FREE DVD

UBUNTU 17.10
GNOME IS BACK IN THE GARDEN

INSIDE A COMPILER

LINUX PRO MAGAZINE

FEBRUARY 2018

C++ CODERS
We compare GCC, Clang, and MSVC

INSIDE A COMPILER
What really happens to that code

What's New in LibreOffice 5.4

Brown Dog Gadgets
More fun with science

Neural Network
Can a program play the Monty Hall game?

GUI Firewall Tools
Airtight security in an easy view

Catch a Fox!
Monitor wildlife traps with a microcontroller

Guetzli
JPEG-quality images with a tiny file size

LINUXVOICE

- Bd, autojump, and Fasd – Cool tools for command-line navigation
- SystemRescueCd: Compact Emergency Kit
- maddog: Your Own Personal Cloud
- Why Has Docker Adopted Kubernetes?

FOSSPicks
- Audacity 2.2.0
- KTouch Tutor
- Console Candy

Tutorial
- Video Streams in FFmpeg

Issue 207
Feb 2018
US$ 15.99
CAN$ 17.99

WWW.LINUXPROMAGAZINE.COM

open
build
service

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

openbuildservice.org

openSUSE®

BETTER HEALTH

Dear Reader,

It is hard to think of writing about anything this month but net neutrality. On November 21, the US Federal Communication Commission (FCC) decided on a 3-2 party-line vote to end the requirement that service providers treat all traffic equally, opening the door to "pay to play" scenarios, where ISPs can shake down Internet companies for access to users.

Much has been said about the end of net neutrality, and it is hard to know how to contribute something that other commentators are not already talking about. I am one who has often expressed dismay at the current state of the Internet, where huge amounts of venture capital flows in with the goal of "doing business" and the primary form of business is spying on people. What is perhaps most striking about the FCC's decision is that it doesn't fix any of these real problems with the Internet and, instead, appears to manufacture a whole set of new problems.

But what is the Internet? What we call "the Internet" is generally thought to have grown from a network connecting a number of universities and research facilities in the 1970s and 1980s, which is correct. But somewhere in the 1990s, there was a concerted effort (at least in the US) to commercialize the Internet, with big companies investing in building cables to carry Internet traffic. The original backbone network no longer exists – today's Internet is a decentralized network of wires owned by different giant companies. Wires, routers, computers, protocols, name servers, secret contracts, switching facilities – the whole hangs together as a unified thing simply because we choose to think about it as a single logical entity, and the definition of that entity is very much subject to debate and manipulation.

The reason why a single regulatory body like the FCC can wield so much influence over our culture is that we have never ever really bothered to define what the Internet is, so people with money and power can race in to define it in a way that increases their money and power.

When the FCC passed the rules protecting net neutrality in 2015, I was surprised by the number of emails our office received – form letters sent as press releases from advocacy organizations – proclaiming the action was socialism or communism, or a government takeover of private enterprise. You can attract a ready-made horde of angry citizens by invoking such themes, which are deeply embedded in our politics and culture. Less acknowledged is that the other side basically does the same thing with their rhetoric about "freedom" and "the right to innovate." (Internet giants like Google and Facebook are also big businesses, with their own shareholders, and if you did something to threaten their livelihood, like, say, suggesting that users should be compensated for surrendering their privacy, they would not hesitate to call you a communist.)

I agree with nearly everyone in the IT industry that forcing Internet companies to pay tribute money to Internet service providers will definitely raise prices, and it will certainly result in less innovation. Where I disagree is with the presumption that, by stopping the end of neutrality, you have "saved the Internet." Preserving net neutrality is just one of many things that would need to happen before you could safely pronounce the Internet "saved."

In addition to the spying, the bullying, the ransomware, and the election tampering, several business questions loom over the Internet that we haven't even begun to address. If the Internet will not be regulated by "market forces," how exactly will we regulate it? Is it fair for one company to suck up as much bandwidth as it can sell and expect everyone else to just move over and slow down? On today's Internet, for instance, in North America, 36% of all traffic goes to one vendor: Netflix. What if that were 50% or 70%? Sooner or later, you would be paying more for Internet service so the network can make room for this additional Netflix traffic – even if you don't subscribe to Netflix [1]. Is Internet Freedom really about the freedom to watch TV shows? If you're going to call the Internet a utility, you need to regulate it as a utility, or we'll end up with the worst of both worlds.

Net neutrality? Yes, of course! But don't kid yourself. That's just the beginning of what we need to do to fix the Internet.

Joe

Joe Casad,
Editor in Chief

Info

[1] Top Sites by Percentage of Downstream Internet Traffic in North America: *http://fortune.com/2015/10/08/netflix-bandwith/*

LINUX MAGAZINE

WHAT'S INSIDE

This month we take a look inside one of the most deeply complex and unfathomable tools in the programmer's toolkit: the compiler. We also compare the GCC, Clang, and MSVC compilers, with the emphasis on support for C++ standards. Other highlights include:

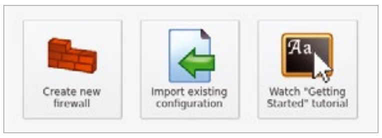
- **Ubuntu 17.10** – Ubuntu gives up the Unity desktop and returns to Gnome. How's that working out? (page 30).
- **Neural Network** – This month's Programming Snapshot puts Python to work on a classic brain teaser. (page 42).

Check out LinuxVoice for alternative directory navigation commands and system rescue with SystemRescueCD.

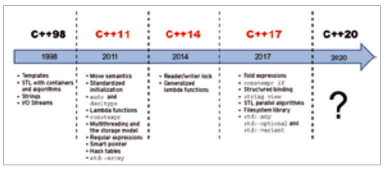
SERVICE

NEWS

REVIEWS


Create new firewall
Import existing configuration
Watch "Getting Started" tutorial

COVER STORIES



```
int main(){

  auto begin = std::chrono::steady_clock::now();
  for (size_t i = 0; i <= tenMill; ++i){
    MySingleton::getInstance();
  }
  std::chrono::duration<double> res= std::chrono::steady_clock::now() -

  std::cout << res.count() << std::endl;

}
```

TWO TERRIFIC DISTROS
DOUBLE-SIDED DVD!

SEE PAGE 6 FOR DETAILS

On the DVD

Linux Mint 18.3 Cinnamon (64-bit)

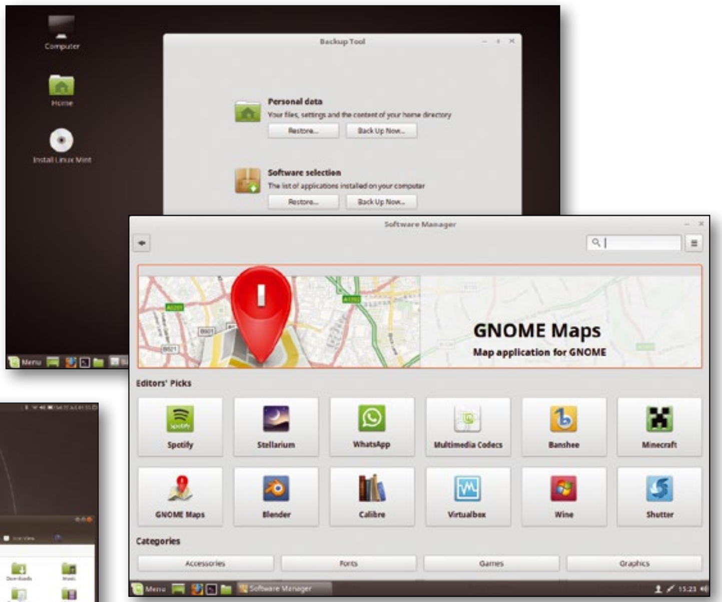
Linux Mint "Sylvia" [1] features updated software and user interface refinements [2]. As a long-term support (LTS) release, it will be supported until 2021. The Software Manager now runs in user mode, is much lighter, and launches three times faster than before. Flatpak applications are fully supported; they run in their own environment in isolation, and they do not affect the rest of the operating system. Backup Tool was almost completely rewritten and also runs in user mode. Other improvements include:

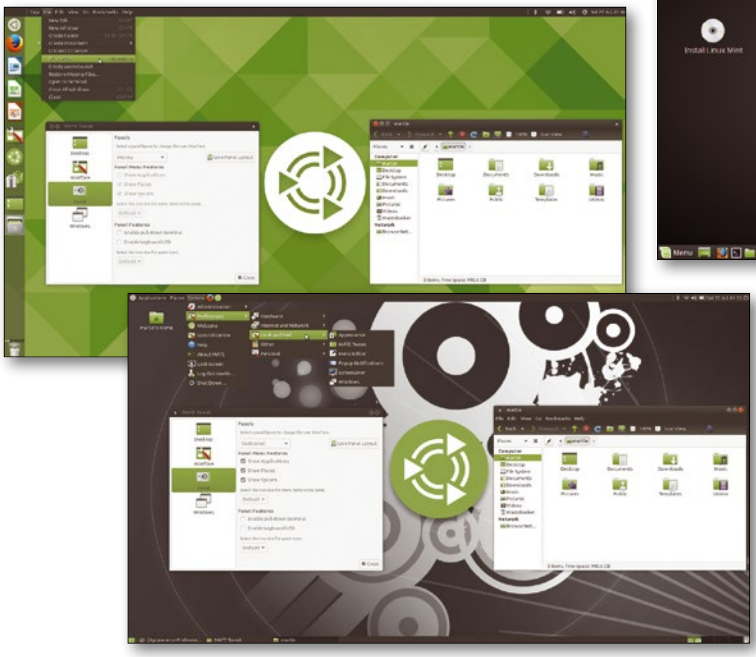
- Modern, cleaner, more consistent user interface
- Gnome Games 3.26 with its own GTK 3.26 environment
- Timeshift for system snapshots

