony
  |- painter.pony
  |- primitives.pony
  |- resource.pony
```

Because internally the threads (aka actors) work sequentially, data races could theoretically still occur between the competing actors; however, applications mutually exchange application data through asynchronous communications. The sender deletes all pointers to variable data before sending, which also successfully prevents data races in this scenario.

Behaviors (asynchronous functions) are used as mailboxes for incoming messages, which Pony calls like normal methods but runs asynchronously. The actor thus continues its work sequentially. The Pony developers did not create the actor model; the programming languages Erlang, Io, D, and Scala also use it.

Installation

The current release of Pony [7] can be installed using the commands in Listing 1 on a 64-bit Debian system. The first line integrates the packet source into the `/etc/apt/sources.list` file. The second line updates the Debian package database, and the third line installs Pony. There is no Debian package for the i386 architecture; a manual compile is needed in this case.

Listing 2 shows the directory tree of the sample application. If you change to the project directory and compile the program with `ponyc`, the result is the binary executable, as shown in Figure 3, which you can launch with the `./blue-horses` command.

The `main.pony` file is the entry point for the Pony program (Figure 4). It uses the class from the `resource.pony` file to create paint stocks for the primitives (colors) from `primitives.pony` and presents these to the actor of the `Painter` type from `painter.pony`. The actor processes the quantity of paint and reports its consumption in the form of a report to `main.pony`.

Classes

Pony summarizes application data in classes, as the `Resource` class illustrates in the example in Listing 3, which stores paint types and their respective stock. The `class` keyword introduces the class

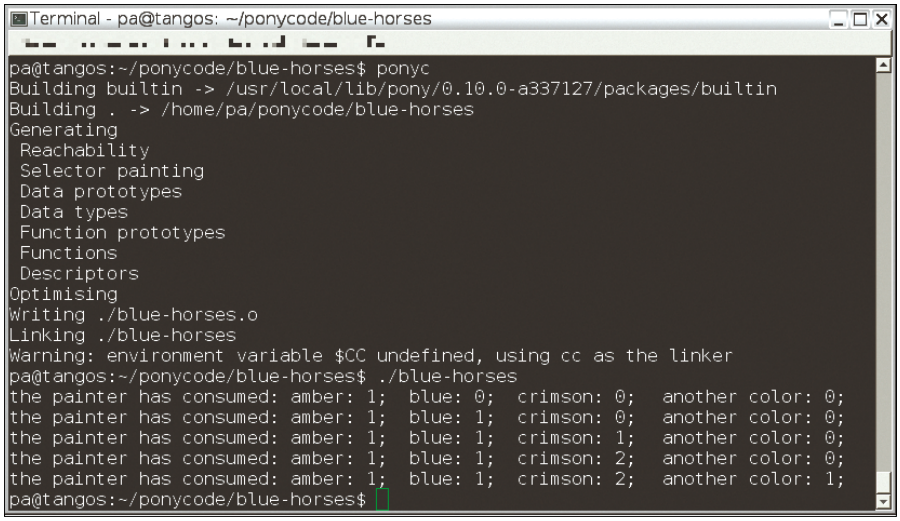


Figure 3: Pony compiles successfully (top), and `blue-horses` reveals paint consumption (bottom).

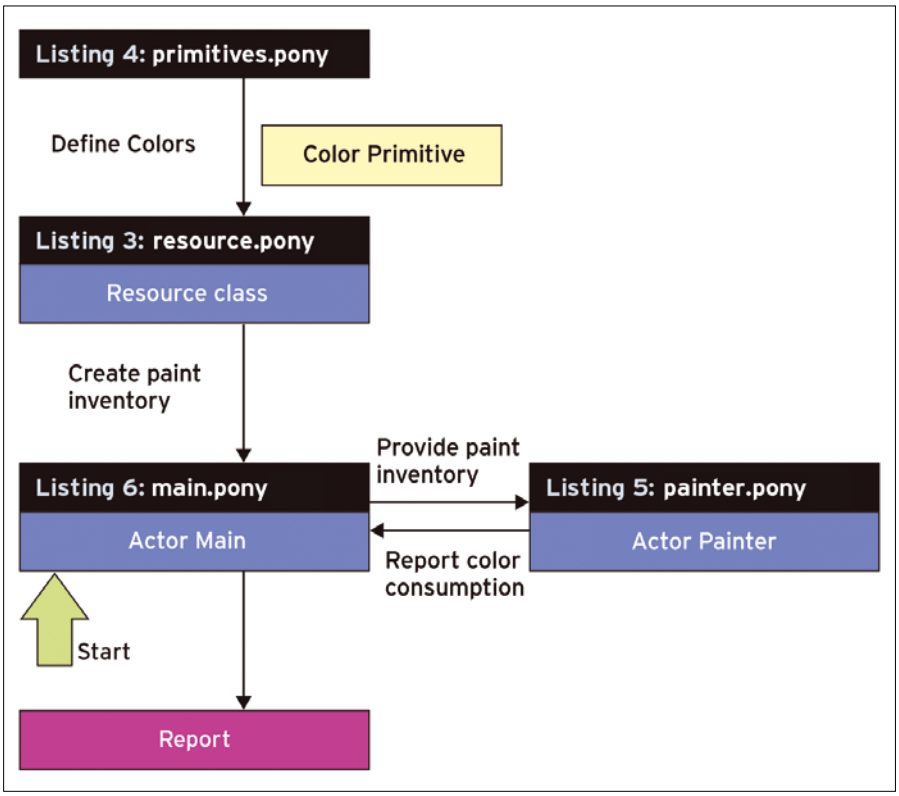


Figure 4: The actor `Painter` processes the quantity of paint and then sends a report on consumption to `main.pony`.

LISTING 3: resource.pony

```
01 class Resource
02   let name: Paint
03   var _amount: U64 = 0
04
05   new create(name': Paint) =>
06     name = name'
07
08   fun level(): U64 =>
09     _amount
10
11   fun ref fill() =>
12     _amount = _amount + 1
```

LISTING 4: primitives.pony

```
02 primitive Amber
02 primitive Blue
03 primitive Crimson
03 primitive Other
05
06 type Color is (Amber | Blue | Crimson | Other)
07
08 primitive Properties
09   fun list(): Array[Paint] =>
10     [Amber, Blue, Crimson, Other]
11
12   fun name(x: Paint): String =>
13     match x | Amber => "amber" | Blue => "blue" | Crimson => "crimson" else "another color" end
```

in line 1, and line 2 declares the public field `name` of the user-defined `Paint` type, which line 6 in Listing 4 again picks up. Thanks to the `let` keyword, the field can only be written to once, in contrast to a field initiated as `var`. The class stores the paint types and the respective stock.

Line 3 of Listing 3 defines the private `_amount` field as an integer data type and sets its value to `0`. Private field names are prefixed by an underscore, and only the owning class instance can access them. The constructor of the `Resource` class introduces the `new` keyword in line 5. The parameter list follows in `create()`. The `name'` variable field is a pointer to an object of the `Paint` type. The next line then saves the `name` field permanently as an instance of the class.

The `fun` keyword introduces the method declarations for `level()` and `fill()` (lines 8 and 11). If the parameter list is followed by a colon, the type of the return value follows. For `level()`, it is integer type `U64` from the private `_amount` field.

The value of the expression in the last line of a method is usually its return value, although the use of `return` lets developers close methods prematurely. The `fill()` method (line 11) increments the value of `_amount` by 1.

Because of the `ref` specification in line 11, the method requires write access for a pointer to an instance of the `Resource` class. These rights are granted by reference capabilities, which I talk about later; however, before I get to that, I'll look at primitives, which are similar to classes.

Primitives

Primitives are classes without fields. Only one instance exists for them, much like `none` and `null` in other languages. Listing 4 begins by declaring the methodless primitives `Amber`, `Blue`, `Crimson`, and `Other` symbolically as the paint types used. Line 6 combines the four primitives using `type` to create the `Paint` alias, `Color`.

In contrast, the `Properties` primitive group functions in the form of methods – more specifically, the two helper methods `list()` and `name()`. The first returns all the paints in a field; the second returns the name of the paint as a string for the primitive passed to it (`x`). The selection in line 13 uses the `match` expression to identify the paints. More composite data types can be found in the *collections* package in Pony's standard library, in the form of `set` and `map`.

Reference Capabilities

Reference capabilities (Table 2), which Pony uses to prevent data races, are a special

TABLE 2: Reference Capabilities

Type	Sendable	Write Permission	Read Permission
iso	Yes*	Exclusive	Exclusive
trn	No	Exclusive	Shared
ref	No	Shared	Shared
val	Yes	No	Shared
box	No	No	Shared
tag	Yes	No	No

*Yes, but only after deleting the sender.

feature that programmers use to avoid setting multiple pointers to a variable object with two or more actors.

Using reference capabilities and dealing with the resulting effects can initially cause some difficulty; programmers will probably need a ramp-up period. The expression `var picasso: Painter iso`, for example, would declare the reference `picasso`, which points to an object of type `Painter`. Thanks to the `iso` capability, it receives exclusive write and read access for the object. Because `picasso` is the only reference to the object, it can be sent safely to another actor after `picasso` is deleted.

To write meaningful programs, Pony also needs the other capabilities from Table 2. Pony sends objects of the `val` capability to all actors, much like a broadcast, because these objects are immutable (read but no write permissions). The `trn` and `box` types provide for data processing in the actor. The `ref` capability acts as a standard for variables and behaves like a pointer in other programming languages; `tag` actors protect against direct access.

Communication in the actor model relies on calling behaviors. The Rust language also limits pointers to avoids data races. Its user rights [8] only allow a reference with exclusive write and read access for each object. Developers grant the exclusive right to write useful programs.

Actors

Actors orchestrate the Pony application. Once started, they wait for asynchronous messages, evaluate the data supplied in accordance with the integrated behaviors, and send the results to other actors. Listing 5 shows the `Painter` actor from the sample application that consumes paints and reports consumption to the `Main` actor. An actor is similar to a class but also saves the behaviors referred to earlier.

The constructor of the `Painter` actor accepts a reference from the `Main` actor in line 5 (the one from Listing 6, line 6) and stores it in the `listener` field. The `history` field (Listing 5, line 3) keeps a record of all `Resource` objects sent via `paint`. Lines 8 to 15 define the `paint` behavior. It also requires an object of type `Resource`. If the level of the resource in line 9 is greater than zero, the `history` field saves another reference when the `push()` method in line 10 is called. Because the `res` reference has exclusive rights (thanks to the `iso` capability), `push()` first uses `consume res` to delete `res`. The next line sends the output to the `stats()` method of line 17 by sending the reference stored in `listener` to the `notify` behavior of actor `Main` (Listing 6, line 12).

Continuing with Listing 5, the `stats()` method iterates in lines 19 to 25 over all paints from the primitives in Listing 4 in a loop defined by the return value of the `list()` method from the same listing.

Internally, loops in Pony do not work directly with fields, but with their iterator objects. Listing 5 generates an object by appending it to `.values()` (line 21). For each primitive, the run-time variable `x` iterates across all received resources. Line 22 totals up the quantity of paint consumed using the `level()` method from Listing 3. The summation uses an `if` expression for filtering. For all non-matching primitives from the history, the `else` branch adds `0` to the sum.

Another decision also needs to be made: If the paint resource `res` in line 9 is empty, the program processes the `else` branch from line 12. The `res` variable has the `iso` reference capability to comply with the required write and read rights from the `fill()` method declaration, which is stated as `ref` (Listing 3). The call in Listing 5, line 13, thus increments the private `_amount` field by one. Then, `paint` sends `res` recursively to itself but deletes `res` once again in line 14 with the help of `consume`.

LISTING 5: painter.pony

```
01 actor Painter
02   let listener: Main
03   var history:  Array[Resource box] = Array[Resource box]
04
05   new create(listener': Main) =>
06     listener = listener'
07
08   be paint(res: Resource iso) =>
09     if res.level() > 0 then
10       history.push(consume res)
11       listener.notify(stats())
12     else
13       res.fill()
14       paint(consume res)
15     end
16
17   fun stats(): String =>
18     var msg: String = "the painter has consumed: "
19     for x in Properties.list() do
20       var sum: U64 = 0
21       for y in history.values() do
22         sum = sum + if y.name is x then y.level() else 0 end
23       end
24       msg = msg + Properties.name(x) + ": " + sum.string() + ";  "
25     end
26     msg
```

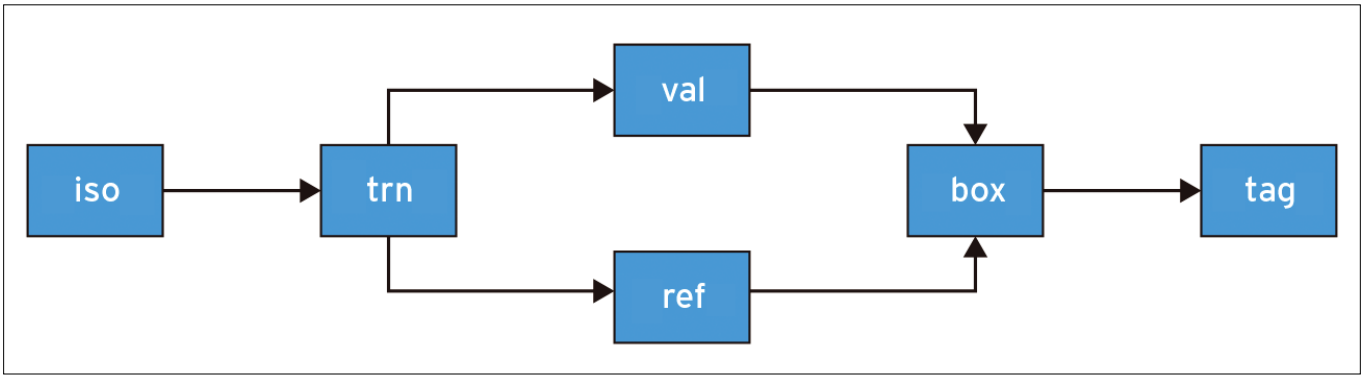


Figure 5: Reference capabilities can be replaced transparently in the direction of the arrow. The subtype is to the left of the arrow in each case.

Subtyping

Listing 5 shows yet a further incompatibility in capabilities: The `history` declaration stipulates `box` (line 3), whereas in line 10, `res` attempts to insert an `iso` reference using `push()`.

Of course `consume` deletes `res`; however, without the capability references, you could not write a meaningful program in Pony. Figure 5 shows the possible replacements that work in the direction of the arrow. For example, an `iso` can replace any other capability, but `trn` can only be used for `val` or `ref`, because `trn` permits additional read references of the `box` capability, which would contradict the exclusive character of `iso`. However, `val` can be replaced by `trn` after a `consume` removes the last writing reference of `trn` type.

If you replace `ref`, you can allow other writing references without problems. The `box` and `tag` capabilities are not critical for `val` and `ref` because the latter cannot write. A blog post by developer John Mumm provides a more detailed explanation [9].

LISTING 6: main.pony

```
01 actor Main
02   var env: Env
03
04   new create(env': Env) =>
05     env = env'
06     let picasso: Painter = Painter(this)
07     for x in [Amber, Blue, Crimson, Other].values() do
08       let res: Resource iso = recover Resource(x) end
09       picasso.paint(consume res)
10     end
11
12   be notify(str: String) =>
13     env.out.print(str)
```

Blue Horses

The `main.pony` file shown in Listing 6 is the entry point in the example program; in the tradition of C, Pony uses `main` as the identifier for the program start point.

Like the familiar `main()` function in C or the `main()` method in Java, Pony defines an actor. In line 4, the actor's constructor accepts a reference to the executing environment of the program as an object of type `Env`. The next lines save the reference in the field with the name `env` in the `Main` instance.

Line 6 creates the actor of type `Painter`, as defined by Listing 5. The constructor uses `this` to receive simultaneously a reflexive reference to the `Main` actor. The `for` loop in lines 7 to 10 iterates over the field with the primitives `[Amber, Blue, Crimson, Other]` or, preferably, its iterator object.

On the basis of field values, line 8 generates an object of type `Resource`, as defined in Listing 3, and stores the reference in the run-time variable `x`. The expression `recover Resource(x) end` introduces a mechanism for arbitrarily setting a capability. In the present form, `recover` generates a reference of type `iso`.

Line 9 transfers the `resource` to the `Painter` actor. A further deletion process, courtesy of `consume res`, occurs before sending. As mentioned earlier, `Painter` finally calls the `notify` behavior in line 12. The message is a `String` type and ends up in the shell thanks to the `env` object.

Conclusions

Thanks to Pony, programmers can write what looks to be secure, high-performance code for concurrent programs. It is free of data races and deadlocks and follows the style of object-oriented languages. The actor model should not cause any trouble, but the difficulty of understanding capabilities could deter many users. The matter is even more complex when developers need to consider the resulting effects of a collection of several capabilities.

If the complexity does not deter you, you can improve the quality of your multithreaded programs; moreover, Pony can be docked on existing projects by integrating C code via the Foreign Function Interface (FFI). ■■■

INFO

[1] Pony: *http://ponylang.org*

[2] Actor model: *http://www.brianstorti.com/the-actor-model/*

[3] Capabilities: *http://man7.org/linux/man-pages/man7/capabilities.7.html*

[4] Rust: *http://rust-lang.org*

[5] Java 9 plans: *http://openjdk.java.net/projects/jdk9/spec/*

[6] Dynamic classes (citation 7): *https://en.wikipedia.org/wiki/Comparison_of_programming_languages_by_type_system#cite_note-7*

[7] Pony on GitHub: *https://github.com/ponylang/ponyc*

[8] The concept of ownership in Rust: *https://doc.rust-lang.org/book/ownership.html*

[9] Sharing reference capabilities: *http://jtfmumm.com/blog/2016/03/06/safely-sharing-data-pony-reference-capabilities/*

7 Years of ADMIN on One DVD

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

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

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

ORDER NOW!

shop.linuxnewmedia.com

MakerSpace

A Rasp Pi wireless access point

Wide Reach

Set up a wireless access point with a Raspberry Pi 3, Ubuntu Core, and snaps. *By Ferdinand Thommes*

Router coverage gaps often have different causes, which repeaters and access points (APs) can remedy. A repeater usually connects to the router over WiFi and amplifies the signal into areas where the router alone is not sufficient, whereas an AP wired to the router by cable sets up a private WiFi network with its own network identifier (SSID). The AP therefore provides additional access to the local network.

A highly portable Raspberry Pi is ideal for setting up a small and cheap WiFi AP suitable for many applications. For example, you could stretch a network into the back garden or provide Internet to an awkwardly located conference room.

The easiest route is to use a Raspberry Pi 3 (RPi3), which already has a WiFi module. Previous models can be prepared for the new task with a dongle, available for just a few dollars. Even the Rasp Pi 3 could benefit from a WiFi stick, because the internal connections of the installed module do not deliver the performance of a good dongle.

In this article, you'll see how to set up a wireless AP, and then I will show you how to provide an additional entry into your local area network through the integration of Nextcloud on an external disk connected to the Rasp Pi.

Good Choice

Not all WiFi sticks work with the Rasp Pi, and those that do work often cannot be used without loading an additional driver. The Edimax and TP-LINK WN725N models are well known to be problem-free on a Pi. Whichever dongle you choose, make sure it supports AP mode. A detailed list of supported dongles, including their properties, can be found on the Embedded Linux wiki [1].

Many of the listed models are powered by the Rasp Pi USB socket. Because the dongle's power consumption depends on its load, among other things, you would be more secure when working with larger amounts of data or numerous accesses by connecting the WiFi stick to an active USB hub.

In the example here, the AP is based on Ubuntu Core. If you want to follow the steps here, you need an operational Rasp Pi connected to the router by Ethernet and reachable via WiFi.

An Account with an Advantage

If you don't already have an Ubuntu One account, you'll need to sign up [2]. This step is important for security, as demonstrated later. After logging in, select the *SSH keys* item and upload a public key, which you will need for an SSH connection to Ubuntu Core. If necessary,

```
Enter passphrase for key '/home/devil/.ssh/id_rsa':
Welcome to Ubuntu Core 16 (GNU/Linux 4.4.0-1030-raspi2 armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Welcome to Snappy Ubuntu Core, a transactionally updated Ubuntu.

 * See https://ubuntu.com/snappy

It's a brave new world here in Snappy Ubuntu Core! This machine
does not use apt-get or deb packages. Please see 'snap --help'
for app installation and transactional updates.
```

Figure 1: SSH access is only possible with a previously created SSH key.

create a new SSH key as root with the command:

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

In the meantime, download the Ubuntu Core image [3]. After unpacking the archive, it expands from slightly more than 300MB to nearly 600MB of disk space.

Ubuntu Core was chosen because it is compact and doesn't have a graphical interface, which would be superfluous on an AP. The system includes the kernel and bootloader from Ubuntu's alternative Snap package system [4] and updates itself. If something goes wrong when updating, the system can be restored automatically or manually to a previous state [5].

Automatic Updates

Automatic updates, especially, are worth their weight in gold, because an AP works quietly in the background, and you could easily forget to update it. Therefore, take the safe and maintenance-free approach, which also allows integration of other Snap applications (e.g., a UPnP media server or openHAB home automation software).

Creating a WiFi hotspot is quick and easy. Start by moving the Ubuntu Core images onto an SD card:

```
$ xzcat <image file> | ➥
  sudo dd of=<device file> bs=8M
```

Use the correct name for the image file and enter the identifier of the device file according to your local setup. To find the correct name of your block device, enter the

```
lsblk -l
```

command. Next, insert the SD card into the appropriate slot on the Rasp Pi and connect the Ethernet cable, monitor, keyboard, and power supply. Shortly after powering on, the computer will prompt you to press the Enter key to start the installation, which could take several minutes.

To continue, configure `eth0` as the default device. Working over WiFi at this point is not possible, because of an error in the installer. The system then prompts you to enter the email address of your Ubuntu account. If you're using a keyboard with an AltGr key, note that it is not implemented correctly, so if you need to use it, type the Alt sequence for the required character with the Num Lock key activated [6] (e.g., Alt + 6-4 for the @ symbol).

After the software has checked the account, you will receive an SSH address to connect to Ubuntu Core. From this point, you can do without the screen and keyboard on the Rasp Pi; just log in from a PC with the specified SSH address (Figure 1).

```
  -h, --help     Show this help message

Available commands:
  abort        Abort a pending change
  ack          Adds an assertion to the system
  alias        Enables the given aliases
  aliases      Lists aliases in the system
  buy          Buys a snap
  change       List a change's tasks
  changes      List system changes
  connect      Connects a plug to a slot
  disable      Disables a snap in the system
  disconnect   Disconnects a plug from a slot
  download     Downloads the given snap
  enable       Enables a snap in the system
  find         Finds packages to install
  get          Prints configuration options
  help         Help
  info         show detailed information about a snap
  install      Installs a snap to the system
  interfaces   Lists interfaces in the system
  known        Shows known assertions of the provided type
  list         List installed snaps
  login        Authenticates on snapd and the store
```

Figure 2: Familiarize yourself with Ubuntu's new Snap packaging system by calling the help switch.

Snapped Open

Snap is used for additional administration. Use the `snap --help` command for more information on administering snaps (Figure 2). However, before you install snaps to complete the AP, assign a new host name (in this example, *pi3*) and set the time zone correctly,

```
$ sudo hostnamectl set-hostname pi3
$ sudo timedatectl ➥
  set-timezone America/New_York
```

then relaunch the system. After logging in again via SSH, install the *wifi-ap* snap and then call the script for interactive configuration,

```
$ snap install wifi-ap
$ sudo wifi-ap.setup-wizard
```

which requires some manual intervention.

In the Wizard, first assign an SSID and, in this case, a password. If you want to set up a public hotspot, respond that you do not want to specify a password. When asked for the IP address, enter *192.168.1.1* or a custom network address. After further questions, which you may answer in the negative (*n*), if appropriate, enable the AP in the last response. The setup is now complete (Figure 3).

```
fe-thomm@pi3:~$ sudo wifi-ap.setup-wizard
Automatically selected only available wireless network interface wlan0
Which SSID you want to use for the access point: ubucore
Do you want to protect your network with a WPA2 password instead of staying open for everyone? (y/n) y
Please enter the WPA2 passphrase: charlest0n
Insert the Access Point IP address: 192.168.1.1
How many host do you want your DHCP pool to hold to? (1-253) 10
Do you want to enable connection sharing? (y/n) y
Which network interface you want to use for connection sharing?
Available are sit0, eth0: eth0
Do you want to enable the AP now? (y/n) y
In order to get the AP correctly enabled you have to restart the backend service:
 $ systemctl restart snap.wifi-ap.backend
2017/07/02 05:48:26 dhcp.range-start=192.168.1.2
2017/07/02 05:48:26 share.network-interface=eth0
2017/07/02 05:48:26 wifi.ssid=ubucore
2017/07/02 05:48:26 wifi.address=192.168.1.1
2017/07/02 05:48:26 wifi.netmask=ffffff00
2017/07/02 05:48:26 share.disabled=false
2017/07/02 05:48:26 disabled=false
2017/07/02 05:48:26 wifi.security=wpa2
2017/07/02 05:48:26 wifi.security-passphrase=charlest0n
2017/07/02 05:48:26 dhcp.range-stop=192.168.1.11
Configuration applied succesfully
```

Figure 3: The automated procedure saves you from manual configuration at various points.

```
fe-thomm@pi3:~$ sudo nano /etc/systemd/system/media-usbdisk.mount
sudo: nano: command not found
fe-thomm@pi3:~$ snap install nano
error: cannot install "nano": snap not found
fe-thomm@pi3:~$ sudo vim /etc/systemd/system/media-usbdisk.mount
sudo: vim: command not found
fe-thomm@pi3:~$ printenv | grep EDITOR
fe-thomm@pi3:~$ sudo editor /etc/systemd/system/media-usbdisk.mount
fe-thomm@pi3:~$ cat /etc/systemd/system/media-usbdisk.mount
[Unit]
Description=Mount USB Disk

[Mount]
What=/dev/sda2
Where=/media/usbdisk
Options=defaults

[Install]
WantedBy=multi-user.target

fe-thomm@pi3:~$ sudo systemctl daemon-reload
fe-thomm@pi3:~$ sudo systemctl enable media-usbdisk.mount
Created symlink from /etc/systemd/system/multi-user.target.wants/media-usbdisk.mount
to /etc/systemd/system/media-usbdisk.mount.usbdisk.mount
fe-thomm@pi3:~$ sudo systemctl start media-usbdisk.mount
```

Figure 4: Only the Vi editor is available for creating a systemd USB mount unit.

Integrating the USB Disk

Installing the Nextcloud file server should be trouble-free with Ubuntu Core. To prepare your system for use with Nextcloud, connect a USB memory stick or hard drive to the AP. You can recognize the correct device by its size in the `lsblk -l` output. Alternatively, call

```
sudo dmesg | tail
```

after connecting, and glean the information from its output.

Once you know the mass storage identifier, create a USB mount unit called `media-usbdisk.mount` in the `/etc/systemd/system/` directory (Figure 4). Add the contents of Listing 1 to this file by adjusting the name of the device file accordingly.

The `Unit` section integrates the external hard drive, but first you have to make it known to the system, so you can use the mass storage:

```
$ sudo systemctl daemon-reload
$ sudo systemctl ➥
  enable media-usbdisk.mount
$ sudo systemctl ➥
  start media-usbdisk.mount
```

To continue, install Nextcloud and allow the snap to interact with the hard drive:

```
$ snap install nextcloud
$ snap connect ➥
  nextcloud:removable-media
```

Now you should give Nextcloud a few minutes while the configuration runs in the background. After a coffee break, create a self-signed certificate,

```
$ sudo nextcloud.enable-https ➥
  self-signed
```

which is sufficient for domestic use. Alternatively, integrate an existing certificate, or create one from Let's Encrypt (Figure 5) with the command:

```
$ sudo nextcloud.enable-https ➥
  lets-encrypt
```

However, you must first prepare the Apache web server – a subject in itself.

```
fe-thomm@pi3:~$ sudo nextcloud.enable-https lets-encrypt
In order for Let's Encrypt to verify that you actually own the
domain(s) for which you're requesting a certificate, there are a
number of requirements of which you need to be aware:

1. In order to register with the Let's Encrypt ACME server, you must
   agree to the currently-in-effect Subscriber Agreement located
   here:

       https://letsencrypt.org/repository/

   By continuing to use this tool you agree to these terms. Please
   cancel now if otherwise.

2. You must have the domain name(s) for which you want certificates
   pointing at the external IP address of this machine.

3. Both ports 80 and 443 on the external IP address of this machine
   must point to this machine (e.g. port forwarding might need to be
   setup on your router).

Have you met these requirements? (y/n) ▮
```

Figure 5: Using a certificate from Let's Encrypt requires some additional work.

LISTING 1: USB Mount Unit

```
[Unit]
Description=Mount USB Disk

[Mount]
What=<device file>
Where=/media/usbdisk
Options=defaults

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

Figure 6: On the laptop, the established ubucore connection is available in the NetworkManager.

Everything is now ready to connect to the AP and to launch Nextcloud. Begin by looking on any computer or other WiFi-enabled device at the list of networks offered and connect to the AP (Figure 6).

Next, call *https://<Nextcloud IP address>* (i.e., the address you assigned to Nextcloud) in your browser. The final step for storing data on the external drive is to select the External Storage app in the Nextcloud GUI and then point */media/usbdisk/* in the admin back end (Figure 7).

Conclusions

Rarely has it been so easy to set up a wireless AP in a private cloud. The development of the AP itself takes about half an hour; including the Nextcloud setup takes about an hour. The procedure described here is ideal for devices that are inconspicuous in everyday life and provide their service out of sight.

The AP remains secure because of the encrypted SSH connection to the account and the Ubuntu Snappy Core automatic updates – with the option of returning to the old state in the event of problems.

The integration of Nextcloud on an external disk connected to a Rasp Pi only represents one of the many options for this AP. If necessary, you can easily set up a hot spot, guest access, or a center for home automation on the Raspberry Pi AP. ■■■

INFO

[1] WiFi dongles: *http://elinux.org/RPi_USB_Wi-Fi_Adapters*

[2] Ubuntu One account: *https://login.ubuntu.com*

[3] Ubuntu Core: *http://releases.ubuntu.com/ubuntu-core/16/ubuntu-core-16-pi3.img.xz*

[4] Snap: *https://wiki.ubuntuusers.de/snap/*

[5] "Snaps and Flatpaks" by Ferdinand Thommes, *Ubuntu User*, issue 30, Fall 2016, pg. 32, *http://www.ubuntu-user.com/Magazine/Archive/2016/30/The-package-formats-Flatpaks-and-Snaps*

[6] Alt keyboard sequences: *https://tools.oratory.com/altcodes.html*

AUTHOR

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

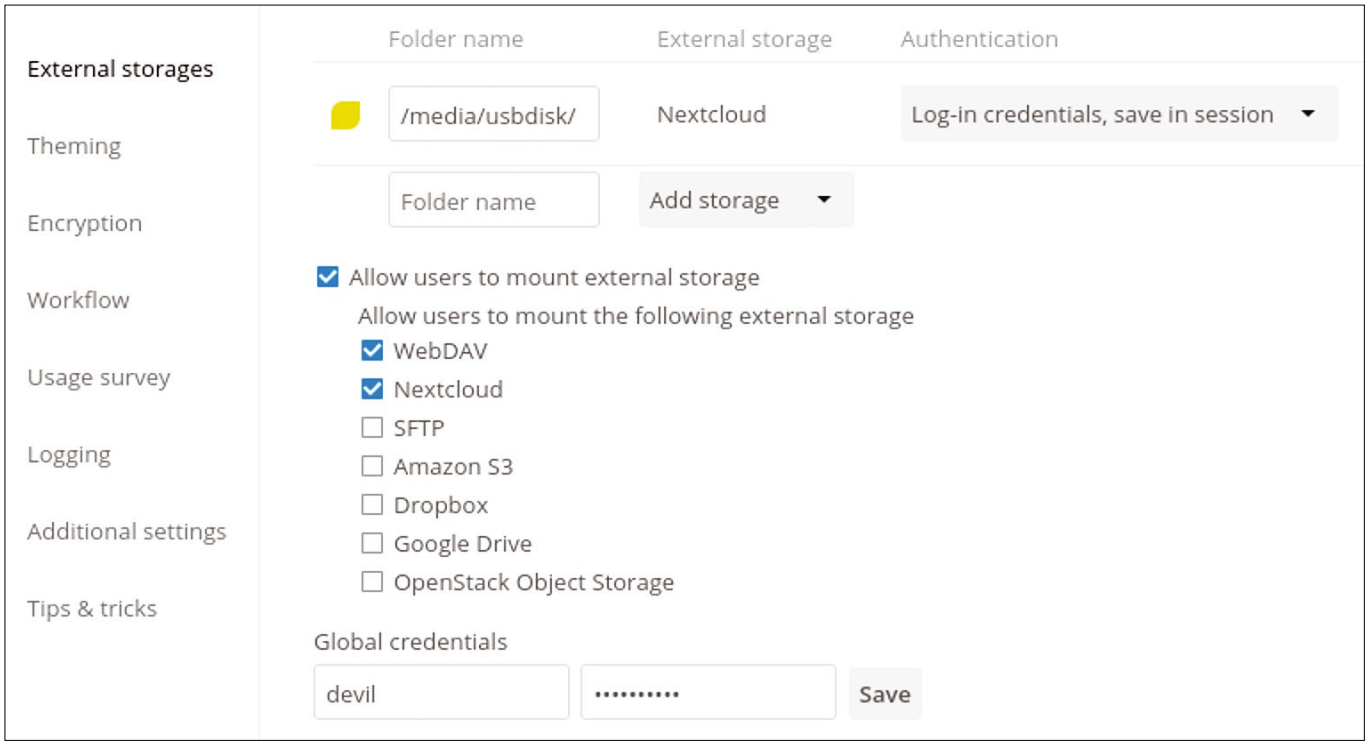


Figure 7: The External Storage app lets you store what you have connected to the Rasp Pi AP on the external drive.

MakerSpace

A Rasp Pi HAT for clustering Pi Zeros

More than Zero

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

When I started in high-performance computing (HPC), the systems were huge, hulking beasts that were shared by everyone. The advent of clusters allowed the construction of larger systems accessible to more users. I always wanted my own cluster, but with limited funds, that was difficult. I could build small clusters from old, used systems, but the large cases took up a great deal of room. The advent of small systems, especially single-board computers (SBCs), allowed the construction of small, low-power, inexpensive, but very scalable systems.

Arguably, the monarch of the SBC movement is the Raspberry Pi [1]. It is now the third best selling computer of all time [2], overtaking the Commodore 64 and behind the PC and the Mac, and has sparked a whole industry around small, inexpensive, low-power but expandable computers that can be used for anything from sensors in the field, to desktops, to retro game consoles, and even to experiments on the International Space Station. The top-end Raspberry Pi, the Raspberry Pi 3 (RPi3), is about $35, and the introduction of the Raspberry Pi Zero (Pi Zero) in 2015, set the low-end price of $5.

People have been building clusters from Raspberry Pi units, starting with the original Raspberry Pi Model A, ranging from two to more than 250 nodes [3]. That early 32-bit system had a single core running at 700MHz with 256MB of memory. You can build a cluster of five RPi3 nodes [4] with 20 cores connected by a Gigabit Ethernet switch for about $300, including a case and case fan.

Fairly recently, a company created a Hardware Attached on Top (HAT) [5] add-on board that you can add to a single "host" RPi2 or 3. It has four micro-USB slots that each connect to a Pi Zero. It provides both power and a network between the Pi Zeros and the host node. The ClusterHAT [6] fits into the GPIO pins on the host (master node) and accepts up to four Pi Zeros in mini-USB ports.

The ClusterHAT kit is a little over $25 and includes the HAT board, four standoffs, a handy USB cable, and some plastic feet if you want to put them on the bottom of your host node (Figure 1). Putting everything together is very easy, and you can watch a video [7] on the ClusterHAT website to show how it's done.

After threading the USB cable between the HAT and the RPi3, attaching the HAT, and snapping the Pi Zero boards into the HAT, you should have something that looks like the Figure 2. Next, attach a keyboard, a mouse, an external power supply, and a monitor (Figure 3). Notice that the Pi Zeros are powered on in this image (i.e., the lights near the boards are lit).

The RPi2 or 3 needs a good power supply capable of 2 to 2.5A and at least one microSD card for the master node. You can put a card in each Pi Zero, if you want, or you can NFS boot each one (although that's a little experimental). I put a 16GB microSD card into each of the five Raspberry Pis.

The ClusterHAT site has created Raspbian Jessie-based [8] software images that have been configured with a few simple tools for the ClusterHAT. Jessie is a little different from past Raspbian versions, and the biggest difference that affects the ClusterHAT is the use of DHCP by default.

For this article, the target cluster configuration uses an RPi3 as the master node and Pi Zeros in the ClusterHAT as compute nodes. The master node will be an NFS server, with `/home` and `/usr/local`, for the four Pi Zeros. Additionally, the cluster will use passwordless SSH and `pdsh` [9], a high-performance, parallel remote shell utility. MPI and GFortran will be installed for building MPI applications and testing.