Ubuntu Mate 17.10 (32-bit)

The developers claim that Ubuntu Mate 17.10 is "by far the best release we've ever produced" [3]. Mate offers panel layouts to suit any need, including Mutiny, which mimics the Unity layout; Cupertino, which mimics Mac OS; Redmond, for that Windows look; Traditional, the oldie but goodie default panel; and three more. Other features include:

- Improved global menu
- Complete Super key support in several panel layouts
- Heads-up display (HUD)





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

Additional Resources

[1] Linux Mint 18.3: *https://blog.linuxmint.com/?p=3457*

[2] What's new in Linux Mint: *https://www.linuxmint.com/rel_sylvia_cinnamon_whatsnew.php*

[3] Ubuntu Mate 17.10: *https://ubuntu-mate.org/blog/ubuntu-mate-artful-final-release/*

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

KubeCon Concluded in Austin, Texas

Kubernetes has become the Linux of the cloud. It has seen massive adoption in the last three years. The first release of Kubernetes was announced in 2014. All three major cloud providers, including Google (the creator of Kubernetes), Microsoft, and AWS now support Kubernetes. Even Docker started offering Kubernetes as an orchestrator along with its own orchestrator Swarm. Cloud Foundry has adopted Kubernetes as Cloud Foundry Container Runtime, and OpenStack vendors have already adopted Kubernetes to deploy OpenStack as an application. All major Linux vendors, including Red Hat, SUSE, and Canonical offer Kubernetes distributions.

The adoption and growth of Kubernetes was the theme of KubeCon, the Kubernetes conference that was held between December 6 and 8 in Austin, Texas. During the conference, Oracle open sourced its Kubernetes tools for serverless deployment and multicloud management.

Microsoft announced that Azure would bring new serverless and DevOps capabilities to the Kubernetes community, and Bitnami launched a new in-cluster Kubernetes Application Consol.

The Kubernetes community announced the 1.0 release of CoreDNS, a cluster DNS for Kubernetes. JFrog and Baidu joined Cloud Native Computing Foundation (CNCF), the home of Kubernetes, as Gold members.

Dell to Disable Intel's Insecure ME

The Intel vPro Management Engine (ME) came under fire recently when security researchers found serious bugs that allowed a remote attacker to take control of the affected systems.

"The exploitation allows an attacker to get full control over business computers, even if they are turned off (but still plugged into an outlet). We really hope by bringing this to light, it will raise awareness about security issues in firmware and avoid possible issues in the future," wrote Embedi, the security firm that discovered the bug.

Intel doesn't share any information about these "secretive" ME technologies. ME modules sit above the operating systems and users have no access or control over the technology. Organizations like Electronic Frontier Foundation (EFF) are calling for more transparency around ME modules. EFF asked Intel to "Provide a way for their customers to audit ME code for vulnerabilities. That is presently impossible because the code is kept secret."

© Chode, 123RF.com

Because Intel doesn't provide any such information, PC vendors and users don't have any means to audit or fix such vulnerabilities. Now one PC vendor has taken steps to protect its users. Dell is now disabling Intel ME in all new systems, and users will have to pay to enable the service.

In a statement to ExtremeTech, Dell said, "Dell has offered a configuration option to disable the Intel vPro Management Engine (ME) on select commercial client platforms for a number of years (termed Intel vPro – ME inoperable, custom order on Dell.com). Some of our commercial customers have requested such an option from us, and in response, we have provided the service of disabling the Management Engine in the factory to meet their specific needs. As this SKU can also disable other system functionality, it was not previously made available to the general public."

PC vendors, especially those selling Linux preloaded systems, are following suite and disabling ME by default. Dell is the biggest PC vendor, and if other vendors start disabling the engine, Intel might be compelled to either open source the technology or offer more transparency around it.

Linus Torvalds' Precious Advice to Security Experts

Linus Torvalds, the creator of the Linux kernel, is no fan of the security community. In his opinion security is just bugs that get exploited. "I don't trust security people to do sane things," said Torvalds, responding to a merge request by one of the top kernel developers Kees Cook.

What ticked Torvalds off this time was that Kees' patch had the potential to break things, and he added a fallback mode. Kees wrote, "This has lived in -next for quite some time without major problems, but there were some late-discovered missing whitelists, so a fallback mode was added just to make sure we don't break anything. I expect to remove the fallback mode in a release or two."

Torvalds refused to merge and said, "If you can make a smaller pull request that introduces the infrastructure, but that _obviously_ cannot actually break anything, that would be more likely to be palatable."

To which Kees responded, "This is why I introduced the fallback mode: with both kvm and sctp (ipv6) not noticed until late in the development cycle, I became much less satisfied it had gotten sufficient testing. I wanted to make sure there was a way for the series to land without actually breaking things due to any missed whitelists."

Torvalds said, "I'm not at all interested in killing processes. The only process I'm interested in is the _development_ process, where we find bugs and fix them."

But this time Torvalds has a valuable piece of advice for security people. He said that the primary focus should be "debugging" and making sure the kernel released in a year is better than the one released today. He dismissed the popular notion of kill processes for bugs. "… the hardening efforts should instead _start_ from the standpoint of 'let's warn about what looks dangerous, and maybe in a _year_ when we've warned for a long time, and we are confident that we've actually caught all the normal cases, _then_ we can start taking more drastic measures'," said Torvalds, "Stop this idiotic 'kill on sight, ask questions later'."

GPLv3 Comes to the Rescue of GPL Violators

Red Hat is working with major tech companies, including Facebook, Google, and IBM, to make it easier for GPL violators to cure violations. The companies are adopting the cure provisions of GNU GPLv3 to help companies fix violations.

One of the biggest concerns when using open source components in commercial products is licence compliance. Multiple efforts are made by organizations like the Linux Foundation to help companies consume open source software without worrying about compliance. However, in some cases, like VMware, organizations such as the Software Freedom Conservancy take aggressive routes that end up hurting collaboration and open source projects in question. Such legal actions also send a shock wave that touching open source can be dangerous.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

More Small Tools • Jeff Layton
We look at some additional tools that you might find useful when troubleshooting HPC systems.

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

Combining Containers and OpenStack Udo Seidel
The OpenStack cosmos cannot ignore the trend toward containers. If you want to combine both technologies, projects like Magnum, Kolla, and Zun come into play. Which one?

Take Your Pick from a Variety of AWS Databases • Steffen Grunwald
We look at the variety of databases available in Amazon Web Services – from relational, to NoSQL, to data warehouses for petabyte data volumes.

Manage Logical Volumes with GUI Tools Erik Bärwaldt
Linux uses the Logical Volume Manager to manage large hard drives and mass storage clusters efficiently. We look at various graphical tools that help serve up logical volumes and volume groups.

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

Cloud Orchestration with Chef James Stanger
Discover the difference between automation and orchestration, and learn to quickly deploy and orchestrate well-defined, scalable business solutions for your organization.

Common DevOps Mistakes • Erez Rusovsky
From industry metaphors, to agile processes, to DevOps, software development is evolving into a mature enterprise. We point out some missteps in the adoption of the DevOps methodology.

Companies don't violate licenses on purpose. "Most GPL violations occur by mistake, without ill will. Copyleft enforcement should assist these distributors to become helpful participants in the free software projects on which they rely," said Joshua Gay of the Free Software Foundation.

However, these companies must have ways to fix the violations. GPLv2, one of the most prominent copyleft licenses, permanently terminates permissions at the moment of violations. This heavy-handed approach discourages cooperation and collaboration and leads to more hostile resolutions like legal actions, in which no one, besides lawyers, has any interest. Linus Torvalds once said that "we lose" the moment we get lawyers involved.

With GPLv3, the Free Software Foundation has created an opportunity for users to address violations. It is a fix for the heavy-handed approach that GPLv2 adopted for violations. GPLv3 provides an opportunity to first time violators to restore all rights automatically once the violations are fixed. It was designed to attract more collaboration and amicable resolutions to violations instead of hostile actions.

Red Hat, Facebook, Google, and IBM are committed to extending the GPLv3 approach to license compliance errors to the software code that each licenses under GPLv2 and LGPLv2.1 and v2.

With the adoption of this balanced approach, companies will feel more comfortable using open source components in their products without the fear of prosecution for any mistaken violation.

"We felt strongly that the large ecosystems of projects using GPLv2 and LGPLv2.x would benefit from adoption of this more balanced approach to termination derived from GPLv3," said Red Hat in a blog post.

This step by Red Hat is a move in the right direction.