At this point, the ClusterHAT should be assembled and the operating system (OS) images copied to the microSD cards for the five nodes. The next steps are to boot the images for the first time on the RPi3 and configure them to meet the previously mentioned target configuration.

Master Node

Because the master node effectively controls the cluster, getting the OS configuration correct is important. Fortunately,

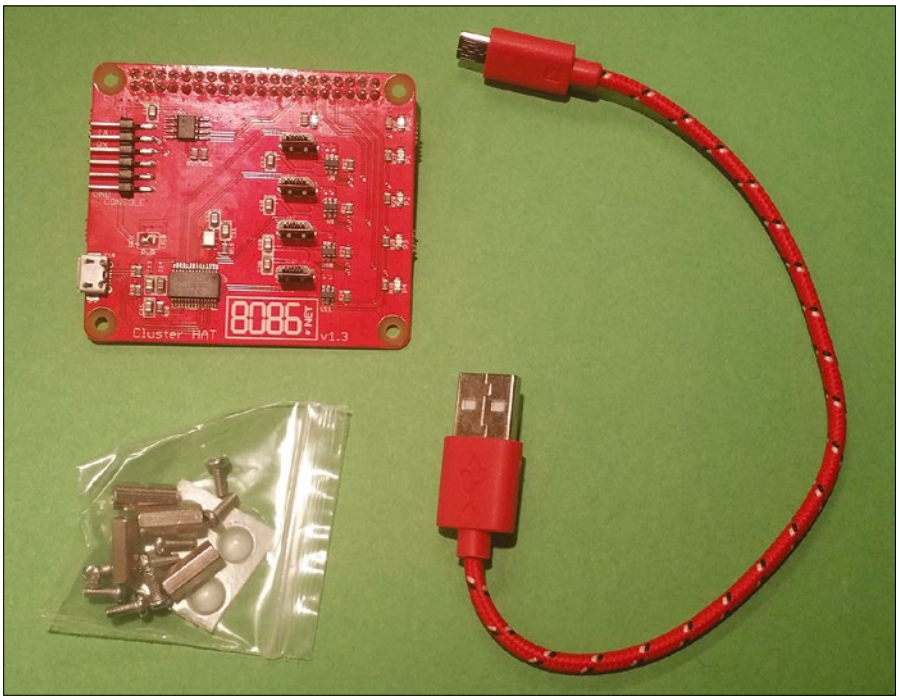


Figure 1: ClusterHAT components.

only a small number of changes need to be made to the image provided on the website.

The first step is to boot the master node (RPi3) with its microSD card. Be sure it is plugged in to your local network and can access the Internet. After the RPi3 boots, you should be in the Pixel desktop [10] (Figure 4). A few "classic" configurations are called for at this point with the help of the `raspi-config` [11] command:

- Expand the storage to use the entire microSD card
- Change the default password for the *pi* account
- Enable SSH (it is no longer enabled by default in Raspbian)
- Switch the keyboard to a US keyboard (by default, Raspbian uses a UK keyboard)

After configuring, you need to install the NFS server packages on the RPi3 (master node):

```
$ apt install nfs-common ➥
  nfs-kernel-server
```

Next, the NFS exports, a list of filesystems to be exported, needs to be created. The file `/etc/exports` should be created or edited to include the following:

```
/home        *(rw,sync,no_subtree_check)
/usr/local   *(rw,sync,no_subtree_check)
```

Notice that the filesystems are exported globally with the "*" wildcard. Although this is usually not a good idea, because anyone could mount the filesystems, I almost always run the system with no Internet access, so I'm not too worried.

To make things safer, you can specify an IP range that can mount the filesystems.

To ensure that the NFS server starts when the RPi3 is booted, run the following commands:

```
$ sudo update-rc.d rpcbind enable
$ sudo /etc/init.d/rpcbind start
$ sudo /etc/init.d/nfs-kernel-server ➥
  restart
```

Note that these commands need to be run whenever the master node is rebooted. The filesystems can be exported and checked to make sure they are actually exported:

```
$ sudo exportfs -ra
$ sudo export's
/home
/usr/local
```

Next, SSH is configured so that passwordless logins can be used on the cluster. Many tutorials on the web explain how to accomplish this. After SSH, both GFortran, the GCC Fortran compiler, and MPICH, a high-performance implementation of the Message Passing Interface (MPI) standard, are installed using `apt`:

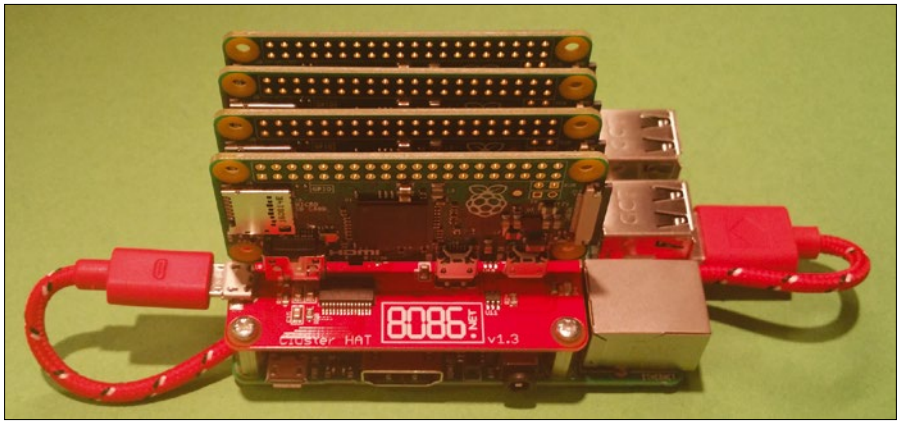


Figure 2: ClusterHAT and Pi Zeros attached to a RPi3.

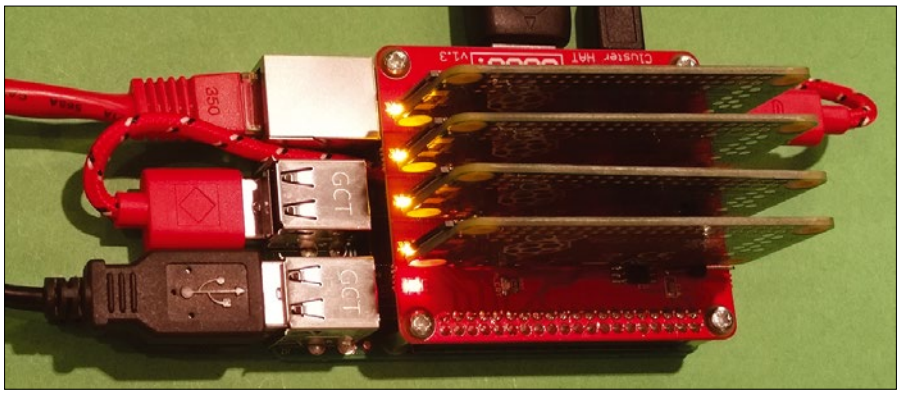


Figure 3: Completed ClusterHAT configuration.

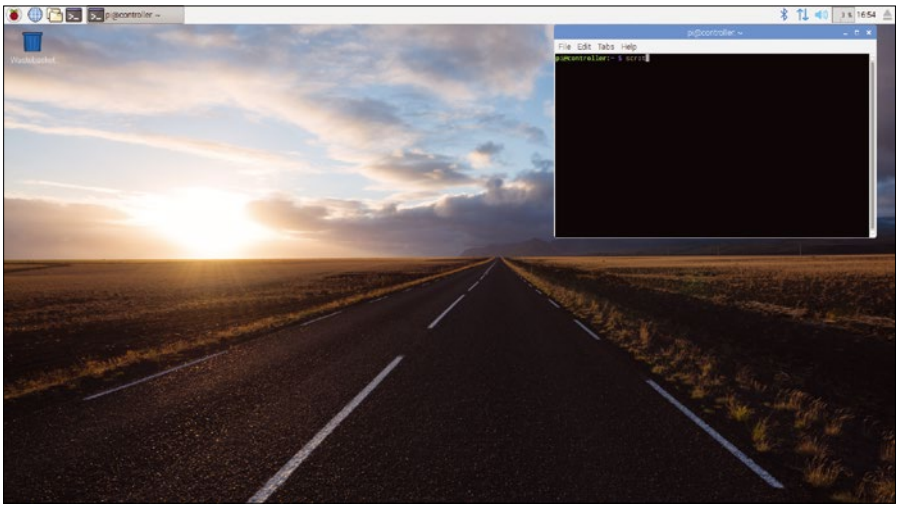


Figure 4: Pixel desktop.

```
$ apt install gfortran mpich
```

The HPC world uses tools that allow commands to run across the entire cluster or a subset of nodes in the cluster. My preferred "parallel shell" tool is `pdsh`. Consult one of my previous articles [12] for directions on how to build, install, and use `pdsh`. I installed `pdsh` for the ClusterHAT in `/usr/local`. A `/home/pi/PDSH` directory was created with a `/home/pi/PDSH/hosts` file that lists the default nodes to be addressed by `pdsh` when no nodes are specified. For the ClusterHAT, the list of nodes is:

```
p1.local
p2.local
p3.local
p4.local
```

As an option, `controller.local` could be added to the list if the master node is to be a default target for commands.

Configuring Compute Nodes

The ClusterHAT site provides several ways to build images for the compute nodes. I took the easier route and downloaded five images – one for the controller (head node or master node), and one each for the four compute nodes – and copied them to a microSD card for each node. By taking this approach, each image could be booted in the RPi3 so that changes could be made before booting the entire cluster.

Unlike the master node, the compute nodes do not boot into the desktop; they just boot to the command line so you can log in to the node.

Booting each image allows you to make some basic changes that are needed on the first boot of a Raspbian system. First, the command `raspi-config` must be used, as discussed for the master node, to extend the filesystem to use the entire microSD card and enable SSH (see the bulleted list above). The third action, changing the password, should be done on all the compute node images; they should have the same password, but not the default, *raspberry*.

To make life easier, I like to have my clusters share a common `/home` for the users and `/usr/local/` for applications shared by the nodes. The ClusterHAT cluster is no exception: I want to mount `/home` and `/usr/local` from the master node to all of the compute nodes. I added the following lines to `/etc/fstab` on all of the compute node images (Listing 1).

Also installed on each compute node are `gfortran` and `mpich`; these were installed on the master node from the Raspbian repositories, so they are not installed in `/usr/local` or `/home`; consequently, they have to be installed on each node.

By default, the Pi Zero nodes are named `p1.local`, `p2.local`, `p3.local`, and `p4.local`. If you look at the ClusterHAT from above, at one end you can see the labels p1, p2, p3, and p4, for the four slots. The master node has the node name `controller.local`.

After setting up the master node, the ClusterHAT, and the compute nodes, it's time for the first boot!

LISTING 1: /etc/fstab Additions

```
controller.pi:/home       /home
nfs    default    0   0
controller.pi:/usr/local  /usr/local
nfs    default    0   0
```

First Boot

Booting is pretty simple: You plug in the HDMI cable to the monitor and then plug in the power cable to boot the RPi3 master node, which should go into the Pixel desktop. The first time I booted the ClusterHAT, I didn't have it plugged in to the network because my router acts as a DHCP server and assigns IPs to the compute nodes. This setup can sometimes cause problems.

Once the master node has booted, it is a good idea to check the node to see if it looks correct – look especially to see that the two filesystems are NFS exported and that `gfortran`, `mpich`, and `pdsh` are functioning.

The ClusterHAT images come with a very useful tool to start and stop the compute nodes. The `clusterhat` tool [13] is a simple Bash script that uses GPIO [14] (General Purpose Input/Output) pin commands to control the power to the compute nodes, allowing you to turn nodes `on` and `off` individually, in groups, or all together and adding a two-second delay between the command for each node. For example, to turn on all of the compute nodes, you run:

```
pi@controller:~ $ clusterhat on all
Turning on P1
Turning on P2
Turning on P3
Turning on P4
pi@controller:~ $
```

People have also taken the spirit of the `clusterhat` tool and created something a little different. For example, `clusterctl` [15] allows you turn the compute nodes on and off, but also lets you determine the status of the node, cut the power, and even run a command across all of the compute nodes.

The first time, it's probably a good idea to boot the cluster nodes one at a time. For example, to boot the first node, run:

```
pi@controller:~ $ clusterhat on p1
Turning on P1
pi@controller:~ $
```

TABLE 1: NPB Results

Test	Class	No. of Cores	Total MOPS (RPi3 Only)	MOPS/Process (RPi3 Only)	Total MOPS (Pi Zeros Only)	MOPS/Process (Pi Zeros Only)	Total MOPS (Pi Zeros + RPi3)	MOPS/Process (Pi Zeros + RPi3)
CG	A	4	198.98	49.75	38.77	9.69	–	–
CG	A	8	–	–	–	–	71.98	9
EP	A	4	25.8	6.45	6.93	1.73	–	–
EP	A	8	–	–	–	–	13.92	1.74
IS	A	4	43.85	10.96	3.99	1	–	–
IS	A	8	–	–	–	–	6.71	0.84
LU	A	4	425.36	106.34	197.88	49.47	–	–
LU	A	8	–	–	–	–	396.22	49.53

Booting nodes one at a time allows each to be checked to make sure everything is installed and has booted properly.

Remember that the master node NFS-exports two filesystems to the compute nodes. Given that the Pi Zeros use a bridged network [16] over USB 2.0, the network performance is not expected to be very good. Therefore, it will take a little longer for the filesystems to mount. One suggestion is to `ping` the node (`ping p1.local`) until it responds. If the filesystems don't mount for some reason, you can use the `clusterhat` tool to turn the node `off` and then `on` again.

After testing each node independently and ensuring that everything works correctly, you can then reboot all of the nodes at once. Now you can test the cluster by running some MPI code on it.

MPI Code Example

I'm not going to cover "benchmarks" on the ClusterHAT, but it is important to illustrate some real MPI code running on the cluster. Rather than run HPL [17], the high-performance Linpack benchmark, and argue over tuning options to get the "best" performance, I find it's better to run the NAS Parallel Benchmarks (NPB) [18], which are fairly simple benchmarks that cover a range of algorithms, primarily focused on computational fluid dynamics (CFD) [19]. They stress the processor, memory bandwidth, and network bandwidth; are easy to build and compile; and come in several flavors, including MPI. Also, different problem sizes or "classes" scale from very small to very large systems.

Because the ClusterHAT is a small cluster, I used only the class A test. In the interest of brevity, I only used the `cg` (conjugate gradient, irregular memory access and communication), `ep` (embarrassingly parallel), `is` (integer sort, random memory access), and `lu` (lower-upper Gauss-Seidel solver) applications with four and eight processors. Four processors included two cases: (1) Pi Zeros only, and (2) RPi3 only. The eight processors case included the RPi3 and the Pi Zeros (a total of eight cores).

For all four applications, performance, measured in millions of operations per second (MOPS), was recorded from the output for the entire MPI group and for each process in the MPI group. These results are tabulated in Table 1.

Summary

The ClusterHAT is one of the most innovative clusters to come along in many years. It's very compact, uses a very small amount of power, runs Linux, and is fairly inexpensive (around $100 for the entire system).

Although it is obviously not designed to be a speed demon, you can use it to learn about common cluster tools and clustering and how to write parallel and distributed software. The ClusterHAT cluster can run Singularity [20] and Docker [21] containers, including resource managers. In a classroom, each student could have a small ClusterHAT cluster on which to run real applications, including AI [22].

The ClusterHAT is one of the coolest HPC pieces of hardware to come out in a long time. ■■■

INFO

[1] Raspberry Pi: *https://www.raspberrypi.org/*

[2] Raspberry Pi sales: *http://www.techradar.com/news/the-raspberry-pi-is-now-the-third-best-selling-computer-of-all-time*

[3] Parallel computing through art: *http://www.seemoreproject.com/*

[4] Five-node Raspberry Pi 3 cluster: *http://climbers.net/sbc/diy-raspberry-pi-3-cluster-2017/*

[5] HATs: *https://www.raspberrypi.org/blog/introducing-raspberry-pi-hats/*

[6] ClusterHAT: *https://clusterhat.com/*

[7] ClusterHAT video: *https://clusterhat.com/setup-assembly*

[8] Raspbian Jesse: *https://www.raspberrypi.org/blog/raspbian-jessie-is-here/*

[9] pdsh: *https://github.com/grondo/pdsh*

[10] Pixel desktop: *https://www.raspberrypi.org/blog/introducing-pixel/*

[11] raspi-config: *https://www.raspberrypi.org/documentation/configuration/raspi-config.md*

[12] "Parallel shell with pdsh" by Jeff Layton, *Linux Pro Magazine*, issue 166, September 2014, pg. 64, *http://www.linuxpromagazine.com/Issues/2014/166/Parallel-Shells*

[13] clusterhat tool: *https://clusterhat.com/setup-control*

[14] GPIO: *https://www.raspberrypi.org/documentation/hardware/raspberrypi/gpio/README.md*

[15] clusterctl: *https://github.com/thagrol/clusterctl*

[16] Bridged network: *https://wiki.debian.org/BridgeNetworkConnections*

[17] HPL: *http://www.netlib.org/benchmark/hpl/*

[18] NPB: *https://www.nas.nasa.gov/publications/npb.html*

[19] CFD: *https://en.wikipedia.org/wiki/Computational_fluid_dynamics*

[20] Singularity: *http://singularity.lbl.gov/*

[21] Docker: *https://www.docker.com/*

[22] AI: *http://mashable.com/2017/06/29/microsoft-puts-ai-on-a-raspberry-pi/*

MakerSpace

The challenge of the free phone

Security Calling

Purism is committed to creating a completely free phone that protects your data, keeps your communications private, and supports your digital rights. *By Bruce Byfield*

Laptops and workstations that are free from the firmware to the operating system are rare, but they do exist, as the Free Software Foundation's Respect Your Freedom certification demonstrates [1]. Completely free phones, however, are still struggling to exist. Jolla makes phones with free software, but the license makes clear that its Sailfish operating system also includes proprietary software [2]. Technoethical is struggling to produce a free phone, but has yet to release one [3], and Canonical's efforts to fund the high-end Ubuntu Edge failed several years ago [4]. Now, however, Purism [5], already known as a successful maker of open hardware laptops, looks on track to produce the first free phone. Having raised more than $1.5 million in a private crowdfunding campaign, Purism is partnering with KDE and Gnome to produce the Librem 5, a phone that is not only free but also private and secure [6].