Linux Kernel 4.14 Released

Linus Torvalds, the creator of Linux, announced the release of Linux kernel 4.14 on November 12, 2017. The release was due earlier but was delayed because of an AppArmor patch that caused regression. Torvalds lashed out at a Canonical developer who found the AppArmor regression but said that it was not a big deal.

Torvalds responded and said, "As far as the kernel is concerned, a regression is THE KERNEL NOT GIVING THE SAME END RESULT WITH THE SAME USER SPACE. The regression was in the kernel. You trying to shift the regressions somewhere else is bogus SHIT. And seriously, it's the kind of garbage that makes me think your opinion and your code cannot be relied on. If you are not willing to admit that your commit 651e28c5537a ("apparmor: add base infrastructure for socket mediation") caused a regression, then honestly, I don't want to get commits from you."

Torvalds chose to delay the release instead of letting the regression go through.

Linux kernel 4.14 is expected to be the next LTS version. Greg Kroah-Hartman, the maintainer of the stable branch of the Linux kernel said, "So, here it is officially, 4.14 should be the next LTS kernel that I'll be supporting with stable kernel patch backports for at least two years, unless it really is a horrid release and has major problems. If so, I reserve the right to pick a different kernel, but odds are, given just how well our development cycle has been going, that shouldn't be a problem (although I guess I just doomed it now …)."

Some of the major highlights of the release include built-in HDMI CEC support for Raspberry Pi that allows users to control their Pi-powered devices from a single controller, as well as significant performance improvements in KVM, Xen, and Hyper-V. The release also improves EFI support, making it more secure and reliable.

OPEN SOURCE DATA
CENTER CONFERENCE

JUNE 12 – 13, BERLIN

CALL FOR PAPERS

until January 31

SIMPLIFYING
COMPLEX IT INFRASTRUCTURES
WITH OPEN SOURCE

OSDC.DE

20
18

Zack's Kernel News

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

By Zack Brown

How To Be a Maintainer

Tobin C. Harding posted a patch to create a new documentation file in the kernel source tree, describing how to be a kernel driver or subsystem maintainer. As with Linus Torvalds' requirements for a usable revision control system long ago, it's a little surprising that the information hadn't already been pulled together. Regardless, Tobin's done it now.

Most of the text was taken from a mailing list discussion between Greg Kroah-Hartman and Linus. Essentially, it describes the Git features and use cases that are most relevant to being a maintainer. For example, all patches need to be digitally signed to confirm that they're really coming from the person they claim to come from. So, to be a maintainer, you need to be able to set up a public key using GPG2 and configure Git to use it by default.

Tobin's doc goes on to say that you'll also need to be able to create pull requests – i.e., to let other maintainers know that you've got some code to share and how they can get it. This involves creating a new named branch that has all the changes you want to share in it. The name can be digitally signed, or not – maintainers differ on their willingness to pull unsigned branches from other maintainers. However, Linus will only pull signed branches into his tree.

Of crucial importance is the message that accompanies your pull request. This message goes into the Git project history and may need to be relied on in the future. As Linus says, "it should not just make sense to me, but make sense as a historical record too." Partly for those reasons, Linus and other maintainers may reserve the right to edit the text of any pull messages they receive, before sending them along to the actual tree.

Once you've got the branch the way you like it, you'll need to generate a pull request against the maintainer's tree you're sending it to. Presumably this is the same tree as yours, although perhaps in a slightly different historical state. The pull request reconciles those differences into something the recipient will be able to make sense of.

Finally, you submit the pull request in an email, just as if it were an ordinary patch.

That's as far as Tobin's doc went. Greg said it was an excellent beginning and proposed adding a section on how to set up Git on a given system.

Dan Williams also liked what Tobin had done so far and suggested two additional sections: one on how to "age" commits in the `-next` tree so that they could be culled or sent along to Linus at a certain point and another describing various techniques to avoid rebasing. In the Git world, rebasing is a way of cleaning up a tree in a certain way, but it also reshuffles patches, essentially rewriting history. When you're in the midst of feeding your patches up the

Author

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

food chain to a higher maintainer, this can mess with that maintainer's ability to see what actually happened in your project's history. The whole issue is a bit of a religious debate, but the Linux kernel has its own set of preferences, handed down from Linus.

Tobin replied that he'd like to read that section himself!

Meanwhile, folks like Mauro Carvalho Chehab offered their own nuanced suggestions for things like Git branch naming conventions (specifically, he felt the names didn't matter much).

It seems as though there is plenty of room for the doc to grow, and it'll probably do so in some fairly fascinating ways. It'll be a mix between specific Git use cases and oral history converted to firm policy. It's nice too, because anyone will be able to extrapolate the doc into their own open source projects.

Power-Up/Power-Down Control

Jon Hunter proposed replacing or extending the power management (PM) domain framework to be more flexible. The current system, GenPD, would let users associate a particular PM domain with a particular device. This domain would define a hierarchical structure of devices that would be powered on and off together, in a given sequence. The problem, Jon said, came when various pieces of hardware didn't necessarily need to be powered on and off together, although they might in certain circumstances. GenPD provided no way to create an alternative way of powering these devices up and down.

As an example, he said that the Tegra124/210 XUSB subsystem involved several pieces of hardware that were able to be used independently of each other. There was no specific need to power them up and down together, except that GenPD provided no alternative.

His proposal, he said, "extends the generic PM domain framework to allow a device to define more than one PM domain in the device-tree 'power-domains' property. If there is more than one, then the assumption is that these PM domains will be controlled explicitly by the consumer, and the device will not be automatically bound to any PM domain." More specifically, he said the new code would "add new APIs for GenPD to allow consumers to get, power on, power off, and put PM domains so that they can be explicitly controlled by the consumer. These new APIs for powering on and off the PM domains call into the existing internal functions, `genpd_sync_power_on/off()`, to power them on and off. To ensure that PM domains that are controlled [both] explicitly (via these new APIs) and implicitly (via `runtime-pm` callbacks) do not conflict, the PM domain `device_count` and `suspended_count` counters are used to ensure the PM domain is in the correct state."

Rajendra Nayak was glad to see this work being done and offered some technical suggestions. In particular, he wanted to be able to track the devices directly to check their power status, rather than relying on the state of an associated variable. He also suggested isolating the implementation from the interface – providing users with a "handle" to access a given device, rather than exposing the contents of particular data structures that might change in future patches.

Meanwhile Ulf Hansson wasn't sure adding a whole new set of API calls could be justified for what he felt might just be a few corner cases, but he acknowledged, "However, we currently know about at least two different SoCs [Systems on a Chip] that need this." So, he seemed to be on board with the idea that something needed to be done, even if it was more lightweight than Jon's original suggestion. He suggested that "it may be better to add some kind of aggregation layer on top of the current PM domain infrastructure" instead of an actual set of API calls.

Geert Uytterhoeven felt that there was probably enough relevant hardware to say that they weren't corner cases but deserved a more robust and flexible power-up/power-down approach.

Meanwhile, Rafael J. Wysocki pointed out that "the PM core takes only one set of PM operation[s] per device into account, and therefore, in every stage of system suspend, for example, the callback invoked by it has to take care of all actions that need to be carried out for the given device, possibly by invoking callbacks from other code layers. That limitation cannot be removed easily, because it is built into the PM core design quite fundamentally." He tended to believe that extending GenPD was not the way to go, "because power-on/power-off operations used by GenPD can be implemented in terms of lower-level power resource control, and I don't see the reason for mixing the two in one framework."

Jon was fine with this, because he felt that maybe GenPD should be a client sitting on top of whatever services his code would offer, "so that power domains are registered once and can be used by either method."

Nevertheless, the whole issue began to seem more complicated, because of Rafael's point about the inflexibility of the underlying PM infrastructure and the difficulty of retooling it to be more flexible.

At one point, Jon confessed that he didn't see a way forward, and Ulf wasn't sure either. There were a couple of alternatives suggested, but the thread petered out inconclusively.

Ultimately, it does seem as though something along the lines of this feature is needed, especially because, as Jon pointed out, his company needs the feature for Tegra hardware. Certainly other users are waiting in the wings, as well, so there'll definitely be a patch to support this at some point.

However, if the underlying kernel infrastructure is really as unfriendly as Rafael seemed to indicate, the whole project may end up being bigger than Jon or anyone else would have hoped.

Smoothing Out Disk Caching

Konstantin Khlebnikov wanted to change the way data is written to disk in Linux. Generally when you write to a file, the OS doesn't necessarily access the disk right away. Disk hardware tends to be slow, so there's a value in batching up writes in memory and dumping them all to disk at certain times. In general, this is all done highly efficiently, so you never actually notice it happening, but in a few cases, you might save a big file and then quickly cold boot the system, only to discover that the new data was never actually written to disk. That's one of the reasons why it's best to power down normally – it gives the system a chance to flush any remaining disk writes before powering off.

That's not what Konstantin was concerned with, though. Write caching is just par for the course and is one of the reasons the Linux user interface is so snappy and responsive. Konstantin's concern was that, although normal operations were indeed snappier with write caching, any task that required low disk latency would run into trouble, because the filesystem load would spike during cache writes, slowing down any other disk operations happening at the time.

He pointed out that although it was possible to tune the behavior of the Linux filesystem to some extent, there were not actually enough adjustable variables to influence this particular use case. The only other available option would be to run the filesystem in "sync mode," which would cause all disk writes to flush as soon as they occurred and would eliminate all caching, although the entire OS – especially the user experience – would grind and lurch.

He said:

This patch implements write-behind policy, which tracks sequential writes and starts background writeback when [they] have enough dirty pages in a row.

Write-behind tracks current writing position and looks into two windows behind it: [the] first represents unwritten pages, [the] second – async writeback.

Next write starts background writeback when [the] first window exceed[s the] threshold and waits for pages falling behind [the] async writeback window. This allows [it] to combine small writes into bigger requests and maintain optimal I/O depth.

Linus Torvalds replied:

This looks lovely to me.

I do wonder if you also looked at finishing the background write-behind at `close()` *time, because it strikes me that once you start doing that async writeout, it would probably be good to make sure you try to do the whole file.*

I'm thinking of filesystems that do delayed allocation etc. – I'd expect that you'd want the whole file to get allocated on disk together, rather than have the 'first 256kB aligned chunks' allocated thanks to write-behind, and then the final part allocated much later (after other files may have triggered their own write-behind). Think loads like copying lots of pictures around, for example.

I don't have any particularly strong feelings about this, but I do suspect that once you have started that I/O, you do want to finish it all up as the file write is done. No?

It would also be really nice to see some numbers. Perhaps a comparison of 'vmstat 1' or similar when writing a big file to some slow medium like a USB stick (which is something we've done very very badly at, and this should help smooth out)?

Jens Axboe was also thrilled with Konstantin's general idea, but he said, "My only concerns would be around cases where we don't expect the writes to ever make it to media. It's not an uncommon use case – [an] app dirties some memory in a file and expects to truncate/unlink it before it makes it to disk. We don't want to trigger writeback for those."

Regarding Jens' issue, Konstantin agreed that "this is [a] case where serious degradation might happen." He felt, though, that there might be some relatively simple workarounds to avoid the issue in most cases.

Dave Chinner also agreed that Konstantin's basic idea was excellent, although he felt it needed some tweaking. He offered a series of problematic use cases, saying, "rapid write-behind behavior might not significantly affect initial write performance on an empty filesystem. It will, in general, increase file fragmentation, increase interleaving of metadata and data, reduce metadata writeback and read performance, increase free space fragmentation, reduce data read performance, and speed up the onset of aging related performance degradation."

He added, "a write-behind default of 1MB is bordering on insane because it pushes most filesystems straight into the above problems. At minimum, per-file write-behind needs to have a much higher default threshold and writeback chunk size to allow filesystems to avoid the above problems."

He also suggested implementing "a small per-backing dev threshold where the behavior is the current write-back behavior, but once it's exceeded we then switch to write-behind so that the amount of dirty data doesn't exceed that threshold."

Linus really liked that idea, although he acknowledged, "part of the problem there is that we don't have that historical 'what is dirty', because it would often be in previous files. Konstantin's patch is simple partly because it has only that single-file history to worry about."

Elsewhere, Andreas Dilger chimed in with his own experience, saying, "Lustre clients have been doing 'early writes' forever, when at least a full/contiguous RPC worth (1MB) of dirty data is available, because network bandwidth is a terrible thing to waste. The oft-cited case of 'app writes to a file that only lives a few seconds on disk before it is deleted' is IMHO fairly rare in real life, mostly `dbench` and back in the days of disk based `/tmp`. Delaying data writes for large files means that 30s*bandwidth of data could have been written before VM page aging kicks in, unless memory pressure causes writeout first. With fast devices/networks, this might be many GB of data filling up memory that could have been written out."

The conversation petered out inconclusively, although I'd expect Linus' preferences to be treated like gold and subsequent versions of the patch to adhere pretty strictly to his and Dave's suggestions. It's not that Linus requires that, but in general, developers tend to want to make him happy, especially when his wishes are so clearly expressed. Back when people were trying to pry out his preferences for a revision control system, there was plenty of resentment when, for example, Tom Lord's old "arch" project would go for years in development and never get picked up for kernel management.

In terms of Konstantin's write-behind patches, it seems like something very similar to his original patch will be accepted fairly quickly, with more tweaks and enhancements going in as well. ■■■

SCALE 16x

The Sixteenth Annual
Southern California Linux Expo

Connecting World Wide Community for 16 Years

March 8th - 11th, 2018
Pasadena Convention Center
www.socallinuxexpo.org

How compilers work

Meticulous Transformer

Compilers translate source code into executable programs and libraries. Inside modern compiler suites, a multistage process analyzes the source code, points out errors, generates intermediate code and tables, rearranges a large amount of data, and adapts the code to the target processor.

By Tim Schürmann

Below the surface, a black box compiler handles complex processes that require good knowledge of machine theory and formal languages. Given the importance of compilers, it is not surprising that compiler construction is standard curriculum for computer science students. If you have never been to a college-level lecture on compiler theory – or if you went to the lecture but need a refresher course – this article summarizes the basics. (See the box titled "Teaching Material for Compiler Construction.")

In simple terms, a compiler goes through three steps: It parses the source code, analyzes it, and synthesizes the finished program (Figure 1).

In the first step, the compiler parses the source code character by character and tries to identify key-

Teaching Material for Compiler Construction

When students of computer science need to learn what holds the programming language world together, the best bet is to attend a lecture on compiler construction. Mathematics professor Peter Köhler held a lecture on compilers in the 2016 summer term at the University of Giessen [1], Germany. With his permission, *Linux Magazine* worked through his notes and summarized the most important points for this article.

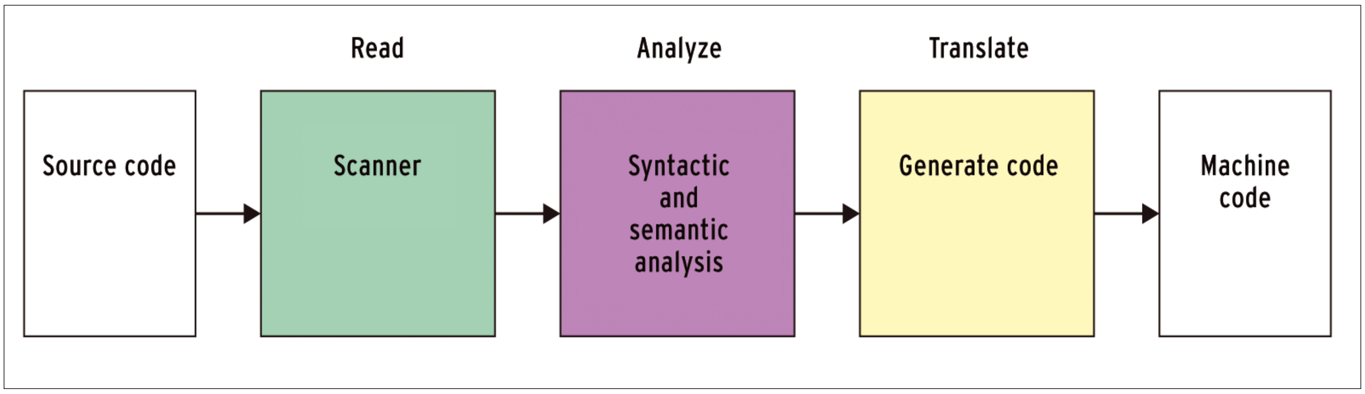


Figure 1: Rough structure of a compiler: parse code, analyze it, and create an executable program.

words, variable names, numbers, strings, and all other components – including comments. The scanner takes care of this lexical analysis.

In the second phase, another component checks the syntax and semantics of the program. For example, if a programmer uses a variable that has not yet been declared, or if a semicolon at the end of an expression is missing, the compiler will discover the problem in this phase.

Modern compilers distinguish between syntactic and semantic analysis. *Syntax* is all about structure, whereas *semantics* is focused on meaning. The component of the compiler called the parser tries to detect the syntax elements. If this phase is successful, the parser calls the semantic routine.

This delineation makes it possible to optimize tasks and analyze the source code more efficiently.

However, syntactic and semantic analysis turns out to be a non-trivial task, as a simple example shows: the compiler for a dumb calculator needs to parse a text file with a calculation expression. The calculator can only add and multiply numbers, where multiplication/division takes priority over addition/subtraction. If in doubt, brackets specify the sequence. For example, the following is allowed,

```
(1+3)*(4+5)
```

but this is not:

```
(1+)*4
```

The compiler must understand which of the two expressions is valid and which is not.

For the compiler to identify that the second expression as faulty, the compiler builder first tells the compiler how the correct source code is structured. This information is conveyed through rule sets. The compiler checks whether the source code follows the rules defined in the rule sets. In the example, a single number is obviously a valid expression. If two `x` and `y` expressions exist, then `x + y`, `x * y` and `(x)` are valid expressions.

The rules can also be written in shorthand. The Backus-Naur Form (BNF) is a very popular form (see Listing 1): The program (`program`) consists of an expression (`expr`). This expression, in turn, exists if it is either a `term` or if it is a `term` with the character `+` and then another expression after. The last rule is necessary because numbers can consist of several digits (`digits`).