Purism has become known in the last few years for its high-end laptops, the Librem 11, 13, and 15. These projects have received more than $2.5 million in funding and have been a critical success, as well. Earlier in the company's history, complaints were made that the firmware of these laptops was not free [7], but according to Todd Weaver, Purism's CEO, "we have been shipping Coreboot [a free BIOS] as default on our products for the past nine months and are the most free hardware available" currently. The laptops still have further to travel along the path to total freedom, as indicated by the company's Freedom Roadmap [8], but the point is that progress is being made. The same is going to be true for the Librem 5, which the campaign page describes as "a fully standards-based freedom-oriented system" – but not as a completely free phone, yet

Such a phone is needed, Weaver says, because "the current mobile landscape is a dark, nightmarish place, where two large for-profit corporations control everything: Google Android and Apple iOS. Phones know more about us than any other device on the planet, and there is no ethical alternative. In the server, laptop, desktop, and router space, the world has convenient options, so ethical choices can be made. In the phone market there is no convenient option; Purism's Librem 5 intends to be that option."

Raising the Bar

To be a free phone, the Librem 5 will not use Android, although free versions of Android like Lineage OS (formerly CyanogenMod) do exist. "That is a short-term and short-sighted solution that does not actually improve the future for

Figure 1: A mock-up of what the Librem 5 phone will look like.

digital rights or ethical computing," says Weaver. Free versions of Android typically lag behind proprietary versions, and security in Android as a whole is often lax. To avoid such problems, the Librem 5 will run PureOS, a Debian-based distribution that emphasizes security – a choice that also means that, as with the Ubuntu Touch, the Librem 5 can be expected to act more like a laptop or workstation than other phones (Figure 1).

Moreover, as though building a free phone is not enough, Purism is increasing the challenge by emphasizing user choice and security. Users will be able to install other distributions "with ease," says Weaver. Moreover, besides the innate security advantages of a Linux distribution and the availability of tools like the Tor browser and Privacy Badger, the Librem 5 is designed with numerous other security features. Applications will run in isolation, so that a compromised app will not allow access to the entire system. Additional security will be provided by a VPN service, default IP-based communication, and end-to-end encryption. Unique among phones, the Librem 5 will also separate the CPU from the Baseband and include hardware kill switches for WiFi and Bluetooth, the webcam, microphone, and Baseband.

"It all comes down to control and convenience," Weaver says. "By bundling together a product that protects users by default, running free software where the source code is available into a convenient product, we offer an easy-to-use product that the user controls, not the parent company." Purism, Weaver emphasizes, "is a Social Purposes Corporation [9], not a traditional for-profit corporation," meaning that its self-appointed ethical standards are as important to it as profits.

Gnome and Plasma Mobile

In addition to its privacy and security features, the Librem 5 is partnering with Gnome and KDE to bring these desktops to the phone.

Gnome needs no introduction and is well-suited to mobile devices. In fact, its use of a working and overview desktop seems modeled on mobile interfaces, and the project has already ventured into mobile computing with the Nokia 770, N800, and N900; the One Laptop Per Child project's XO laptop; and FIC's Neo1973 mobile phone. To be used on the Librem 5, a media release by the GNOME Foundation explains that it "plans to enhance Gnome shell and general performance of the system with Purism to enable features on the Librem 5" [10]. So far, however, no other details have been released about the changes to Gnome.

KDE's partnership, though, may be more problematic. Purism will not be including KDE's standard Plasma desktop environment but Plasma Mobile, a recent project that is still in development. According to KDE developer Sebastian Kügler, Plasma Mobile's reference implementation currently runs only on the "LG NexusX, Nexus 5, and, thanks to the Halium project, which is a hardware abstraction to allow Plasma Mobile to run on devices which are currently only supported by Android, the number of supported devices is growing." If all goes as planned, the Librem 5 is probably going to be "the first phone to ship with Plasma Mobile out of the box." Weaver expresses his confidence that Plasma Mobile will be available "within the timeframe needed to get product into users' hands."

Few people have seen Plasma Mobile (Figure 2). No emulation is available, and first-hand experience is limited to those who have installed it on one of the handful of devices it currently supports. However, from the videos available online [11], Plasma Mobile features touchscreen support – obviously – and swiping from the edges reminiscent of both Ubuntu Touch and Plasma Active, KDE's previous attempt to provide a mobile interface. Its development benefits by sharing "more than ninety percent" of its code with Plasma, the subsystem for KDE's graphical interface on the desktop. First introduced in KDE 4.0, Plasma abstracts the interface from the rest of the system, making it easy to exchange interfaces. Unsurprisingly, it has a history of doing just that, not only with Plasma Active, but also with Plasma Netbook, which suggests that readying Plasma Mobile for the Librem 5's proposed shipping date in July 2019 should offer no insolvable problems.

The main advantage of both Gnome and Plasma Mobile over Android and iOS is that they are developed as open source operating systems, which at least theoretically should offer greater privacy and control over data. Both should appeal to individual users and corporations alike with a need to control their data and communicate securely.

By using these two interfaces, the Librem 5 may accomplish the goal of convergence – a single interface for all devices – that Canonical has recently de-emphasized. That would be particularly ironic in the case of Gnome, to which Canonical's Ubuntu distribution has returned while almost stopping convergence development. However, as Kügler points out while talking about Plasma Mobile, convergence may be an idea whose time has finally arrived. "It takes a learning curve for users, and, I think, advancements in technology to bring it to market," Kügler says. "You need rather powerful hardware, the right connectors, and the right hardware components, so it's not an easy end goal. [However], the path to convergence already bears huge benefits, as it means more efficient development, more consistency across different form factors, and higher quality code." The result promises to be a phone that acts more like a laptop or workstation, rather than just an Android clone.

Answering Doubts

The Librem 5 crowdfunding campaign succeeded, but it sparked considerable debate. Although the alliances with Gnome and Plasma Mobile can be taken as strong votes of confidence, the campaign's announcement has sparked debate about whether such an ambitious project can succeed where so many others have failed.

Weaver's response is that Purism can benefit from past mistakes and is in a better position than similar projects. Thanks to crowdfunding and because the Librem 5 is aimed at a relatively narrow market niche, the Librem 5 does not need to sell tens of thousands of phones to be a success. Nor, despite many online criticisms, does it need a large app store to rival Android's Google Play; with access to a distribution's packages, the Librem 5 should offer thousands of apps, at least some of which have versions designed for mobile devices. Furthermore, advances in both computing power and software like Plasma Mobile have made development easier than in the past.

According to Weaver, the project should especially benefit from its clearly defined goals and the fact that "the awareness of privacy and security concerns are at an all-time high." Because of the long time between funding and release, Purism is delaying the release of specifications as long as possible, but even a mid-level phone with the Librem 5's security features would likely have a strong chance of finding a small but appreciative audience.

Another concern is what I call the Arts and Crafts Dilemma, after the idealistic design movement at the start of the Twentieth Century: That is, to deliver small runs of ethical products, manufacturers have to charge high prices that keep their products out of the hands of most buyers. Weaver admits that "we do have to charge prices that are higher than possible if we had huge volumes." However, he adds that "we are comparable in price to other high-end hardware from Apple, Dell, and Lenovo, and our Librem 5 phone is significantly less than some mobile phone offerings." For example, the Apple X is retailing at $999, compared with the Librem 5's $599 price point.

Figure 2: The little-known Plasma Mobile desktop is a variant on KDE's standard Plasma desktop.

In the end, Weaver says, "people fund Purism products because they believe what we believe – that ethical products are important – so we can leverage that to form a beachhead, and expand up into the supply chain, and pass on volume discounts to users in the future. It is a strong, long-term business strategy designed to change the future for the better." It is also an ambitious project, whose emphasis on privacy and security is finally becoming generally appealing. As it evolves, the Librem 5 should go a long way toward proving the practicality of open hardware. ■■■

BRUCE BYFIELD

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

INFO

[1] Respect Your Freedom Certification: *http://www.fsf.org/resources/hw/endorsement*

[2] Sailfish EULA: *https://jolla.com/sailfish-eula/*

[3] Technoethical phones: *https://tehnoetic.com/mobile-devices*

[4] Ubuntu Edge: *https://www.indiegogo.com/projects/ubuntu-edge#/*

[5] Purism: *https://puri.sm/shop/librem-5/#wpneo-tab-description*

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

[7] Accusations: *https://hackerfall.com/story/the-truth-about-purism-why-librem-is-not-the-same*

[8] Freedom roadmap: *https://puri.sm/learn/freedom-roadmap/*

[9] Social Purpose Corporations: *https://en.wikipedia.org/wiki/Social_purpose_corporation*

[10] GNOME Foundation announcement: *https://www.gnome.org/news/2017/09/gnome-foundation-partners-with-purism-to-support-its-efforts-to-build-the-librem-5-smartphone/*

[11] Plasma Mobile videos: *https://www.youtube.com/results?search_query=Plasma+Mobile*

Power users often want more than the usual defaults; they like things smooth and streamlined – partly because anything worth doing is worth optimizing, but also because, well, some people just like tweaking things to make them work better. In fact, that's one of the ways you get to be a power user – by pushing the limits to perfect your user experience and learning along the way.

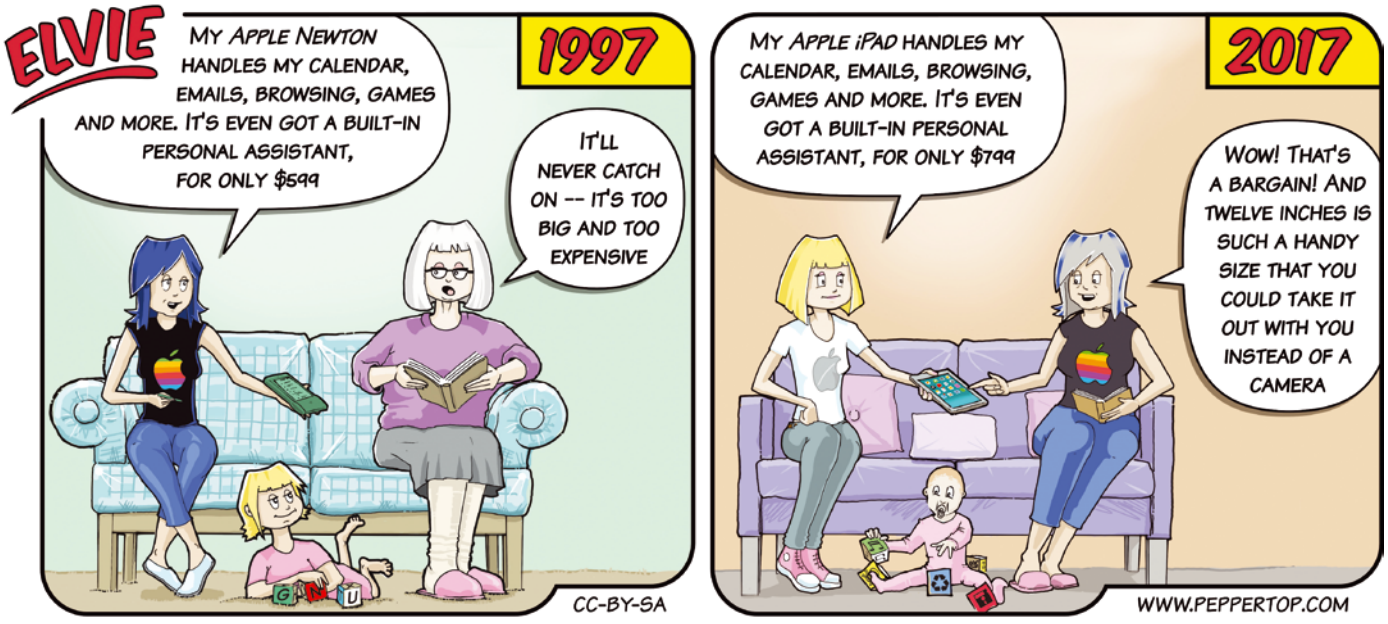
This month's tutorial on Devilspie2 offers lots of play space for tweakers everywhere. This cool tool lets you write scripts to customize the size and location for screen windows. You can even manage workspaces through automated scripts.

Elsewhere inside, we introduce you to Web Security Dojo, a Linux appliance that serves as a teaching tool for hands-on learning about web security and penetration testing.

– Joe Casad

Image © Olexandr Moroz, 123RF.com

LINUXVOICE



NEWSANALYSIS

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

Opinion

Privacy and Re-Decentralization

Learning lessons from the business success of Linux and applying them to privacy. BY SIMON PHIPPS

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

As a society, we're discovering that viewing technology in isolation is insufficient. We really need to have a holistic, joined-up view of how IT works. All over the world as the economy dips, we see authoritarianism on the rise. As technologists, we are culpable in not having foreseen this – in having believed technology is amoral and that we can continue to focus on cool stuff and not worry about what people are going to do with it.

The biggest challenge may be the centralization of personal data – not just facts, but opinions, too. Global data giants are delivering "cloud" services that may be implemented by distributed systems but are centrally controlled. That offers a control point for authoritarians.

We need to find a way to take Google, Facebook, Twitter, and all the rest out of the center of the diagram and put them as nodes in the diagram. It's not that they are inherently bad, but they create a "hostage to fortune" by having so much information and so much control in places where authoritarian governments can, with the stroke of a pen, decide to make it impossible for you to have the freedoms that the UN Convention on Human Rights declares as inviolable.

It's very important that we continue to promote software freedom. It's extremely unlikely that we will have any civil liberties left if software freedom is eliminated. All the same, I don't think software freedom alone is enough for protecting our civil liberties. I'm thus concerned about all our digitally expressed civil rights; I'm concerned about the over-centralization of the web, as well as our overdependence on proprietary technology.

For the firsts steps of re-decentralizing the web, I think we need to focus on privacy. As I wrote in Issue 203, privacy is not just about protecting facts, it's also about protecting the ability to connect facts and deduce context from them. We have privacy when we control the disclosure of ourselves to others.

That's why, with caveats, I think the new General Data Protection Regulation (GDPR) is a force for good. The burden of GDPR on companies is massive. Consider the bureaucracy and administration they have to manage; the architectural challenge for the software that they'll need to develop is massive. Thus companies will prefer not to aggregate nonessential personal information, because the administrative and bureaucratic burden of doing so is too great upon them.

Some fear new restrictions will harm progress, but the lessons of Linux suggest otherwise. The Linux kernel is still licensed under GPLv2. The GPLv2 license, reputedly, has all sorts of challenges for businesses. If you ask a start-up what license they're going to use for their software, they'll never tell you GPLv2 or GPLv3; it's always either going to be AGPL – if they want to use fear as the way they generate revenue – or it's going to be BSD if they want to maximize adoption.

Yet we see around Linux this massive community of people who are using GPLv2-licensed software. So what's going on there? Well it turns out that you can run a perfectly good business using GPLv2-licensed software, as long as the rules give nobody a benefit over anybody else. Companies have adapted to the inconveniences: They maintain a history of where the software came from; they maintain a habit of publishing under open source licenses. So now they've adapted – they faced the regulatory pressures, they faced the legal challenges, and they have adapted to remain profitable in the environment.

Corporations are like animals; they respond only to hunger and fear, where hunger is profit motive and fear is competitive pressure. When corporations get hungry or fearful, they do things that change their behavior. You can't easily persuade a corporation to act ethically, but you can appeal to its fear and hunger. This reality was the whole point of the difference between open source and free software. Free software is an idea for people – it's something ethically compelling – whereas open source is something I use to persuade my employer, showing the business benefit of working with software that's licensed that way. They are the same thing, but expressed for different audiences.

Rather than just relying on ethical arguments, we must devise equivalent mechanisms for privacy. To change Facebook's behavior, for example, we need laws such that they will make more profit and suffer less harm if they do things that don't erode privacy. We have to make it hurt for companies to abuse personal data and the context that interpolating it delivers. We have to demonstrate that the attitude Equifax has shown, and continues to show, is not just unacceptable morally, but also will lead to the company's officers going to prison and will result in the shareholders losing their money.

In a connected world, privacy won't just happen by itself. We need thoughtful laws to create the competitive pressures that make it happen. ■■■

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®.

Even if you aren't a programmer, you can help spread the word about Free and Open Source Software and Hardware. BY JON "MADDOG" HALL

Spreading the FOSS Word

"How can I help Free Software?" is a question I hear a lot. Many people do not know how to program, and they feel helpless when they want to help Free and Open Source Software (FOSS).

There are many ways to help the Free Culture movement, which includes FOSS, Open Hardware, and Free and Open Data.

The first way is simply to use FOSS. Think about the things you do or want to do, and search the Internet for software that might help you with that work. Search engines are your friend, and just a couple of searches for "thing-that-you-want-to-do" combined with "your-favorite-Free-Software-Operating-System" might find exactly what you need. Yes, you may need to download a few projects to see which one is really the best for your needs, but at least you do not have to pay money for that privilege. You might also have to try several closed-source proprietary software products to find the one you want, and with closed source, "kicking the tires" will cost you both time *and* money.

After you find the FOSS you like (looking first in your distribution's software repository is a good idea) and after you use that FOSS a little while, you can tell other people about it. This will both help the authors find more users (if no one uses the software, developing it is useless) and build a base of others who may help answer questions about the FOSS in the future. In the old days, we called this movement "birds of a feather" (from the expression "birds of a feather flock together"), and it is about people helping other people to use software in a better way.

There are also levels of "telling other people." Telling your family and friends is a start, but telling your school board or your small business association is another. These are the "birds" that have needs for software and will multiply the number of people using the software and "flocking" by a hundred fold or more. Find out if your school or university has a computer club, and volunteer to give a talk about Free and Open Source Software and Hardware (FOSSH). Yes, that includes topics like the Raspberry Pi or other GNU/Linux-based single-board computers.

Please do not forget talking and writing to your government officials, stressing open access to data that can be used by projects like OpenStreetMap and OpenGIS.

Another way you can help FOSS is by being diligent in reporting software bugs. You might think that developers hate hearing about bugs, but they really want well-written bug reports. Please do not just say that "the software broke." Tell them what version of the software you are using (often in the Help menu), what version of operating system you are on, what architecture you are using (ARM, AMD, etc.), and a detailed description of what happened with the software. Also, *before* you submit your bug report, scan the existing bug reports to see if the bug was previously reported (it may already be fixed!) and if you can add any information to an already existing bug report.

While I am talking about "bugs," reviewing documentation and giving suggestions on parts of documentation that are confusing or out of date is useful to the project. If you are fluent in different human languages, helping with translations is always appreciated.

Some people have creativity in photography, illustrations, and music. Creating Creative Commons-licensed media for use is also appreciated by FOSSH projects and is fun to use in your own posters and work.

Putting on a FOSSH event or having FOSSH as part of a larger event is another good way to spread the love. You can start small and build it up over time. Even attending an already existing event shows support for the FOSSH community.