These rules, known as production or derivation rules, form the grammar of the source language. Like English grammar, this grammar provides rules that build the programming language. Abbreviations, such as `term` or `factor`, are known as symbols. All symbols on the left-hand side are referred to as *non-terminals*, all others are *terminals*. Terminals would include `digit`, `+`, `*`, `)`, and `(`.

One programming language may include several grammars. The choice would depend on the language and requirements. The grammar used in the preceding example can lead to infinitely long expressions and programs because some rules have a recursive structure. The grammar used in the example is considered a context-free grammar (or type 2). A grammar is considered regular if all rules follow one of the two forms:

```
U ::= t
U ::= Vt
```

In this case, `t` is a terminal character; `U` and `V` are non-terminal characters.

Top-Down Parser

To the delight of developers, a grammar can be used to create a compiler quite elegantly: For all non-terminals, the developer creates a function that takes care of the corresponding evaluation. If necessary, the function calls on other functions to help. For example, the function `expr()`, which belongs to `expr`, has a structure like the one briefly outlined in Listing 2.

Listing 1: Backus-Naur Form

```
program ::= expr
expr ::= term | term + expr
term ::= factor | factor * term
factor ::= number | ( expr )
number ::= digit | digit number
```

Listing 2: expr()

```
int expr() {
        int value = term();
        if (File.ReadNextChar() == '+') value += expr();
        return value;
}
```

According to the second rule in Listing 1, a correct expression such as 1+2 always starts with a term. In Listing 2, the expr() function includes term(), which still needs to be implemented for an evaluation in the first step. In this example, term() returns 1.

In the second step, expr() then looks at the next character in the source code. If the next character is a +, the second rule of Listing 1 requires an expression to follow, which is why expr() calls itself. The result is that expr() returns 2, which the compiler also adds to the previously computed intermediate result. The compiler would then have arrived at the end of a valid expression, which is why expr() returns the calculated value in the example.

At this point, a mature compiler could generate a suitable instruction in machine code. The developer generates functions according to the same principle for all other non-terminals. If you then apply program() to the source code file, you automatically get the translated program – in the example, this would be the calculated number. Because processing starts with program, computer scientists also call this symbol the start symbol.

The calls to the individual functions can be displayed as a tree in this scenario: the syntax tree.

Bottom-Up Parsers

In the process described so far, the parser simply proceeds from the start symbol (in the example program) and then tries to find appropriate grammar rules for each character that has been read. Such parsers are known as top-down parsers.

Other parsers are bottom-up parsers and proceed according to an opposite scheme: They try to replace the source code with grammar rules (reduction) until only the start symbol remains. In practice, such bottom-up parsers usually use an empty stack at the beginning, into which they dump all previously detected, non-terminal characters and all the characters they read. After each new character is read, they check whether the topmost elements on the stack match the right side of a grammar rule. If this is the case, parsers take the elements from the stack and place the non-terminals from the left side of the identified grammar rule on top. The whole process runs until the start symbol is left and the end of the source code is reached. If that doesn't work, there's probably a mistake. A bottom-up parser that works according to this pattern is also known as a shift-reduction parser.

The programming of the scanner and parser is a straightforward consequence of the grammar of the language. This process can also be automated: Tools such as Flex [2] or Bison [3] automatically generate the source code for matching scanners and parsers from the grammar (Figure 2).

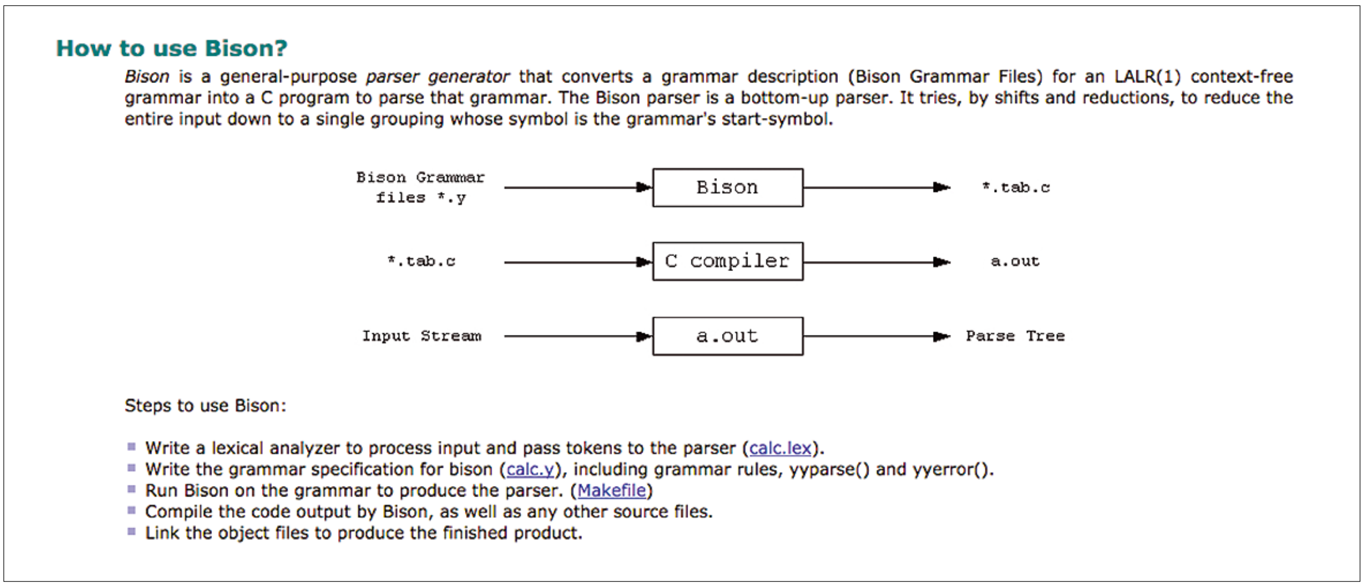
Whenever the parser needs more information, it requests it from the scanner. In the example in Listing 2, the expr() function would thus not read the next character itself; instead, File.ReadNextChar() simply consults the scanner.

Scanner Internals: Machines

The source code of most programming languages consists of numbers, variable names, keywords, operators, and separators (delimiters). Names and keywords are composed of several letters and characters.

A finite-state machine helps the scanner detect int, &&&, and all other components of the language. Imagine it like a ticket vending machine: The customer throws in 20 cents, and the machine changes its status to "20 cents paid." If the customer inserts a dollar, the machine then changes the status to "120 cents paid". This continues until the machine reaches the "price paid in full" status. Similarly, you can make a machine that accepts letters from the source code. The machine switches each character read to a different status: If the scanner has read an i, it changes to the state "i read"; if an n is added, it changes to the state "n read." If the t follows, the machine has finally read the whole term int and changes its status to "int identified."

This machine has an initial status and at least one final status. The machine for detecting int has detected the initial "no character read yet" status and the two final "int recognized"


How to use Bison?

Bison is a general-purpose *parser generator* that converts a grammar description (Bison Grammar Files) for an LALR(1) context-free grammar into a C program to parse that grammar. The Bison parser is a bottom-up parser. It tries, by shifts and reductions, to reduce the entire input down to a single grouping whose symbol is the grammar's start-symbol.

Steps to use Bison:

- Write a lexical analyzer to process input and pass tokens to the parser (calc.lex).
- Write the grammar specification for bison (calc.y), including grammar rules, yyparse() and yyerror().
- Run Bison on the grammar to produce the parser. (Makefile)
- Compile the code output by Bison, as well as any other source files.
- Link the object files to produce the finished product.

Figure 2: Bison is a well-known parser generator for the Linux environment (from the Bison Tutorial by Lan Gao (*http://alumni.cs.ucr.edu/~lgao/teaching/bison.html*).

and “not the keyword int” statuses. When the final status occurs, computer scientists say the machine has accepted the word `int`.

You can also envision a scenario in which boxes (nodes) represent the individual states (Figure 3). The start and end states are highlighted accordingly. Arrows (edges) indicate from which status the machine transitions to another state. The transition will only take place if the machine has read one of the characters written on the arrow.

A regular grammar can be used to construct not only a parser, but also a finite-state machine that accepts the language of the grammar: First, create a status for all non-terminals, plus the start and end statuses. Then, when it reads a character `t`, it makes a transition from the `s` state to the `n` state if there is a grammar rule in the `n ::= st` form. Incidentally, this process creates a bottom-up parser for the grammar.