Contributing funds to a project is also a way to help. While the software is often free of charge, development systems, Internet fees, and travel to events all cost projects money. Think about what you would have to pay for the software you use, and even a fraction of that money donated to a FOSSH project would help move it forward.

Finally, there is Subutai, a peer-to-peer cloud computing platform. Just by installing Subutai and using it, you generate goodwill that can then be given to projects to help them buy resources such as disk storage, computing power, and other necessary resources. Creating use cases, which Subutai calls blueprints, not only generates goodwill but makes it easier for you to do your work and help others do their work by reusing and improving on your blueprints.

Each contribution you make to Software Freedom and Open Source makes your life and the lives of countless others better. ■■■

Learning about web security with Web Security Dojo

Master Class

Protecting your own websites from attack either costs a lot of money or requires a lot of expertise. Web Security Dojo helps you learn to think like an expert.

BY ERIK BÄRWALDT

Security is now a major focus for Internet users and companies. Unfortunately, the sophisticated nature of recent attack techniques, as well as the ever-increasing surveillance ambitions of the authorities and data-mining corporations, continues to complicate the quest for a safe and secure Internet.

A specialized Linux environment called Web Security Dojo [1] offers an easy way for everyday users and beginning professionals to learn about web security. Dojo is designed to provide practical, hands-on exercises on web security and intrusion techniques.

The Dojo virtual appliance is available on SourceForge [2] as an image of around 2.3GB in OVA format. The Dojo is suitable to run in VirtualBox from version 5.0 and also in VMware. After you download the image, install a test environment in VirtualBox by specifying the storage path for the OVA file in the newly opened dialog via the *File | Import Appliance…* menu. Then create a new virtual machine from the appliance (Figure 1).

The virtual machine is available in VirtualBox as *Dojo 3.0*. After booting, Dojo launches as a Xubuntu 16.04 32-bit system with the XFCE desktop (Figure 2).

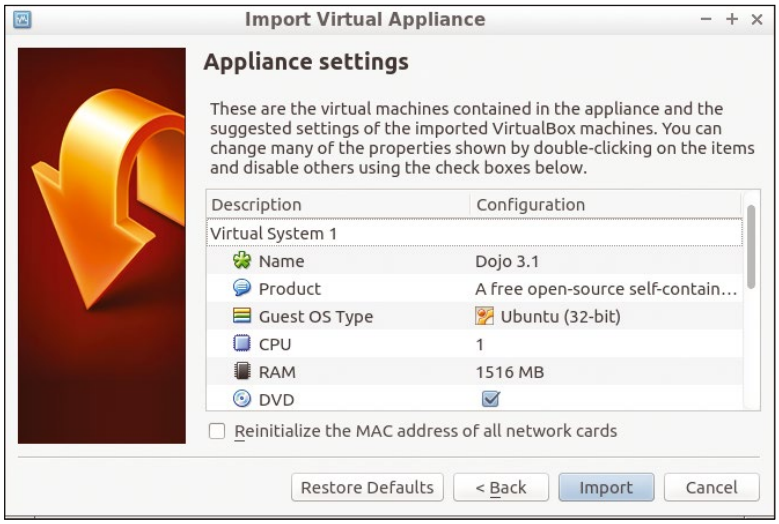


Figure 1: The image creates a preconfigured virtual machine.

Start and Finish

The Web Security Dojo virtual learning environment includes various services that are configured to serve as targets for simulated attacks. The services listed in the Targets menu cover a wide range of possible attack scenarios. Some of these target services are already active by default; others must be launched manually.

Some of the services in the Targets menu are web-based applications that require a proxy service. These proxy services are available in the form of Firefox add-ons (Figure 3). Since targets and tools already exist on the same system, you do not need an active Internet connection for the lessons.

Application

Firefox is the center of Web Security Dojo. When Launched, Firefox first offers the option to call the Damn Vulnerable Web Application (DVWA) page [3], a preconfigured test environment that familiarizes the user with a variety of vulnerabilities in web applications (Figure 4). In the DVWA window, log in with the username *admin* and password *password*.

In the menu on the left, you will find various attack technique options, such as *Cross Site Scripting (XSS)*, *SQL Injection*, *CSRF*, or *Brute Force*. For the various scenarios, you will receive background information in the form of links to related websites and wikis.

Figure 2: The XFCE desktop from Dojo is anything but peaceful.

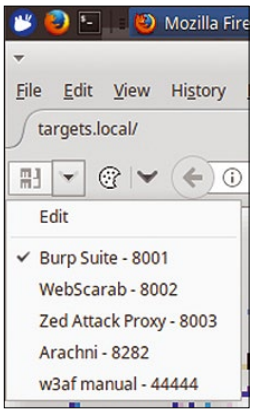


Figure 3: Dojo provides the proxy services that are required for some applications in the form of Firefox add-ons.

In addition to DVWA, Dojo has other tools for more advanced attack scenarios. For example, you will find the Java application WebGoat, which is part of the OWASP Project [4]. Launch WebGoat using the *WebGoat Start* script in the Targets menu, and then click on the WebGoat link in Firefox on the homepage. You can authenticate using *guest* as a username and password.

The application provides a brief introduction and lists various test scenarios in a vertical scrollbar on the left edge of the screen (Figure 5). Subgroups partly summarize individual categories. For example, under *Authentication Flaws* , you will find tests for authentication vulnerabilities.

Several options appear at the top edge of the screen. Click on *Show Solution* to display the solution to a scenario; *Show Plan* provides additional didactic information. *Show Source* familiarizes you with the source code, and *Restart Lesson* launches the active task again. *WebGoat Stop* from the menu stops the service.

Google Gruyere and McAfee's Hacme Casino are two other toolkits for learning protection technologies for web pages. You have to manually launch these tools via the Targets menu before the web pages are available in Firefox. Gruyere, which is named after the cheese, portrays several typical methods for hacking a

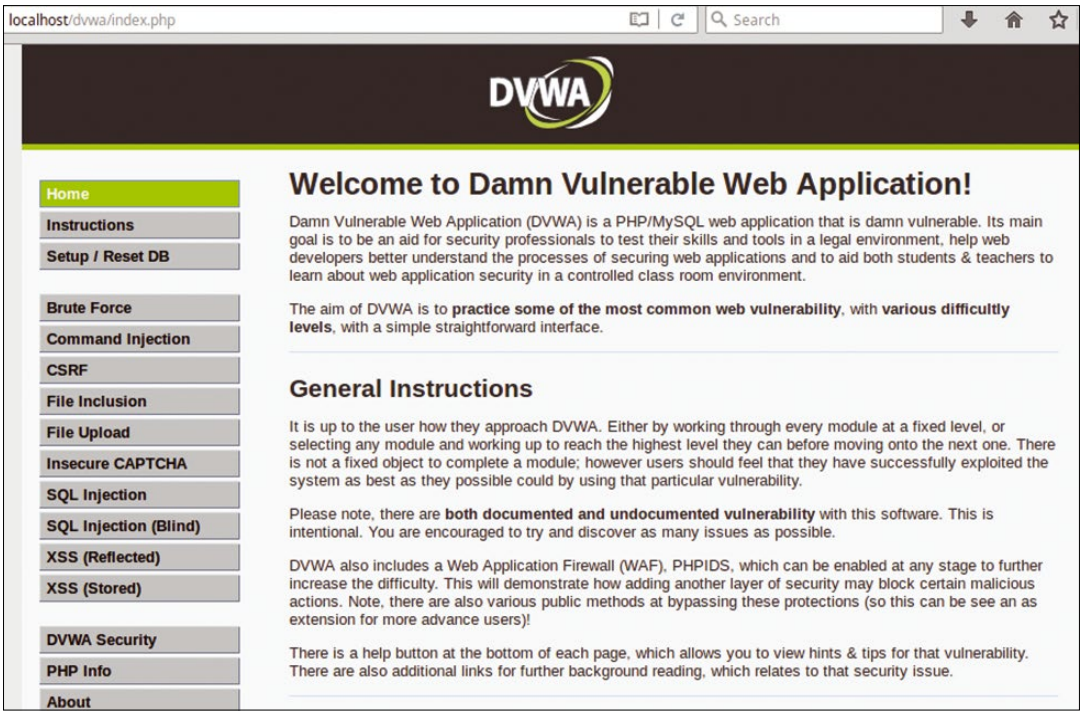


Figure 4: DVWA provides an initial overview of attack scenarios.

website and familiarizes you with solutions that prevent such attacks.

Hacme Casino is extremely playful and looks like a gambling website; however, it also serves as a learning tool, letting the user trace through some common attack techniques. A detailed manual for Hacme Casino is available in English with many practical examples [5].

In the Tools menu, you will find a wide range of tools and scanners for your own research. These tools includes the security scanner Arachni, the browser exploitation framework BeEF, the Metasploit Framework, and the w3af framework – including a command-line version. DirBuster, an application written in Java for brute force attacks, and BurpSuite are also available. Pure command-line applications such as Skipfish, SqlMap, or Skavenger Shell round out the portfolio.

Documentation

The manufacturer has put a lot of effort into documentation for Web Security Dojo. You'll find plenty of PDF and HTML files for the various tools, as well as several video tutorials hosted on YouTube. The documents and videos make it easier for beginners to install and get acquainted with the system. You can also find some basic information on the desktop in the `README.html` and `GettingStarted.html` files.

Instructions for the main suites and frameworks are available in the *Documentation* folder. The Zim desktop wiki is available for you to record your own notes. To launch Zim, click the Zim icon on the desktop.

Conclusions

Web Security Dojo provides an excellent training opportunity for budding security professionals who want to become familiar with the basic mechanisms for protecting web applications. The OVA image is easy to install, and the XFCE desktop is easy to configure. The developers have carefully adapted the tools and test environments for their intended use, so you can get started with the practical exercises right away.

Keep in mind that, although Web Security Dojo is a powerful tool for learning about web security, it is not intended as a production-ready pen test distribution – the developers recommend Kali Linux [6] if you need to do some real penetration testing. ■■■

Info

[1] Web Security Dojo: *https://www.mavensecurity.com/resources/web-security-dojo*

[2] The Dojo at SourceForge: *https://sourceforge.net/projects/websecuritydojo/files/*

[3] DVWA: *http://www.dvwa.co.uk*

[4] OWASP: *https://www.owasp.org*

[5] Hacme Casino handbook *https://github.com/spinkham/Hacme-Casino/blob/master/install/userguide/HacmeCasino_UserGuide.pdf*

[6] Kali Linux: *https://www.kali.org*

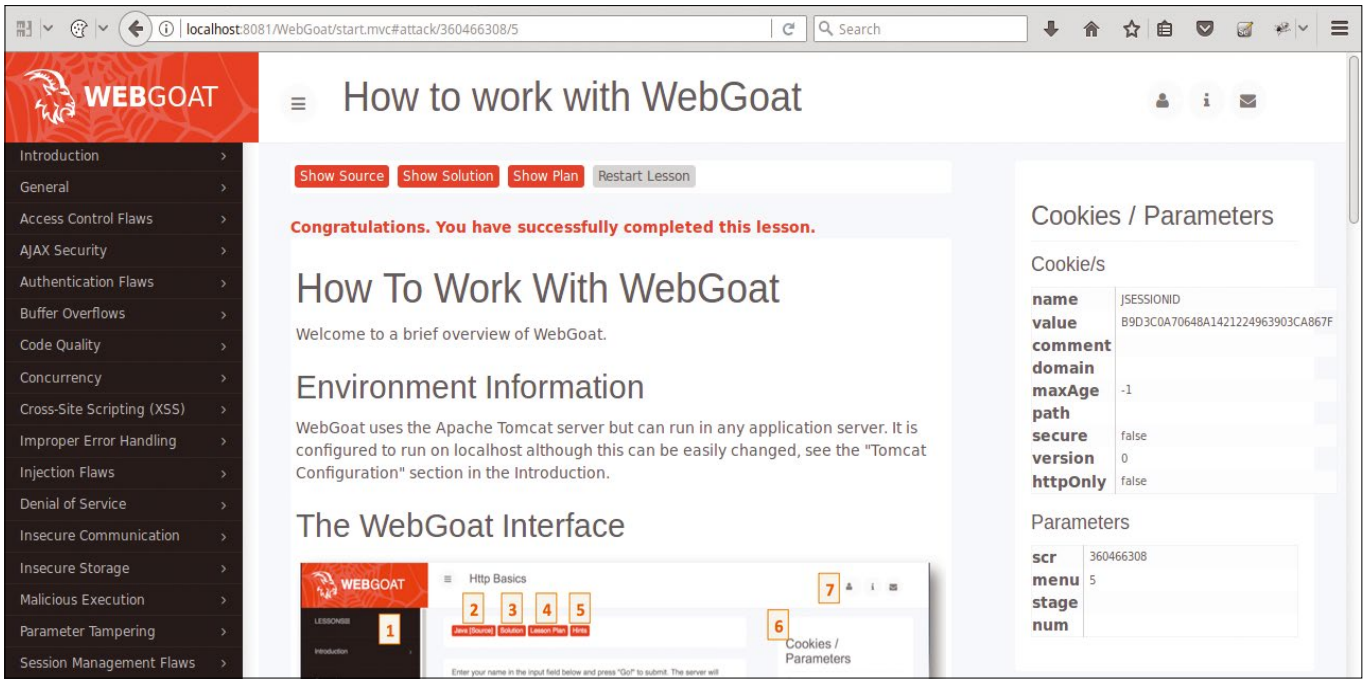


Figure 5: WebGoat is more suitable for advanced users.

FOSSPicks

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

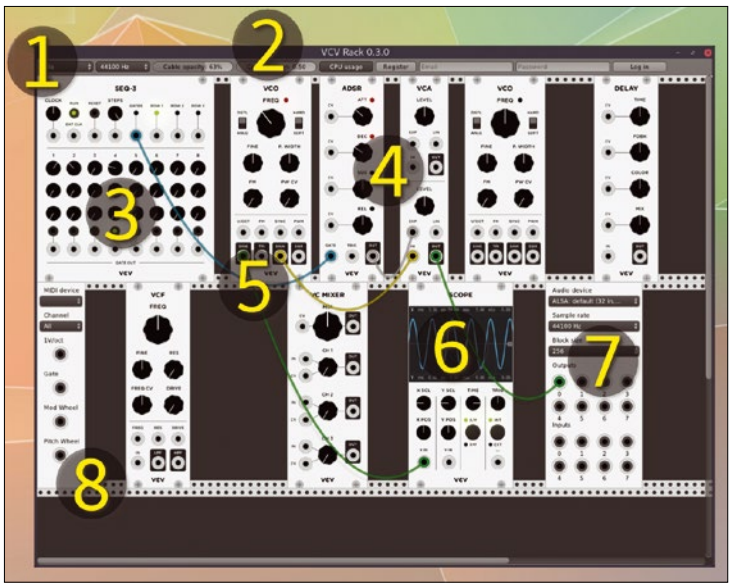
Graham tears himself away from updating Arch Linux to search for the best new free software. BY GRAHAM MORRISON

Modular synth studio

VCV Rack

Before audio synthesizers were neatly packaged into boxes that contained a keyboard and all the components necessary to make a sound, they were modular. This meant you needed to link each component together with patch cables in a way that created the kind of sounds you wanted. Voltages would modulate parameters to produce audio, which could then be routed into different processing modules. After recording or playing the sound, you'd deconstruct the patchwork of interconnections and start again. This is how Delia Derbyshire worked at the BBC Radiophonic Workshop, for example, or how anyone made a sound before Bob Moog came along and changed everything.

Integrated circuits (ICs) eventually pushed these behemoths aside, much as ICs and microprocessors did vacuum tubes. Programmable, neatly packaged synthesizers have now dominated for the last 30 years. But modular synthesizers are coming back, thanks to the combination of a smaller packaging format (called Eurorack), ARM processors, cheap home brew electronics, and a backlash against screens and workstations. Constructing your own sound sources from a mess of patch cables is cool again; sounds are often generative and experimental, with users meeting up to share setups and making the entire scene feel a little like a modular maker community.



1 **Presets:** Unlike real modular synthesizers, you can save and load presets with VCV Rack. 2 **Cable tuning:** Both the appearance and elasticity of cables can be adjusted. 3 **Eurorack modules:** Using the form factor of real modules lends VCV Rack an uncanny feeling of realism. 4 **Bundled modules:** VCV Rack includes a decent collection of eight modules, such as sound generators, processors, and utilities. 5 **Patching:** Cables connect control voltages, audio, trigger, and gate impulses, just like real hardware. 6 **Oscilloscope:** This modules even has a real-time updating input display. 7 **Audio interface:** Get audio output into your headphones and audio input into your virtual modular. 8 **MIDI:** Connect external keyboards to the software.

The only problem is that this is an expensive hobby. Each module is often constructed by a few individuals and produced in small batches. They need specific power supplies and interfaces to get them talking to your computer. And that's why VCV Rack is one of the most brilliant pieces of audio software I've come across in a while. It's a software emulation of the infrastructure needed to host these modules, connect cables, edit parameters, and save patches. It even includes a batch of modules to get you started. There's a VCO for sound generation, a VCF for filtering sound, and both a VCA and an ADSR envelope generator for changing volume over time. There's also a delay effect, a mixer, an oscilloscope, and a sequencer – a batch of modules that would cost you a small fortune if you happened to be building your first physical modular system.

With VCV Rack you simply right-click the modules into your virtual rack and start connecting outputs to inputs. It sounds just as good as the real thing. But I've left the best feature until last: You can install other modules from external sources, and this seems to be Rack's raison d'être. Rack's author, Andrew Belt, has cleverly taken the open source firmware behind some of the best Eurorack modules available, from Mutable Instruments, Befaco, and Synthesis Technology, and emulated their physical connections in software – from their seven-segment displays to their huge knobs. This means you can run virtual recreations of real hardware running the real firmware for free from a Linux desktop, and it's amazing.

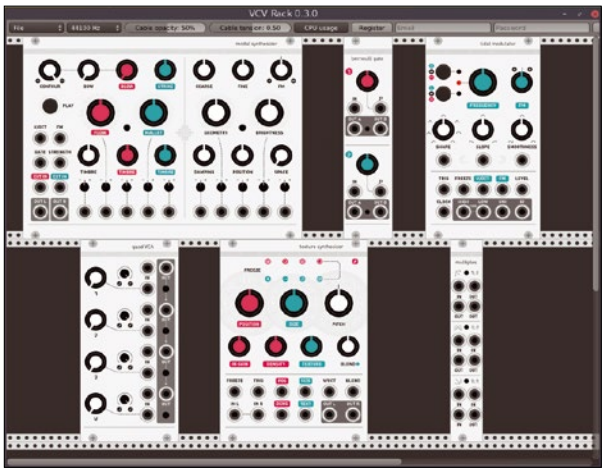
Project Website
https://vcvrack.com/

Virtual audio modules

Audible Instruments

One of the best things about VCV Rack (see opposite) is that it can host unofficial virtual re-creations of real hardware by using the hardware's original open source firmware to provide the same functionality in software. It's analogous to a retro console or computer emulator re-creating the hardware and legally using the original ROM. With VCV, you can connect virtual audio, voltage, and signal cables between the modules, just as you can with the physical hardware. If you connect the audio output to your audio interface, you'll hear the same audio. This is because most of these modules aren't analog, but usually built around ARM microcontrollers and algorithms, with all of the clever sound generation done within its open source firmware.

The Audible Instruments suite of plugins for VCV Rack is inspired by real and very popular hardware created by Mutable Instruments, and the suite is currently in active development, although not all modules are functional. Functionality ranges from 100% in the macro oscillator to 0% in some of the modules that are purely analog and are unlikely to be emulated. But for the modules with good support – the macro oscillator (Braids), modal synthesizer (Elements), tidal modulator (Tides), and texture synthesizer (Clouds) – performance is almost identical. I know this because I have a couple of the modules myself, and it's remarkable that I can now run the same setup on my computer desktop that I run on my physical setup. The only slight hitch is that it's all unofficial; I hope more exposure will mean more people get into Eurorack modules and buy modules from the likes of Mutable Instruments, rewarding the company for its open policy toward firmware hacking and hardware development.



Use open source software to unlock the potential in open source firmware running virtual hardware.

Project Website

https://github.com/VCVRack/AudibleInstruments

Desktop planetarium

Stellarium 0.16.0

There can't be many Linux users who haven't played with Stellarium. It's an amazing desktop planetarium application that lets you see a crisp and perfectly clear night sky from anywhere in the world, from any time, in perfect detail. It's capable of drawing stars, constellations, planets, nebulae, and even Earth-bound features, such as the ground, a 3D landscape, and the atmosphere. With the conjunction of a major astronomical event with a major software Stellarium release, it's worth taking another look. Whether you're looking at a solar eclipse or the view from a different planet, Stellarium's great strength is its wonderful OpenGL graphics that are capable of painting the sky in extraordinary detail.

If you're into making serious observations, Stellarium will help you to find what you're looking for, and the latest release helps, with the new AstroCalc extension that can show you what's up this evening, as well as read your location from a GPS and support the RST2 telescope system. If you're having a sky party with a few friends, the new RemoteSync feature can allow all of your Stellarium instances to talk to each other so that you can stay in sync with the view.

The new version has new catalogs of peculiar galaxies and Hawaiian and Belarusian star lines and even includes 3D models for irregular solar system objects, such as minor bodies like asteroids and small moons.

Relive the 2017 solar eclipse from the safety of your own home.

Stellarium really is a brilliant project and can help kindle a love of astronomy and the wider universe, even when you're sitting beneath Britain's inclement skies. The fact that major features and add-ons like these are still being developed is worthy of praise. Thanks Stellarium team!

Project Website

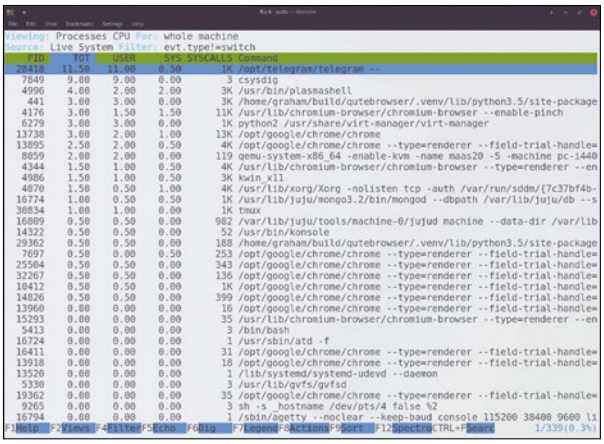
https://sourceforge.net/projects/stellarium/

System monitor

Sysdig

When it comes to system monitoring, most of us have our own regimen built around the simple tools typically installed on most Linux systems. But sometimes, it helps to have all these tools wrapped into a single package, and that's exactly what Sysdig attempts to do. It's capable of watching raw network traffic and monitoring processes and filesystems as a stream of activity that you can either watch as raw data, save for later, or process with a command-line GUI. You can then filter and focus just as you might with an individual tool. Sysdig uses a kernel module to perform this magic, and the default installation is via `curl` with root privileges. You obviously don't want some random code shuffling through your kernel and your clock cycles with that kind of power. Therefore, I'd strongly recommend that anyone installing this on anything other than a test system should take a look at the source code or at least find a package from a maintainer you trust.

One target platform for Sysdig is the world of containers; it includes Docker monitoring, alerting and troubleshooting with Kubernetes, Mesos, and Swarm integration, so a level of isolation may be tolerable. By default, running Sysdig will throw a flow of system activity to your terminal that can, for example, be saved or filtered with arguments to show network connections. For general use, though, it makes better sense to run `sudo csysdig`. This command presents the data through a very useful and rather easier-to-handle curses interface. Pressing F2 cycles through the various monitoring modes, from processes and threads to running containers and Kubernetes controllers, which is ideal for remote servers, especially when monitoring both network traffic and the processes using the connections.



Despite the complexity of Sysdig, the curses user interface is easy to navigate and has plenty of excellent text-based help.

Project Website

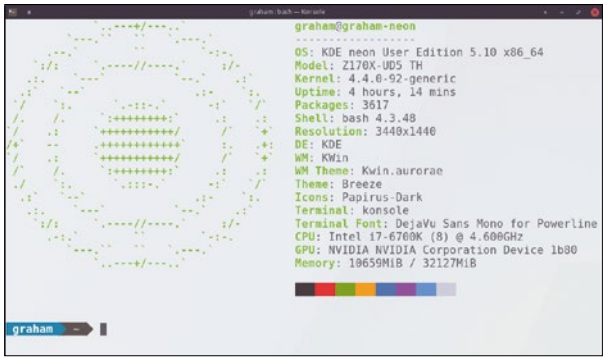
https://www.sysdig.org/

Profile eye candy

Neofetch 3.3.0

Neofetch is a script that initially feels inconsequential but quickly becomes important. On the surface, all it really does is generate some command-line eye candy that displays important details about your current installation. These details include which operating system/distro you're using, your motherboard model, RAM, CPU, desktop environment, and theming options. But it does this in such a clear and concise way that it has become something of a cultural phenomenon. Users will include a thumbnail in their signatures, or as part of an online avatar, or share the output on subreddits and forums. Neofetch has become a shortcut to your Linux (or even Mac OS and Windows) personality or fingerprint. But all that belies how capable it is and the underlying complexity behind making sense of so many different ways to pull data from your system.

There are more than 50 configuration options for changing the way the output is presented, and there's support for Linux, Mac OS, iOS, BSD, Solaris, Android, Haiku, GNU Hurd, MINIX, AIX, IRIX, and Windows (and even Cygwin/MSYS2/MinGW/Windows 10 Linux subsystems). This release adds lots of new distros, including Endless OS and Netrunning, more ASCII logos, embedded image formats, and which GPU driver you're running. Patches for these options have been provided by lots of contributors. This is perhaps because of one of Neofetch's best features – it's written in Bash, and many of its functions are hard coded with huge arrays of `case` and `if` statements. It might not be best programming practice, but it's easy to understand, modify, and contribute toward, and it is perfect for beginners and people who may never have contributed to a project before.



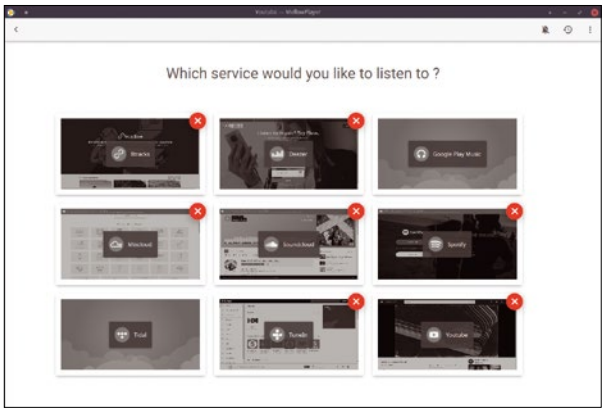
Even if you run Haiku or GNU Hurd (or Neon!), Neofetch will be able to pull your system profile and make it look good.

Project Website

https://github.com/dylanaraps/neofetch

Cloud music player

MellowPlayer 3



If MellowPlayer could just add ad blocking, I'd use YouTube more for audio than video.

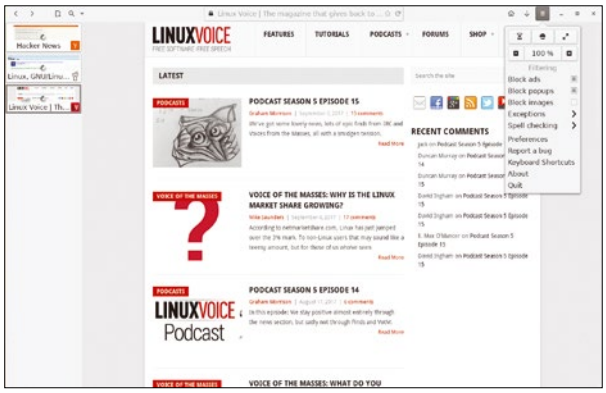
A few music players attempt to harness the various disparate music sources on the web, which is a worthwhile endeavor because it's often difficult to keep track of what appears where and which services are still running. Will SoundCloud still be around in two years, for example, or how long will all those Prince bootlegs be available on YouTube? (They've already taken them down.) A single place for all of your online music sources makes sense, and this is what MellowPlayer is attempting to be. Unlike relying on third-party libraries and unofficial API access, MellowPlayer is really a container for the web interfaces of these services, which is how it supports so many. 8tracks, Deezer, Google Play Music, Mixcloud, SoundCloud, Spotify, Tidal, TuneIn, and YouTube are all supported if you have the accounts to access them. However, you will need access to QtWebEngine compiled with proprietary plugins for many to work, including SoundCloud and Spotify.

What makes MellowPlayer better than a series of bookmarks is that your listening is now all within a single application. You can use your desktop to control playback, and audio is delivered outside of your browser. Each service has notifications and the same user interface, although the service navigation is courtesy of that service's web application, including annoying advertisements. You can even add your own online music sources with the Create Plugin wizard. The process isn't quite point-and-click and involves the creation of a group of configuration files, but it shouldn't be too difficult either. If new services can be added and the plugin system expanded (maybe to include ad blocking?), MellowPlayer could future-proof itself against changing services and perhaps become just as useful as local desktop players.

Project Website
https://github.com/ColinDuquesnoy/MellowPlayer

Web browser

Eolie 0.9



If you want to try Eolie without worrying about installation, the Flatpak download is distro agnostic and includes all the dependencies.

There's always space for another web browser, especially a minimal one that works well with Gnome and Gtk+. Not including its Python dependencies, Eolie's install size is a mere 1.62MB on Arch. What's even more surprising is that Eolie is a lot more than a simple wrapper around the Python libraries that do all the web rendering. It's a very slick, quick, and functional browser that includes nearly all the features you'll ever need. The user interface is exceptional and quite unique in a variety of ways. One is that a side pane is used to host tabs rather than a horizontal bar. This functionality merges into a bookmarks system, as you quickly move between the thumbnails of the tabs you have open. It works brilliantly, and it's great to see a system that seems built for the wide displays most of us now use. Even the thumbnails are wide and show a useful preview of the page. But these aren't the only graphical innovations in Eolie.

Rendering is quick; new private pages can be instantiated quickly; images, ads, and pop-ups can be disabled and reenabled from a clear filtering menu; and WebKitGTK means it works with all the sites and pages you'd expect. There's even the facility to sync your bookmarks, history, and password with Firefox Sync if you need to, which would also be useful if you wanted to use both browsers. The ability to download all images from a single page is built into the main application. Combined with Gnome's window decoration minimalism, the entire Eolie package feels very modern, yet it is still in the early development phase. If development keeps at the current pace, we'll have another brilliant first-class browser on the Gnome desktop just in time for Ubuntu with Gnome to land.

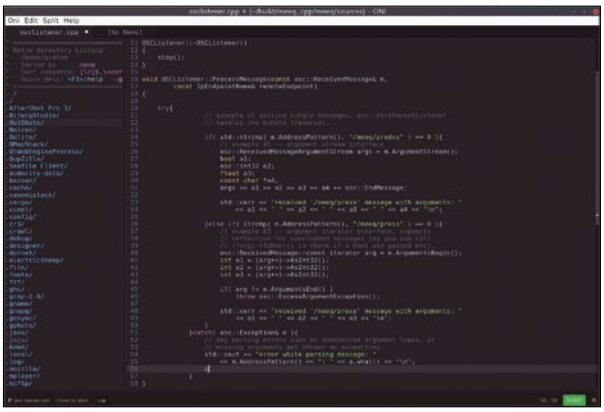
Project Website
https://gnumdk.github.io/eolie-web/

Neovim IDE

Oni

Vim is a wonderful text editor that can be made to do many different tasks. But one of the best things about vim is that after your muscles have mastered its plethora of key combinations and commands, those movements can be used in lots of additional tools. Web browsers, IDEs, and even the Bash terminal can be made to interpret the same commands and keystrokes used by vim, making you super-efficient and cool looking. Which is why Oni – an IDE powered by Neovim (with a little help from React and Electron) – should be of interest to any developer with vim muscle memory.

Oni is built atop Neovim, the substantial community refactor of vim, and adds the kind of features you typically find in an IDE. It includes info overlays for functions, code completion, syntax and compilation error highlighting, and fuzzy search, alongside a new status bar. Of course, many of these can be added to Neovim (and vim!) via third-party engines, but Oni's strength is having all these features together in a self-contained package with minimal setup required. You can start coding from the moment you install Oni. Quick access to menu options allows you to split the views, open up a file manager, and insert text, removing some of the vim learning and remembering burden. Various vim plugins are also included, such as `Targets.vim` and `commenary.vim`, alongside a couple of excellent color schemes (`vim-monokai` and `onedark.vim`). Oni also makes editing the configuration file easy, with a direct link from the menu and a few common options commented out, such as for changing the font size. Behind the lovely UI, Oni also features a new plugin environment that replaces the rather arcane VimL with JavaScript. Best of all, Oni is just as quick and powerful as the original vim, without requiring the years of memory sacrifice.



Make the vim coding learning curve a little easier with Oni.

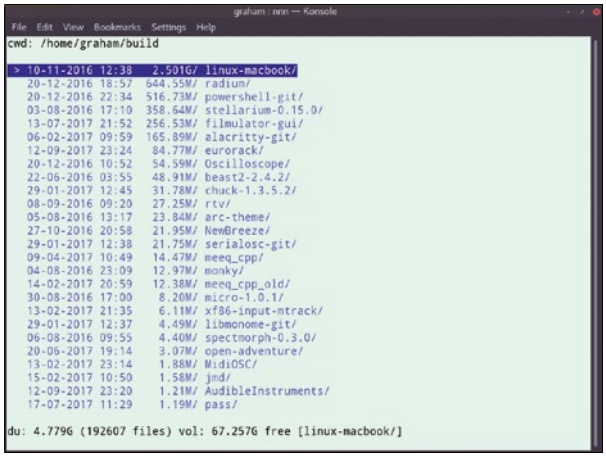
Project Website
https://github.com/extr0py/oni

Fast file management

nnn

File management on the command line is about as fast as file management can get. You type short commands to move, copy, and delete files and directories, switching between local and remote locations with ease. But there are still times when a file manager similar to those found on Gnome, Xfce, or KDE provides a visual overview of your files and folders that makes better sense, especially when dealing with multiple files or exploring documents. This is when you need `nnn` (Noice is not Noice, of course), a fork of the ace terminal file manager noice. After launch, for example, `nnn` loads instantly, and you can immediately use the arrow keys to skip about your filesystem. Press the right arrow on a file, and the default application will load to view whatever MIME type is associated with the file

Starting `nnn` with the `-S` argument will create a catalog of how much storage each folder and file is consuming (probably with `du`) and output this information within the simple file view. This is a good way of identifying fat folders or downloads, and it works better than trying to do the same thing with `sort` on the command line. This being the command line, various keyboard shortcuts are used to access the majority of functions. Pressing *D*, for instance, shows in-depth details on a file, including blocks, MIME type, and file contents. The tilde (~) takes you quickly home or an ampersand (*&*) to the directory in which you started. A powerful set of filters let you search as you type for files matching your criteria in the current folder, such as `\.png` for all PNG image files, and the search will match hidden files if run as root.

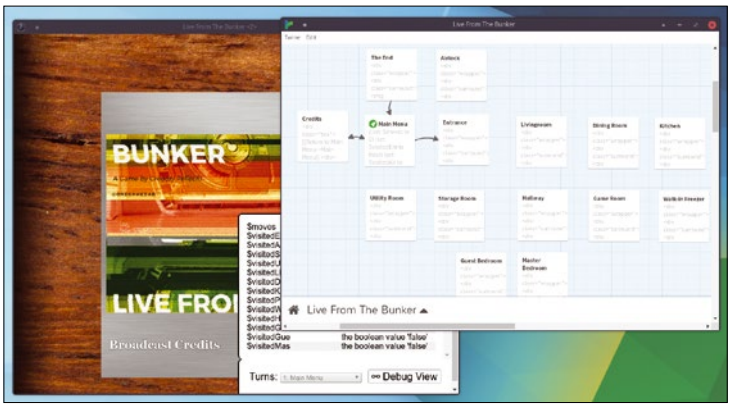


If you need a tiny and quick command-line tool for file management, `nnn` is easy to launch, run, and close, and it doesn't get in the way.

Project Website
https://github.com/jarun/nnn

Story engine

Twine 2.1.3



Many example games and free games written with Twine are hosted on The Interactive Fiction Database (*http://ifdb.tads.org/*).

Twine is a little different from typical games because it's not a pure game. Instead, it's a way to construct nonlinear stories that could be interpreted as a game but could equally be a piece of interactive fiction or a book similar to those "choose your own" adventure novels from the 1980s. At the heart of the story creation is basic chunks of text and links to different passages. You could offer the reader a description of a location, for example, and then offer a series of links for them to decide how they want the story to develop, such as whether you go north, south, east, or west. It's very similar to web design, using HTML and CSS, and that makes Twine equivalent to an IDE, helping you manage the various elements of a story, change links, and even add extras, such as health or inventory management, without needing to worry about cookies or code like JavaScript. Each location becomes a node in a story map showing how each node relates to each other and how you progress from one location to another. Creating games puts the emphasis on imagination and writing, making it ideal for children. It could even be useful for people planning to write a book, because you can use nodes to hold locations and characters and experiment with how they interact through the text and an evolving story. The final output can also be seamlessly hosted online, just as you might a website. The great thing about this approach is that it means almost anyone can create a game, but it also means that those games are going to be a little different, and certainly not of the first person shooter variety.