In a simple scenario, you could read the next character and check, with `if` queries or a `switch` construction, which charac-

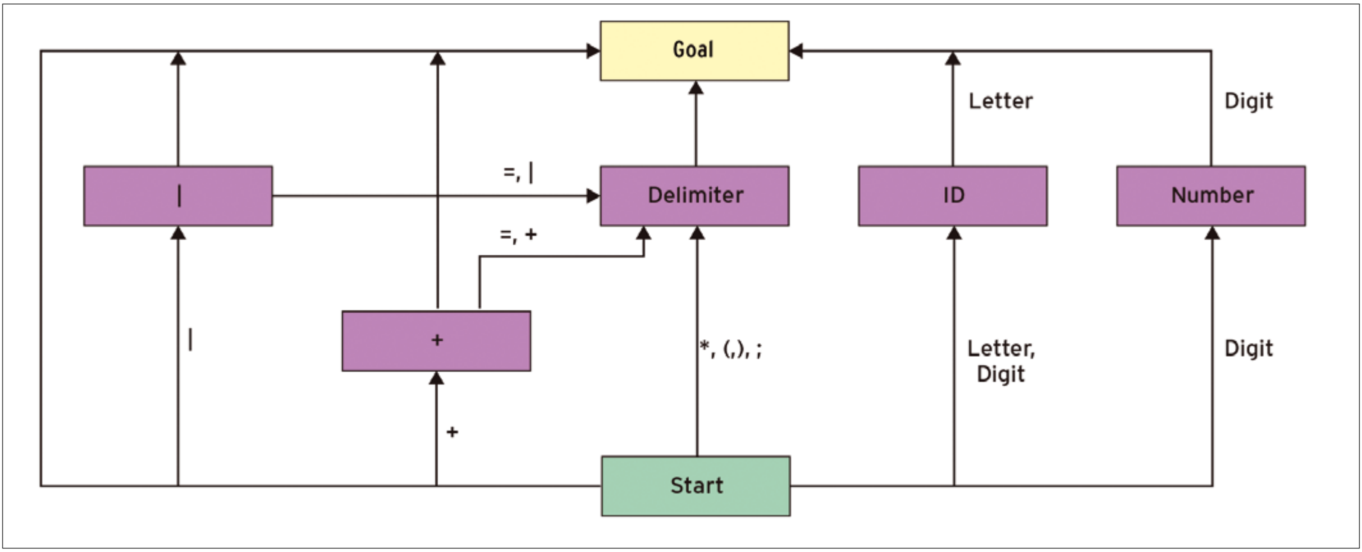


Figure 3: The machine changes status as soon as it encounters a character, digit, or symbol.

ter it is and finally note, in the form of a variable, the new state to which the machine has changed.

Once the scanner has identified the next element in the source code (such as the keyword `int`), it replaces the element with a symbol and returns it to the parser. The aforementioned Flex tool also uses the strategy with finite-state machines.

Symbol Table

To map the source code compactly in memory, the scanner replaces each recognized keyword, each variable name, and all other elements with a symbol. You could replace the rather long variable names with two numbers: The first number is used as a substitute for the variable name. The second number specifies the row in a table, the symbol table, which contains the variable name used by the developer. During syntax and semantics analysis, additional information, such as the type of variable, appears in the symbol table.

To find an entry in the symbol table as quickly as possible, many compilers run the variable name through a hash function that can be calculated quickly. This step spits out a number, which the compiler then uses as an index in the symbol table. If the compiler encounters the variable name again later on, it simply calculates the hash value and immediately gets the location in the symbol table with all important information about the variable – such as its type. The compiler does not have to search through the entire table.

Both the scanner and the semantic routine generate a number of tables in addition to the symbol table. Among other things, these tables contain the nesting structure for the loops and the loop variables. The compiler repeatedly accesses the information in the tables, even later on.

Internal Representation of the Source Code

The scanner passes the symbols that it has determined to the parser. The syntax and semantic analysis ends with an internal representation of the source code. The compiler and the languages are responsible for what this representation looks like. The program could be present as a syntax tree or in Polish notation. Many compilers also use quadruples, for example, from the `A = B + A` statement it would be:

```
+,      B,      A,      T1
=,      T1,             A
```

`T1` is a temporary variable created by the compiler.

So far, the compiler has only analyzed the source program. For this reason, experts also refer to this first phase as the analysis phase.

Generating Code

In the next step, another component optimizes the internal presentation. As a rule, the compiler optimizes the run time and assigns memory locations to the variables. In the preceding example, the compiler would try to eliminate the temporary variable `T1`.

In the last phase, the compiler finally generates the executable machine code. Generally, programmers call this object code or simply code. Under Linux, it is usually either a (dynamic) library or the executable program.

Some compilers also produce assembler code, which is then converted into machine language by a downstream assembler. The compiler could generate the following code from `A = A + B`:

```
lda a   ; load a in the accumulator
add b   ; add b to the accumulator
sto a   ; store accumulator to a
```

Control structures such as `if`, `while`, and `for` can usually be mapped with the jump instructions of the processor. Complex loops, such as `for`, may optionally replace a (longer) `while` loop.

Because it knows the processor instruction set, and the information from the tables, the compiler also makes the code more compact. The stack is used for function calls: Before starting a function, the compiler dumps its arguments and the return address onto the stack. The processor then performs the function. Finally, the compiler has to clean up the stack; current processors use special commands to support the compiler in this task.

Conclusion

This article offered a brief look at how compilers parse and prepare source code. Keep in mind that not all compilers adhere to the approach described in this article. For example, one-pass compilers do without an internal representation of the code and generate the object code in a single pass. Also, the steps presented in this article sometimes do not occur in distinct phases but are, instead, interwoven. ■■■

Interpreter, Assembler, and Translator

Unlike the compiler, an interpreter reads the source code and executes it directly; thus, no object code is generated. The classic interpreters analyze each command in the source code one after another.

Modern interpreters convert the complete source code into a special optimized internal representation. The interpreter then executes this intermediate or byte code much faster. Sometimes a just-in-time compiler translates the internal representation into machine language, which further increases the execution speed. Java uses this procedure.

An assembler is a special form of the compiler that translates a program into assembler or machine language. Since assembler is usually a symbolic representation of the machine commands, both languages are similar. The generic term translator usually refers to all three (i.e., compilers, assemblers, and interpreters).

Info

[1] Compiler construction lecture notes (in German): *http://www.staff.uni-giessen.de/~gc1079/*
[2] Flex: *https://github.com/westes/flex*
[3] Bison: *https://www.gnu.org/software/bison/*

ADMIN
Network & Security
Container Security
Keep intruders out of your Docker containers
WWW.ADMIN-MAGAZINE.COM

LINUX PRO MAGAZINE
POWER MANAGEMENT
Keep your laptop running longer
WWW.LINUXPROMAGAZINE.COM

GCC, Clang, and MSVC compilers with C++

Perfect Match

Due to the fast pace of updates of the C++ standards, compiler builders have been busy. How do the most popular compilers fit with the standards and what are the differences? *By Rainer Grimm*

The quiet times are over for C++. A full 13 years passed between the C++98 and C++11 standards, but since then, new standards have appeared every three years, with C++14, C++17, and preliminary work on C++20. The C++ standardization committee already shows signs of enthusiasm for the next cycle.

With so many versions of C++ out in the world (see the "Blessings of Diversity" box), a developer could easily wonder, how do the compiler makers keep up with it all? This article looks at support for C++ standards in three popular compiler alternatives:

- GCC: The GNU Compiler Collection (GCC) [1] is the quintessential free software compiler. Originally created by Richard Stallman in 1987 as the compiler for the GNU project, GCC is now supported by a large community of developers and is found on almost all Linux distributions.
- Clang: This popular free compiler collection [2] uses the LLVM compiler as a back end. Clang is actually the front-end component. Clang/LLVM is designed for a high degree of compatibility with GCC. The Clang project enjoys the support of several major vendors, including Apple, Google, Microsoft, Intel, and AMD, possibly because they find Clang's permissive free software license easier to integrate with commercial projects than the GPL3 license attached to GCC.

Blessings of Diversity

Some may wonder why so many different compilers exist. The fact that many computer architectures exist is only half the answer. The rest of the answer is that different compilers have different distinguishing features.

All three major compilers run on the x86 architecture, and each of the three has a special area in which it impresses. The Clang compiler scores points with understandable error messages and an open architecture that supports many powerful tools. GCC is distinguished by the fact that it supports most hardware architectures and is consequently gaining in importance in the embedded world with the GNU ARM Embedded Toolchain [4]. For MSVC, integration with Visual Studio adds massive value for many C++ developers.

Ultimately, this diversity in compilers is a blessing for all C++ developers. Each of the three compilers sets standards and, at the same time, sets the bar higher for the others. The fact that the compilers continue to compete for developers' favor will ultimately benefit the developers at the end of the day.

Author

Rainer Grimm is a trainer, a seminar lecturer on modern C++ and Python, and a published author, including *C++*, *C++11 für Programmierer*, and *C++-Standardbibliothek* (O'Reilly), as well as *The C++ Standard Library* and *Concurrency with Modern C++* (Leanpub).

- MSVC: This compiler is for Microsoft's Visual C++ (MSVC) environment [3]. In the past, Linux support in a major Microsoft development tool would have been unthinkable, but today, versions of MSVC run on Linux, and extensions are available for developing Linux applications on MSVC.

In this article, you'll also learn about some online compiler front ends that let you test your code for different compilers and standards, which often leads to valuable insights, optimizations, and troubleshooting tips.