Project Website
https://twinery.org/

Strategy

TripleA

If you enjoyed the maps in Defender of the Crown, you'll love TripleA.

TripleA, not to be confused with the developer Triple, Eh? of Lumo fame, is a strategy board game similar in style to Axis & Allies and Risk, only played with a mouse on your desktop. You take charge of an entire country in the middle of a World War II-like scenario, although this can be changed and modified. You have to research, buy, move, and fight your way to victory, expanding your influence, making allies, and destroying your enemies. Gameplay takes place on a political map of the globe where you buy your technology, move units, and make combat moves against adjacent territories on the map. You can play against the computer, play online, match yourself against other players in a lobby, and even play via email.

This game has been in development since 2001. As a result, it is now a very mature and stable application with a committed community of players. However, you'll need to brush up on its rules if you're going to take on any of these players. Fortunately, TripleA's Rule Book is a good example of what happens when a game reaches maturity; documentation like this is usually left until some mythical point in the future. However, the Rule Book is a well-designed and well-written PDF that steps through the game from setup to strategy. It even uses annotated illustrations to explain ideas, stepping through an example conflict and the various moves, deployments, and technology phases you can go through to win. The final section is a tactical handbook, and it is definitely worth a look – if you're brave enough to play online – if for no other reason than it will give you an idea of how serious players approach the game.

Project Website
http://www.triplea-game.org

Devilspie2: Scripted Window Actions

Stop battling your window manager to position things as you like – make scripts do all the hard work!

BY MIKE SAUNDERS

How many window managers (WMs) and desktop environments can you name? If you've been around the Linux or BSD scene for a while, you can probably come up with 10 or 20 – and there are even more. For new users, this sheer variety may seem perplexing. After all, a WM just lets you shove windows around on the screen, right? Why not just make a single WM to rule them all?

Well, as you know, different WMs and desktop environments target very different types of users. Some prefer minimalistic WMs with tiny window borders, driven primarily by keyboard shortcuts – whereas others want all-singing, all-dancing showcases of beauty and functionality. In any case, they all have limitations, especially in terms of automation.

Now, why would you want to automate a WM? Don't they already do some tasks automatically, like putting new windows in certain places to reduce overlap? Well, yes, but that's about it. Wouldn't it be awesome if you could create custom scripts to move, resize, minimize, or maximize windows automatically when they pop up? Or move them to specific workspaces or virtual desktops? Or even make them sticky across all desktops?

That's what Devilspie2 [1] does. It's a "window matching utility, allowing the user to perform scripted actions on windows as they are created." As the name suggests, it's a follow-up to an older program called Devilspie that did a similar job but used its own custom scripting syntax. Devilspie2, however, uses the popular Lua scripting language [2], letting you create powerful setups for almost any purpose.

Over the next few pages, I'll show you how to get started with Devilspie2, perform basic window operations, and build up more complex scripts. You'll see how tedious, everyday window management tasks can be automated, letting you focus on getting things done instead of trying to grab teensy little window borders with the mouse (or remembering a bunch of keyboard shortcuts).

Fire It Up

Before starting to using Devilspie2, you might be thinking: Which WM or desktop environment is most suitable for this program? Ultimately, it works with pretty much every WM out there, but if you want to focus on getting things done quickly, it's worth also investigating a lightweight WM or desktop environment such as Openbox, LXDE, or even Xfce. The good thing is that Devilspie2 works independently, so the techniques you learn here can be applied to other WMs, even if you change further down the road.

Devilspie2 is available in the package repositories of most distributions, so just fire up your package manager to grab it, or get it from the command line on Debian/Ubuntu-based distros like so:

```
sudo apt-get install devilspie2
```

Make sure to specify the *2* at the end, though, because the older version is still provided in many repositories – but it won't work with this tutorial! If you can't find a binary package for your distro, you can compile the source code, which is available on the program's website; it doesn't have many dependencies.

In a terminal window, enter `devilspie2` to start it, and you'll be greeted with:

```
No script files found in the script folder - exiting.
```

You'll land back at the command prompt. That's not very friendly, is it? Well, Devilspie2 on its own doesn't do anything in particular – you need to give it some actions (via scripts) before it becomes a useful tool. But where do you place these scripts?

If you go to `.config/devilspie2/` in your home directory -- which was created when you ran Devilspie2 the first time a moment ago – you'll see that it's empty, so create a simple script to give Devilspie2 something to do. Before you can start working with windows, you need to learn how to identify them, so create a plain text file called `debug.lua` with the following content:

```
debug_print("Application: " .. get_application_name())
debug_print("Window: " .. get_window_name());
```

Save this in `.config/devilspie2/` and then run the program:

```
devilspie2 --debug
```

With this command, Devilspie2 loads the `debug.lua` script you created and also interprets the `debug_print` commands you added. Leave Devilspie2 running in the terminal, and try clicking around in other windows on your desktop and starting new programs. Each time you open a new window (e.g., by starting a program), you'll see some output in the terminal – this shows how Devilspie2 is identifying each window (see Figure 1).

This output will be important when you begin writing scripts. By taking note of the application and window names, you can use them to tell Devilspie2 to work with specific programs and windows. Usually the program name is the same as you'd find in your menu or package manager – but sometimes there are subtle differences, which is why it's important to do this step.

For instance, if you start the Thunderbird email client and write a new message, you end up with two windows: one for the main Thunderbird view, and one for the message you're writing. As you can see in the Devilspie2 output, they both come under *Thunderbird* for the app name, but the window names vary. You can stop Devilspie2 from running by pressing Ctrl+C in the terminal window.

I Like To Move It

Now that you can identify apps and windows, you can do things with them! Say that you want to set the exact position and size of new Firefox windows as they're opened. Using the debugging script you created before, you can see that the main window of Firefox is called, sensibly enough, *Mozilla Firefox*. But how do you tell Devilspie2 to select this window? Try adding this to the `debug.lua` file, and run Devilspie2 again:

```
if (get_window_name() == "Mozilla Firefox") then
  set_window_geometry(50, 100, 800, 600);
end
```

If you're not familiar with Lua, you can see that the syntax is a bit like a mixture between C and Bash scripting. In any case, for Devilspie2, you're not going to be making very complex scripts – so this syntax is all you need to know.

And as you've probably guessed, you ask Devilspie2 to check whether the currently selected window has the title *Mozilla Firefox*, and if so, you set its geometry. But what do these four numbers mean? In this case, they refer to *x*, *y*, width, and height values, so you tell Devilspie2 to place newly created Mozilla Firefox windows at 50 pixels from the left of the screen and 100 from the top and to set the window at 800 pixels wide and 600 pixels high.

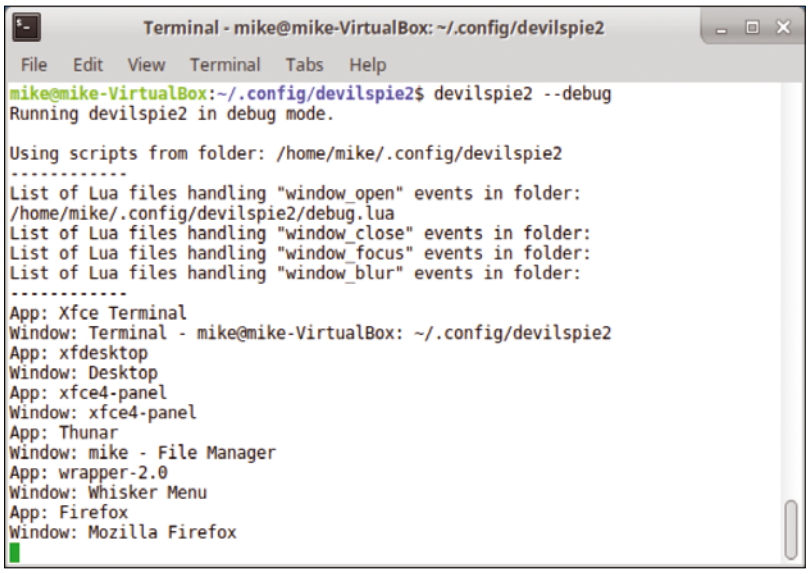


Figure 1: In debug mode, Devilspie2 shows how it identifies apps and windows, so you can use those names in your scripts.

Try running Devilspie2 and opening and closing Firefox, resizing and repositioning the window each time. Whatever you do, when you restart Firefox, it always appears at the exact same position with the exact same size. This is very useful if your favorite WM tries to be "clever" by automatically shuffling things around, but you really want a certain app always to be in a specific place and with a specific size.

Many programs remember their window size when you close them and, indeed, whether they were running in a maximized state. Some apps don't pay attention to this, however, but Devilspie2 can help out here as well. Say you always want LibreOffice Writer to start in maximized mode, regardless of what it remembers from last time (or what the WM wants to do with it). Using the debugging info, you see a potential problem: The window names for LibreOffice Writer can change (Figure 2).

For instance, with a blank new document it's *Untitled 1 – LibreOffice Writer*, but if you create another document or open an existing one, then the name will change. This means that a direct match with the Devilspie2 `get_window_name()` routine isn't possible – you need to be a bit more flexible. Fortunately, Lua lets you do this by using `string.match()`, like so:

```
if string.match(get_window_name(), ➥
                "LibreOffice Writer") then
  maximize();
end
```

In this case, you're just checking to see whether the string `"LibreOffice Writer"` can be found some-

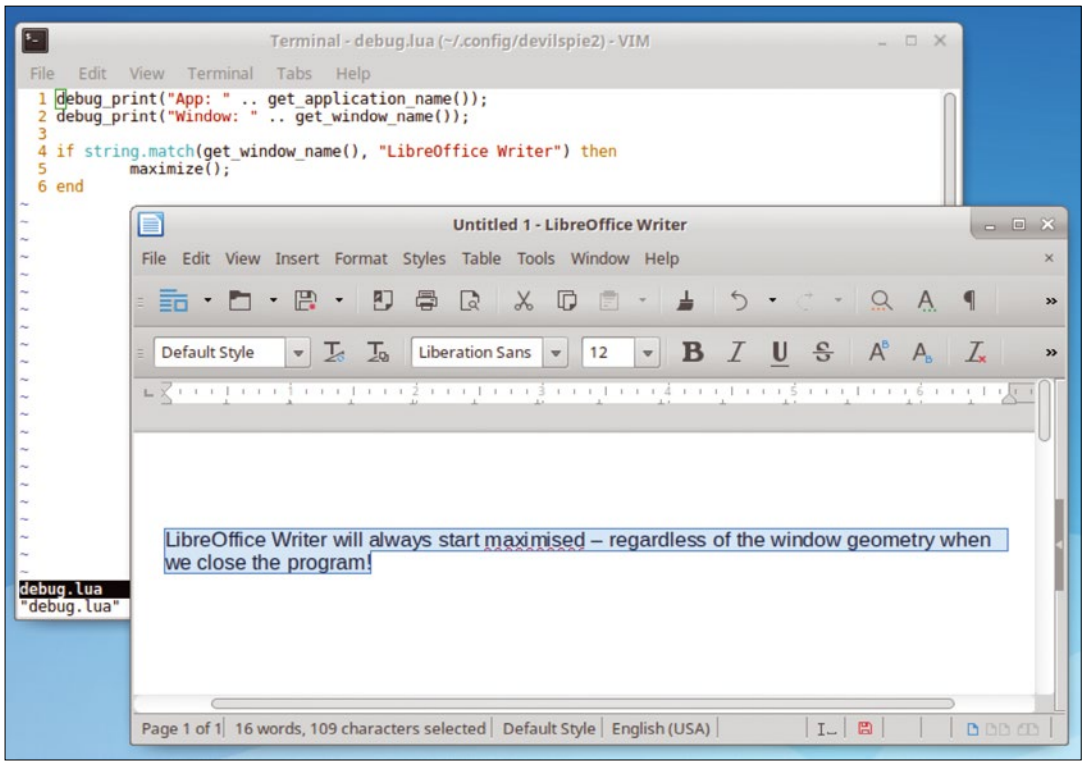


Figure 2: Here, you tell Devilspie2 always to start LibreOffice Writer in maximized mode, regardless of the window geometry when it was closed.

where in the output of `get_window_name()` – the strings don't have to be identical, but if there's a match, you use `maximize()` to make the window fill up the screen (but not covering docks or menus).

Try it – every time you open a new LibreOffice Writer window, it will start maximized, regardless of the window geometry from the last execution of the program. Note that some extra maximize functions are available in Devilspie2, such as `maximize_vertically()` and `maximize_horizontally()`, along with `unmaximize()`, which you can use if an app stubbornly refuses to start in anything but a maximized mode. If you always want a certain program to start minimized, you can do that with the `minimize()` routine, as well.

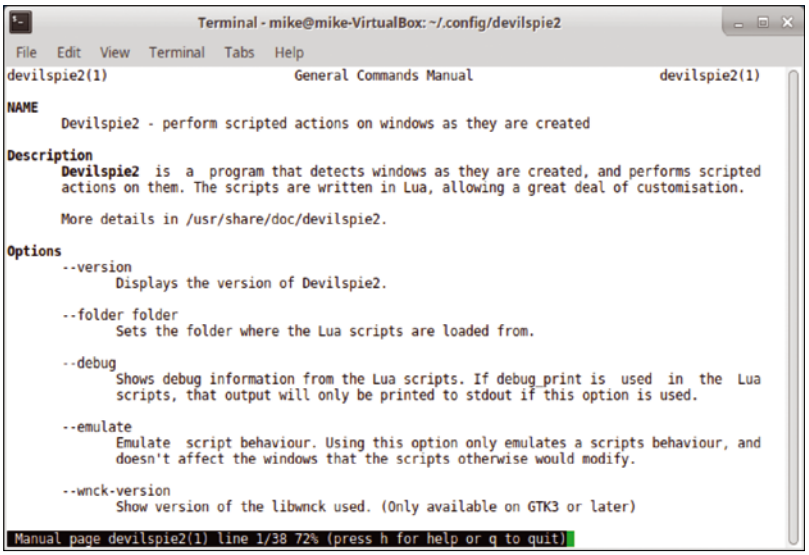


Figure 3: The Devilspie2 documentation isn't great, but the README lists all available functions.

Working with Workspaces

Here's another scenario: You always want to open a specific app on a specific workspace. Some more advanced WMs provide facilities to do this, but Devilspie2 lets you do it regardless of the WM or desktop environment you're using. Starting with the LibreOffice example, again, say you always want LibreOffice windows – whether Writer, Calc, or the options or save dialogs – to open in the second workspace.

You can try to do a match using `get_window_name()` again, but if you open LibreOffice and do various tasks, you can see that they all come under the app name *LibreOffice 5.4*. (Of course, if you're running an earlier version of the suite, you will see a different number there.) Instead of trying to match windows by name, just tell Devilspie2 to put everything under `"LibreOffice 5.4"` on the second workspace:

```
if (get_application_name() == "LibreOffice 5.4") then
  set_window_workspace(2);
end
```

Now try launching various LibreOffice components, such as Writer, Calc, and Impress; you'll see that they all get placed automatically on the second workspace, thanks to Devilspie2. (If the splash screen appears elsewhere, you might need to set up an additional rule covering just `"LibreOffice"` for the app name, alongside `"LibreOffice 5.4"`.)

One problem here is, you might find it a bit jarring that certain apps start up on other workspaces – wouldn't it make more sense to switch

Alternatives: wmctrl and xdotool

As you've seen, thanks to Lua scripting, Devilspie2 is quite easy to work with. If you want a smaller tool for interacting directly with windows, though, then take a look at `wmctrl` [5]. This is a "command-line tool to interact with an EWMH/NetWM compatible X Window Manager" – which means pretty much all major WMs written (or updated) in the last decade. Examples include Blackbox, IceWM, Englightenment, and Window Maker.

As with Devilspie2, you can use wmctrl to move windows around, resize them, put them on specific workspaces, and do other tasks. It's a more demanding tool to use, in that some familiarity with X Window System underpinnings is important, but the documentation is fairly good. You can even do some more advanced things like changing window titles on the fly and resizing the current desktop.

Another tool that's slightly related is `xdotool` [6], which lets you "simulate keyboard input and mouse activity, move and resize windows, etc." (i.e., you can create scripts that send virtual key presses and mouse clicks to windows) (Figure 4). It's especially useful for automating certain tasks, such as clicking on a series of buttons in a complicated dialog, so you can even use it for automatic testing of GUI software. Like `wmctrl`, it's not the easiest program to use out of the box and has a somewhat steep learning curve, but it's worth putting in some time to learn the basics. Still, `wmctrl` and `xdotool` are both included in the package repositories of all big-name distros, so you don't have to go far to find them.

to that workspace beforehand? This is also possible using `change_workspace()` and placing a numerical value between the brackets.

There are some other useful things you can do regarding workspaces as well. For instance, with the `pin_window()` routine, you can tell Devilspie2 to make a window appear across all workspaces; that is, it's always visible, even when you switch to a different workspace. This setting is useful for certain small widgets like system monitors or music players that you always want to have visible in the corner of the screen.

Similarly, Devilspie2 provides ways to add and remove "decorations" from windows. In X Window System parlance, these decorations refer to the titlebar and handles around the outside of the window. If you have a cool little applet or monitor that you want to put in a specific place on the screen, without the WM adding extra fluff around it, Devilspie2 is a godsend: Use it to position the window exactly on the screen, make it available on all desktops with `pin_window()`, and then remove its decorations.

Going Further

Now you've seen how Devilspie2 can – over time – make your workflow more efficient by automating all of the fiddly little window management tasks that you do every day by hand, but it includes many other routines, as well, such as making windows invisible from task lists or pagers, placing them always on top of other windows, or centering them in the screen (like a splash screen).

At the time of writing, Devilspie2's manual [3] was very much incomplete and didn't even include examples, but that's what this tutorial is for! You can, however, bring up a rather dry but detailed list of additional window management routines in the README [4] (Figure 3). Let me know how you get on with this handy little tool and how it makes a difference to your daily work! Also check out two other WM tools in the "Alternatives: wmctrl and xdotool" box that provide similar functionality for automating tasks, but with other features and benefits. ■■■

Info

[1] Devilspie2: *http://www.nongnu.org/devilspie2/*

[2] Lua: *https://www.lua.org*

[3] Devilspie2 manual: *http://www.gusnan.se/devilspie2/manual.php*

[4] Devilspie2 README: *http://git.savannah.gnu.org/cgit/devilspie2.git/plain/README*

[5] wmctrl: *http://tripie.sweb.cz/utils/wmctrl/*

[6] xdotool: *http://www.semicomplete.com/projects/xdotool*

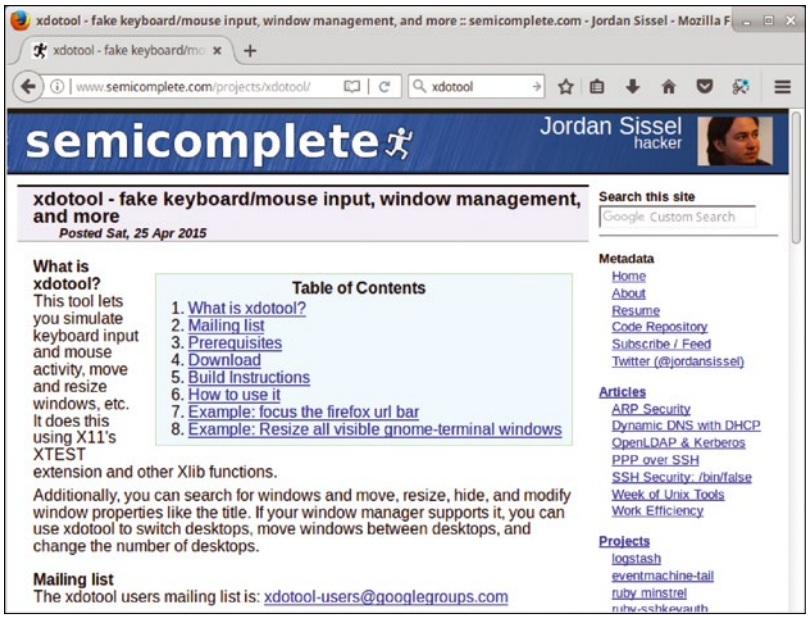


Figure 4: Xdotool, a great alternative for automation, sends virtual clicks and key presses to apps.

Swap stops in for SUSECON 2017

Prague Time

SUSE travels to Prague to celebrate its 25th anniversary. BY SWAPNIL BHARTIYA

As a SUSECON regular, I recognized the historical importance of this year's SUSECON, which took place on September 25-29 in historic Prague, Czechia at the Hilton Prague hotel. This year, SUSE celebrated its 25th anniversary. SUSE, which is the oldest Linux company, was founded just a few months after Linus Torvalds released the Linux kernel.

SUSECON alternates between North America and Europe. This year it was Europe's turn. Prague was an expected location because it is the second largest base of SUSE/openSUSE developers after Nuremberg.

I flew out of DC on September 23rd and arrived in Prague on the 24th. Next morning, I headed for a press conference and one-to-one interviews with SUSE executives.

According to Nils

It has become a tradition for me to kick-start my SUSECON interview series with Nils Brauckmann, the CEO of SUSE. Before I talk business, I'll start with a fun fact for SUSE fans. SUSE has its own band that plays at the event. The band is made up of SUSE engineers and executives, and trust me, they are amazing. Brauckmann himself is a great drummer, but he is stage shy. I have been asking him for the last two years to make a stage debut at SUSECON, but? I will keep trying. (He said he has better drummers on his team.) As consolation, I did get to see Ralf Flaxa, SUSE president of engineering, at the keyboard. Not the computer keyboard – the one on stage.

Brauckmann has been heading SUSE ever since it was acquired by Attachmate, and he has steered SUSE out of troubled waters and into the safe hands of Micro Focus.

According to Brauckmann, the company has been growing tremendously. This year, the company registered more than a 21% increase in revenue, which brings it in the ballpark of $303 million. That's smaller than Red Hat's $2.4 billion annual revenue, but SUSE only recently got back on its feet, so the achievement is certainly significant.

Within one year, SUSE has already made two acquisitions: in addition to buying openATTIC, they have acquired HPE's portfolio of OpenStack and Cloud Foundry technologies/IP. Brauckmann hinted at the possibility of more acquisitions if they are needed to help SUSE meet its goals.

I also interviewed Michael Miller, president of strategy, alliances, and marketing; CTO Thomas Di Giacomo; and Ralf Flaxa.

On the Stage

The main event started on September 25th when Brauckmann and Miller took over the stage to reflect on SUSE's 25 years, its growth, and its future. Although SUSE is still primarily a Linux vendor, it's trying to grow beyond that label and become a technology leader that offers a wide range of solutions to run modern workloads.

On the 2nd day, Miller and Giacomo (aka Dr. T) took over the stage, where they talked about the coordination between marketing and engineering. Giacomo invited K. V. Srinivasan from Microsoft to the stage. Srinivasan has been one of the leading contributors to the Linux kernel. Srinivasan talked

Images courtesy of Swapnil Bhartiya

about the work SUSE and Microsoft have been doing to add Linux support to Microsoft's Azure cloud.

Abby Kearns, executive director of Cloud Foundry, was a surprise guest who talked about Cloud Foundry and how SUSE is using Cloud Foundry to build a Cloud Application Platform. Kozo Otsuka, CTO for Fujitsu, also joined Miller and Giacomo and talked about the decade-old relationship between SUSE and Fujitsu to offer mission-critical solutions to customers.

On the third day, Flaxa and Giacomo talked about continuous integration and continuous delivery. They also highlighted the synergy between the openSUSE community and SUSE business.

Flaxa mentioned two gems from openSUSE and SUSE communities: Package Hub and Open Build Service. These two projects enable developers to package applications – not just for SUSE and openSUSE, but also for competing projects.

Giacomo mentioned the growing influence of containerization, and that an increasing number of SUSE products are becoming containerized. Gary Thome, vice president, chief technologist for software defined infrastructure at HPE, joined the duo to talk about the work HPE is doing with SUSE to help users solve their challenges.

SUSE engineers a panel discussion, in which they talked about the pace of innovation and how SUSE copes with change. Then SAP's Martin Fassunge talked about the work SAP and SUSE have been doing together and what makes SUSE a smart choice for SAP workloads.

Thoughts

One interesting aspect of the show was that most SUSE executives were wearing Tumbleweed T-shirts, which reflects the significance of Tumbleweed across SUSE products. The emphasis on openSUSE's Tumbleweed distribution also reflects the strong relationship between the openSUSE community project and SUSE the company.

One of the biggest highlights of SUSECON is always Demopalooza, where the teams give amazing demos with "beer as a service" delivered to attendees at their seats. This year some technical failures occurred during the demos. There was even one minor accident. A free Raspberry Pi was thrown out into the audience and it hit a guy in the forehead. Ouch.

All three days, the entertainment was provided by SUSE's own band. Brauckmann never did play the drums, but he did join the band to sing the happy birthday song. SUSE threw a lavish birthday party for itself at the historical Zofin Palace.

Rumor has it that the next SUSECON will take place in a Canadian city. ■■■

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*.

SC17

Date: November 12–17, 2017

Location: Denver, Colorado

Website: *http://sc17.supercomputing.org*

SC17 showcases work in HPC, networking, storage, and analysis by the international HPC community. The core of the conference is the Technical Program, with peer-reviewed content for every track, including invited talks, panels, research papers, tutorials, workshops, posters, and Birds of a Feather (BoF) sessions.

Open Source Forum

Date: November 15, 2017

Location: Yokohama, Japan

Website: *http://events.linuxfoundation.org/events/open-source-forum*

The Open Source Forum at TKP Garden City is an invitation-only event designed to advance the open source industry in Japan by bringing the hottest open source technology topics and people together to collaborate. You can request an invitation before November 10, 23:59, JST.

Black Hat Europe

Date: December 4–7, 2017

Location: London, England

Website: *https://www.blackhat.com/*

Tune in to the very latest in research, development, and trends in information security over four days – two days of deep technical hands-on Trainings (skill building for both offensive and defensive hackers) and two days of the research and vulnerability disclosures in Briefings (latest information on security risks and trends).

EVENTS

SPTechCon	November 12–17	Washington, DC	http://www.sptechcon.com/
SC17	November 12–17	Denver, Colorado	http://sc17.supercomputing.org/
DPDK North America Summit	November 14–15	San Jose, California	http://events.linuxfoundation.org/events/dpdk-north-america-summit
ContainerConf	November 14–17	Mannheim, Germany	https://www.containerconf.de/
Open Source Forum	November 15	Yokohama, Japan	http://events.linuxfoundation.org/events/open-source-forum
Open Compliance Summit	November 16–17	Yokohama, Japan	http://events.linuxfoundation.org/events/open-compliance-summit
Open vSwitch Fall Event	November 16–17	San Jose, California	http://events.linuxfoundation.org/events/open-vswitch-fall-event
Linux Presentation Day 2017.2	November 18	Europe-wide in numerous cities	http://www.linux-presentation-day.org/
OSMC (Open Source Monitoring Conference)	November 21–24	Nuremberg, Germany	https://www.netways.de/events/osmc/overview/
Black Hat Europe	December 4–5	London, England	https://www.blackhat.com/
KubeCon and CloudNativeCon North America	December 6–8	Austin, Texas	http://events.linuxfoundation.org/events/kubecon-and-cloudnativecon-north-america
AGL Showcase at CES 2018	January 9–12, 2018	Las Vegas, Nevada	http://events.linuxfoundation.org/events/agl-showcase-at-ces-2018
Open Source Leadership Summit	March 6–8, 2018	Sonoma Valley, California	http://events.linuxfoundation.org/events/open-source-leadership-summit
Embedded Linux Conference	March 12–14, 2018	Portland, Oregon	http://events.linuxfoundation.org/events/embedded-linux-conference

CONTACT INFO

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Managing Editor
Rita L Sooby, rsooby@linux-magazine.com

Localization & Translation
Ian Travis

News Editor
Swapnil Bhartiya

Copy Editor
Amy Pettle

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen

Cover Image
© Denis Ismagilov, 123RF.com

Advertising – North America
Ann Jesse, ajesse@linuxnewmedia.com
phone +1 785 841 8834

Advertising – Europe
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 99 34 11 48

Publisher
Brian Osborn, bosborn@linuxnewmedia.com

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
616 Kentucky St.
Lawrence, KS 66044 USA

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)
Fax: 1-785-856-3084

For all other countries:
Email: subs@linux-magazine.com
Phone: +49 89 99 34 11 67
www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

Printed in 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, 616 Kentucky St., Lawrence, KS, 66044, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 616 Kentucky St., Lawrence, KS 66044, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany.

United States Postal Service Statement of Ownership, Management, and Circulation

1. Publication Title: Linux Pro Magazine
2. Publication No.: 1752–9050
3. Filing Date: 09/30/17
4. Issue Frequency: Monthly
5. No. of Issues Published Annually: 12
6. Annual Subscription Price: $124.95
7. Complete Mailing Address of Known Office of Publication: 616 Kentucky, Lawrence, KS 66044;
Contact Person: Brian Osborn; Telephone: 785–856–3080
8. Complete Mailing Address of Headquarters or General Business Office of Publisher:
616 Kentucky, Lawrence, KS 66044
9. Full Names and Complete Mailing Addresses of Publisher, Editor, and Managing Editor:
Brian Osborn, Publisher, 616 Kentucky, Lawrence, KS 66044
Joe Casad, Editor, 616 Kentucky, Lawrence, KS 66044
Rita Sooby, Managing Editor, 616 Kentucky, Lawrence, KS 66044
10. Owner: Linux New Media USA, LLC, 616 Kentucky, Lawrence, KS 66044; Stockholders: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany
11. Known Bondholders, Mortgages, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages, or other Securities: None
12. Tax Status: Has not changed during preceding 12 months
13. Publication Title: Linux Pro Magazine
14. Issue Date for Circulation Data: October 2017
I certify that all information furnished on this form is true and complete. Brian Osborn, Publisher

15. Extent and Nature of Circulation		Avg. No. Copies Each Issue During Preceding 12 Months	No. Copies of Single Issue Published Nearest to Filing Date
a. Total Number of Copies (Net Press Run)		7590	7310
b. Paid Circulation	b. (1) Paid Outside-County Mail Subscriptions	1297	1123
	b. (2) Paid In-County Subscriptions	0	0
	b. (3) Sales Through Dealers & Carriers, Street Vendors, Counter Sales	3065	2789
	b. (4) Other Classes Mailed Through the USPS	158	176
	c. Total Paid Distribution	4520	4088
d. Free Distribution	d. (1) Outside-County	0	0
	d. (2) In-County	0	0
	d. (3) Other Classes Mailed Through the USPS	0	0
	d. (4) Free Distribution Outside the Mail	292	200
	e. Total Free Distribution	292	200
	f. Total Distribution	4812	4288
	g. Copies Not Distributed	2778	3022
	h. Total	7590	7310
	i. Percent Paid Circulation	94	95
16. Total Circulation includes electronic copies.			
a. Paid Electronic Copies		4722	4796
b. Total Paid Print Copies (Line 15C) + Paid Electronic Copies		9242	8884
c. Total Paid Print Distribution (Line 15F) + Paid Electronic Copies		9534	9084
d. Percent Paid (Both Print & Electronic Copies)		97	98

✓ I Certiy that 50% of all my distributed copies (Electronic & Print) are paid above a nominal price.

AUTHORS

NOW PRINTED ON recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

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.

Approximate
UK / Europe Dec 04
USA / Canada Dec 29
Australia Jan 29
On Sale Date

Issue 206 / January 2018

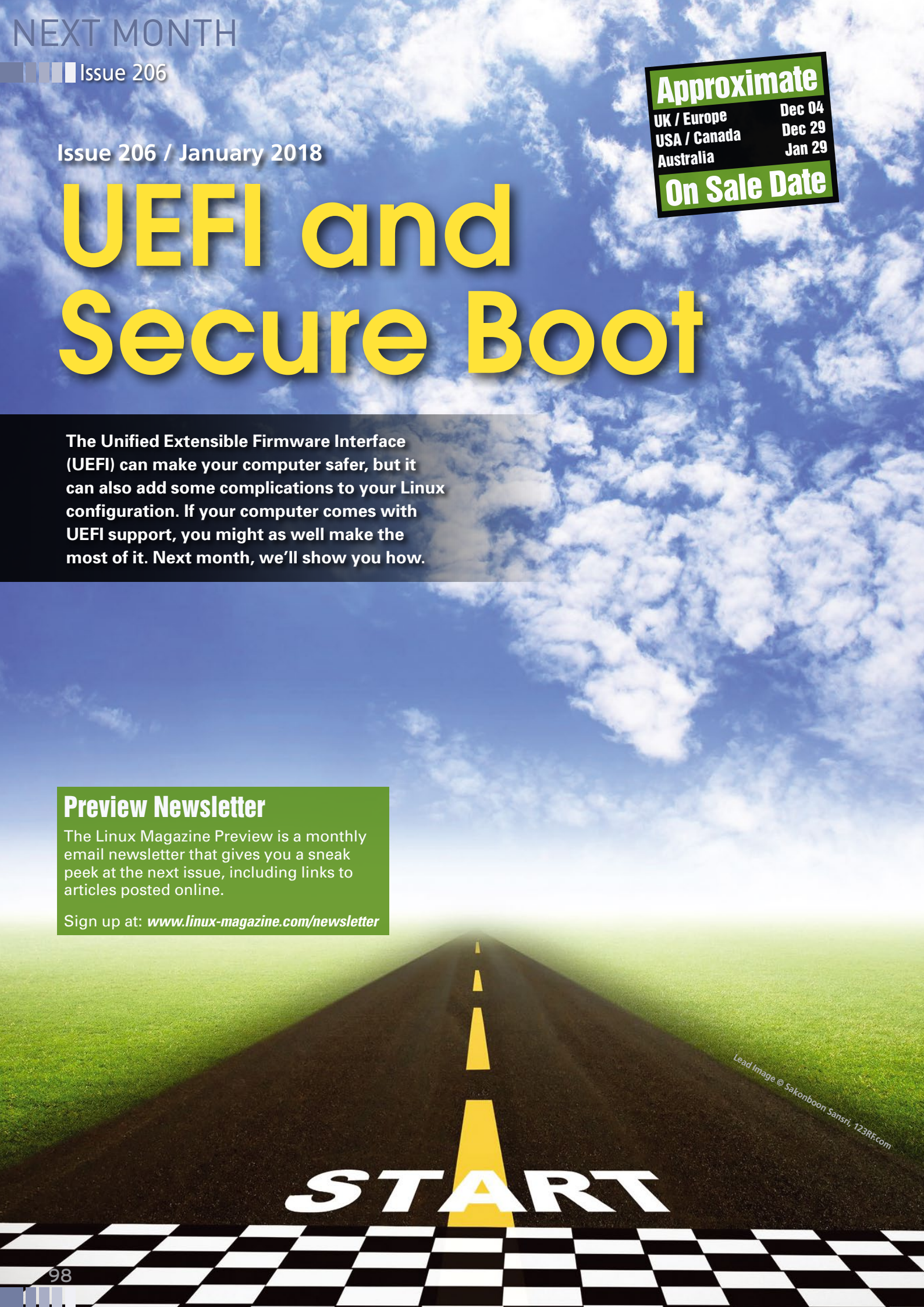
UEFI and Secure Boot

The Unified Extensible Firmware Interface (UEFI) can make your computer safer, but it can also add some complications to your Linux configuration. If your computer comes with UEFI support, you might as well make the most of it. Next month, we'll show you how.

Preview Newsletter

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: ***www.linux-magazine.com/newsletter***



THE COMPLETE LINUX VOICE

ARCHIVE DVD: ALL 32 ISSUES!

3,400 PAGES OF open source LOVE

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

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

Order now!

shop.linuxnewmedia.com

SUPERMICRO®

MicroBlade®

Highest-Density • High-Performance • High-Efficiency • Cost-Effective
Enterprise, Data Center, Web Applications, HPC and Cloud Computing Solutions

6U MicroBlade

3U MicroBlade

- Supports Intel® Xeon® Scalable Processors
- Best density and power efficiency with up to 56 UP or 28 DP server nodes
- M.2 NVMe SSD and HW SAS3 RAID support
- Ethernet switches supporting 1G, 2.5G and 10G downlinks with 96% fewer cables
- Redundant 2200W Titanium Level (96% efficiency) digital power supplies
- Battery Backup Power (BBP) modules with integrated power supply and cooling

intel XEON PLATINUM inside™

We Power the Cloud

SUPERMICRO X11 SERVER PLATFORMS

TITANIUM

nvm EXPRESS

SUPERMICRO®

Intel Inside®. Powerful Productivity Outside.

Learn more at supermicro.com/X11