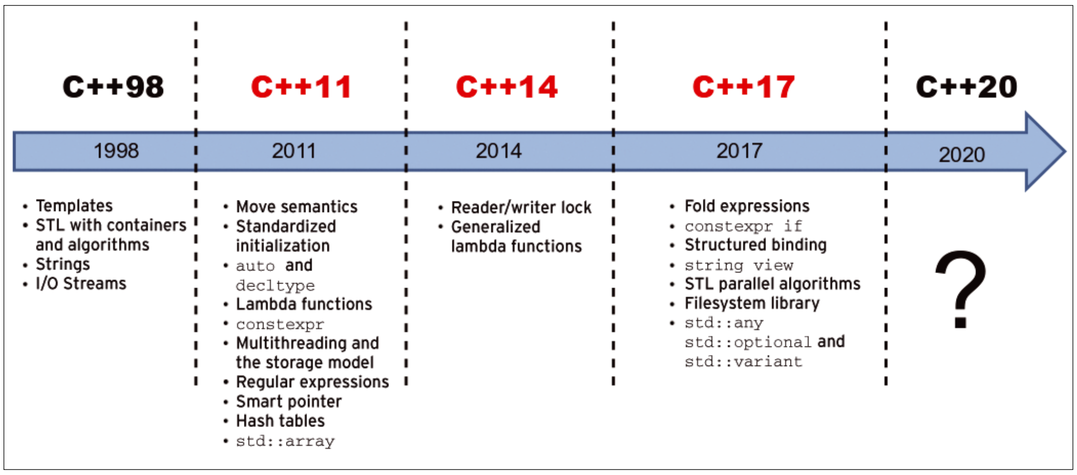


Figure 1: The C++ standard evolves and extends with each new version.

Compilers and Standards

An estimated 95 percent of all C++ developers rely on the GCC, Clang, or MSVC compilers. The timeline in Figure 1 shows the existing C++ standards, with the exception of C++03, which was only released as a bug fix for C++98 in 2003. This article focuses on C++11, C++14, and C++17. I will avoid looking back (C++98) or into the future (C++20).

Table 1: Compiler Support

C++ Compiler	C++11	C++14	C++17
GCC	4.8.1	5.0	7
Clang	3.3	3.4	5
MSVC	19.0	19.1	19.1(in part)

Table 1 shows which compiler versions support the C++11, C++14, and C++17 standards.

The table refers to support for the C++ core language. Sometimes that support is only partially complete. MSVC 19.1, for instance, only partially supports C++17, and this support requires the third update. Tables 2 and 3 show all the details of C++17 support [5].

If you are not satisfied with the information provided on the big three compilers, or if you are interested in the C++ compilers that other Unix platforms have to offer, you can also find more detailed information on the current C++ standards at the C++ Reference website [5].

Support for the C++ standard library (STL) varies a little. There is currently no implementation of the parallel STL in C++17. You have to do some tinkering work to find a solution: The High Performance Parallex (HPX) library [6], for instance, is a framework for parallel and distributed applications that already implements the parallel STL.

Table 2: C++17 Support (Part 1)

C++17 Features	Paper	Version	GCC	Clang	MSVC
New auto rules for direct-list-initialization	N3922	c++17-lang	5	3.8	19.0
`static_assert` with no message	N3928	c++17-lang	6	2.5	19.1
Typename in a template template parameter	N4051	c++17-lang	5	3.5	19.0
Removing trigraphs	N4086	c++17-lang	5.1	3.5	16.0
Nested namespace definition	N4230	c++17-lang	6	3.6	19.0
Attributes for namespaces and enumerators	N4266	c++17-lang	4.9 (Namespaces), 6 (Enumerators)	3.6	19.0
`u8` character literals	N4267	c++17-lang	6	3.6	19.0
Allow constant evaluation for all non-type template arguments	N4268	c++17-lang	6	3.6	no
Fold expressions	N4295	c++17-lang	6	3.6	no
Remove deprecated use of the register keyword	P0001R1	c++17-lang	7	3.8	19.1
Remove deprecated `operator++` (Bool)	P0002R1	c++17-lang	7	3.8	19.1
Removing deprecated exception specifications from C++17	P0003R5	c++17-lang	7	4	no
Make exception specifications part of the type system	P0012R1	c++17-lang	7	4	no
Aggregate initialization of classes with base classes	P0017R1	c++17-lang	7	3.9	no
Lambda capture of `*this`	P0018R3	c++17-lang	7	3.9	19.1
Using attribute namespaces without repetition	P0028R4	c++17-lang	7	3.9	19.1
Dynamic memory allocation for over-aligned data	P0035R4	c++17-lang	7	4	no
Unary fold expressions and empty parameter packs	P0036R0	c++17-lang	6	3.9	no
`__has_include` in preprocessor conditionals	P0061R1	c++17-lang	5	yes	19.1

Important Compiler Flags

In order to use the correct C++ standard, you need to specify it for GCC or Clang. Both support the same flags. Thus the

```
g++ -std=c++11 dataRace.cpp
```

call compiles the `dataRace.cpp` file source code according to the C++11 standard. You can also specify C++14 or C++17. This step is not necessary for newer versions of GCC and Clang, but not all compiler-to-C++ standard combinations can cope without this specification, and it will not do any harm to specify. MSVC does not require you to specify the C++ standard.

Listing 1: A Simple Race Condition

```
#include <thread>

int main(){

  int globalVar{};

  std::thread t1([&globalVar]{ ++globalVar; });
  std::thread t2([&globalVar]{ ++globalVar; });

  t1.join();
  t2.join();

}
```

There are some other interesting compiler flags for developers. For example, `-O3` (for GCC and Clang) and `/Ox` (for MSVC), for example, generate a maximum optimized program, and `-Wall` (for GCC and Clang) or `/Wall` (for MSVC) set the maximum warning level.

Sanitize

Sanitizer [7], a tool that checks whether the code correctly uses addresses, threads, and memory, has been available since GCC 4.8 and Clang 3.2. MSVC users will have to try their luck with a Windows port [8].

A small example will help illustrate the power of the Thread Sanitizer [9]. The program in Listing 1 contains code that can clearly lead to a race condition, since several threads access the `globalVal` variable at the same time. More precisely, both threads try to change the variable at the same time.

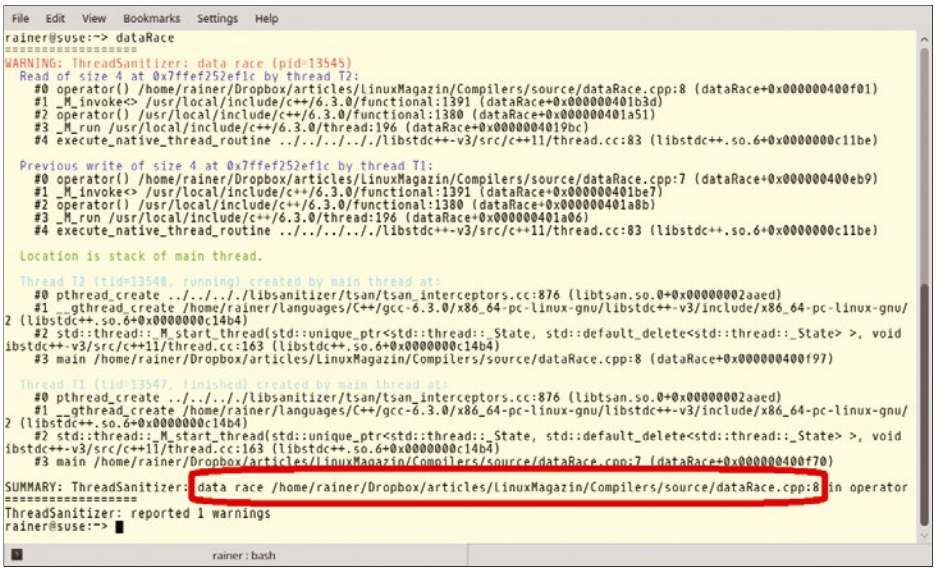
Compile Thread Sanitizer into GCC and Clang by using the `-fsanitize=thread` flag:

```
g++ -std=c++11 dataRace.cpp  -fsanitize=thread -pthread ↩
-g -o dataRace
```

then run the program compiled here with GCC; it will generate output that identifies the potential race condition (Figure 2). At the end, the output also shows the sequence of the built-in data race.

Table 3: C++17-Support (Part 2)

C++17 Features	Paper	Version	GCC	Clang	MSVC
Template argument deduction for class templates	P0091R3	c++17-lang	7	5	no
Non-type template parameters with auto type	P0127R2	c++17-lang	7	4	no
Guaranteed copy elision	P0135R1	c++17-lang	7	4	no
New specification for inheriting constructors (DR1941 et al)	P0136R1	c++17-lang	7	3.9	no
Direct-list-initialization of enumerations	P0138R2	c++17-lang	7	3.9	19.1
Stricter expression evaluation order	P0145R3	c++17-lang	7	4	no
`constexpr` lambda expressions	P0170R1	c++17-lang	7	5	19.1
Differing begin and end types in range-based for	P0184R0	c++17-lang	6	3.9	19.1
`[[fallthrough]]` attribute	P0188R1	c++17-lang	7	3.9	19.1
`[[nodiscard]]` attribute	P0189R1	c++17-lang	7	3.9	19.1
Pack expansions in using-declarations	P0195R2	c++17-lang	7	4	no
`[[maybe_unused]]` attribute	P0212R1	c++17-lang	7	3.9	19.1
Structured bindings	P0217R3	c++17-lang	7	4	19.1
Hexadecimal floating-point literals	P0245R1	c++17-lang	3	yes	no
Ignore unknown attributes	P0283R2	c++17-lang	yes	3.9	no
`constexpr if` statements	P0292R2	c++17-lang	7	3.9	19.1
Init-statements for `if` and `switch`	P0305R1	c++17-lang	7	3.9	19.1
Inline variables	P0386R2	c++17-lang	7	3.9*	no
DR: Matching of template template-arguments excludes compatible templates	P0522R0	c++17-lang	7	4	no
Standardization of parallelism TS	P0024R2	c++17			no
`std::uncaught_exceptions()`	N4259	c++17	6	3.7	19.0
Splicing maps and sets	P0083R3	c++17	7		no
Improving `std::pair` and `std::tuple`	N4387	c++17	yes	4	19.0
Improving `std::pair` and `std::tuple`	N4387	c++17	yes	4	19.0
Elementary string conversions	P0067R5	c++17			no
`std::string_view`	N3921	c++17	7	4	19.1



```
rainer@suse:~> dataRace
==================
WARNING: ThreadSanitizer: data race (pid=13545)
  Read of size 4 at 0x7ffef252ef1c by thread T2:
    #0 operator() /home/rainer/Dropbox/articles/LinuxMagazin/Compilers/source/dataRace.cpp:8 (dataRace+0x000000400f01)
    #1 _M_invoke<> /usr/local/include/c++/6.3.0/functional:1391 (dataRace+0x000000401b3d)
    #2 operator() /usr/local/include/c++/6.3.0/functional:1380 (dataRace+0x000000401a51)
    #3 _M_run /usr/local/include/c++/6.3.0/thread:196 (dataRace+0x0000004019bc)
    #4 execute_native_thread_routine ../../../.././libstdc++-v3/src/c++11/thread.cc:83 (libstdc++.so.6+0x0000000c11be)

  Previous write of size 4 at 0x7ffef252ef1c by thread T1:
    #0 operator() /home/rainer/Dropbox/articles/LinuxMagazin/Compilers/source/dataRace.cpp:7 (dataRace+0x000000400eb9)
    #1 _M_invoke<> /usr/local/include/c++/6.3.0/functional:1391 (dataRace+0x000000401be7)
    #2 operator() /usr/local/include/c++/6.3.0/functional:1380 (dataRace+0x000000401a8b)
    #3 _M_run /usr/local/include/c++/6.3.0/thread:196 (dataRace+0x000000401a06)
    #4 execute_native_thread_routine ../../../.././libstdc++-v3/src/c++11/thread.cc:83 (libstdc++.so.6+0x0000000c11be)

  Location is stack of main thread.

  Thread T2 (tid=13548, running) created by main thread at:
    #0 pthread_create ../../../../libsanitizer/tsan/tsan_interceptors.cc:876 (libtsan.so.0+0x00000002aaed)
    #1 __gthread_create /home/rainer/languages/C++/gcc-6.3.0/x86_64-pc-linux-gnu/libstdc++-v3/include/x86_64-pc-linux-gnu/
2 (libstdc++.so.6+0x0000000c14b4)
    #2 std::thread::_M_start_thread(std::unique_ptr<std::thread::_State, std::default_delete<std::thread::_State> >, void
ibstdc++-v3/src/c++11/thread.cc:163 (libstdc++.so.6+0x0000000c14b4)
    #3 main /home/rainer/Dropbox/articles/LinuxMagazin/Compilers/source/dataRace.cpp:8 (dataRace+0x000000400f97)

  Thread T1 (tid=13547, finished) created by main thread at:
    #0 pthread_create ../../../../libsanitizer/tsan/tsan_interceptors.cc:876 (libtsan.so.0+0x00000002aaed)
    #1 __gthread_create /home/rainer/languages/C++/gcc-6.3.0/x86_64-pc-linux-gnu/libstdc++-v3/include/x86_64-pc-linux-gnu/
2 (libstdc++.so.6+0x0000000c14b4)
    #2 std::thread::_M_start_thread(std::unique_ptr<std::thread::_State, std::default_delete<std::thread::_State> >, void
ibstdc++-v3/src/c++11/thread.cc:163 (libstdc++.so.6+0x0000000c14b4)
    #3 main /home/rainer/Dropbox/articles/LinuxMagazin/Compilers/source/dataRace.cpp:7 (dataRace+0x000000400f70)

SUMMARY: ThreadSanitizer: data race /home/rainer/Dropbox/articles/LinuxMagazin/Compilers/source/dataRace.cpp:8 in operator
==================
ThreadSanitizer: reported 1 warnings
rainer@suse:~>
```

Figure 2: The compiler detects a race condition thanks to Thread Sanitizer.

- Difficult to understand error messages suddenly make sense when another compiler compiles the code. Clang stands out above all with its understandable error messaging.
- Code is easier to verify. Undefined behavior in the source code has different effects. The program provides the wrong or supposedly correct result. Undefined behavior also means that source code cannot be compiled or the program crashes at run time. Testing your code online with different compilers and compiler versions will verify its correctness simply but effectively.

An example is shown in Listing 2. This code leads to undefined behavior. Since C++11, C++ has supported the delegation of constructors. This should not create a recursion, but it does in the example. The constructor for `char` calls the constructor for `double`, which in turn calls the constructor for `char`… and so on.

How do the Wandbox front end (Clang, GCC) and the Visual C++ Online Compiler (MSVC) deal with recursion? Whereas GCC exits with a segmentation fault at run time (Figure 3), the MSVC executable goes into an endless loop that is abruptly interrupted by the compiler front end. Only Clang is really trustworthy (Figure 4). The Clang compiler notices at build time that a recursion appears in the program.

Finally, the previously mentioned Explorer compiler takes over the job. Explorer generates the assembly instructions for an impressive number of compilers and compiler versions, supporting GCC, Clang, MSVC, and many others. These instructions are shown in the graphical front end side by side with the source code. To make it easier to map the two, Explorer color highlights the source code lines and the corresponding assembler

Listing 2: Recursive Calling of Constructors

```
struct C {
  C(char) : C(42.0) {}    // ill-formed due to recursion
  C(double) : C('a') {}   // ill-formed due to recursion
};

int main(){
  C('a');
  C(3.5);
}
```

Rescue Network

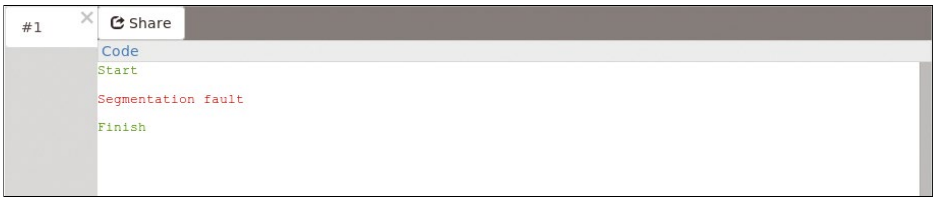
If your own compiler does not yet support the C++ standard you need, the web might provide a solution. The web is home to a wide range of online compilers for C++ that use one of the three major compilers in the background. Arne Mertz offers an excellent overview of online compilers [10]. This article explores a few of the most common options.

The Wandbox compiler [11] supports an impressive variety of GCC and Clang versions. The Online Visual C++ compiler [12] is used for Windows. More impressive is the Explorer compiler [13] by Matt Godbolt, which generates assembler code for a variety of compilers.

The online C++ compiler Coliru [14] has a unique selling point: You can integrate it into any website. For example, the online C++ reference at *cppreference.com* uses it [15].

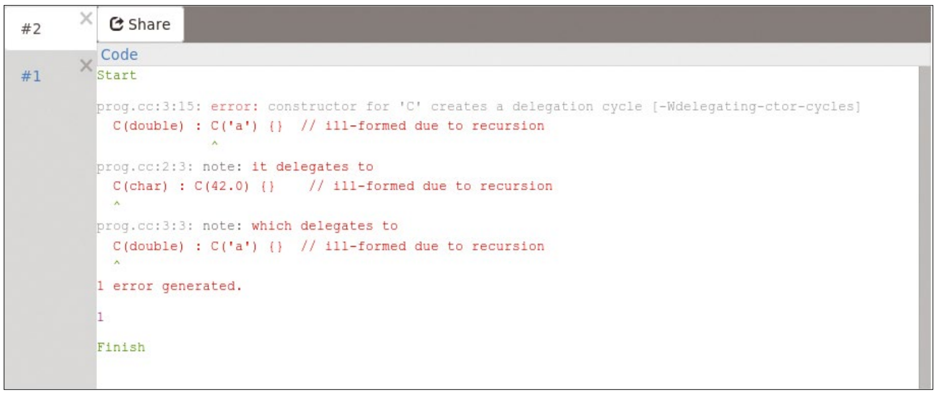
Online compilers have the following benefits:

- They let you test new features of the C++ standard. For example, you can test whether a compiler upgrade makes sense.



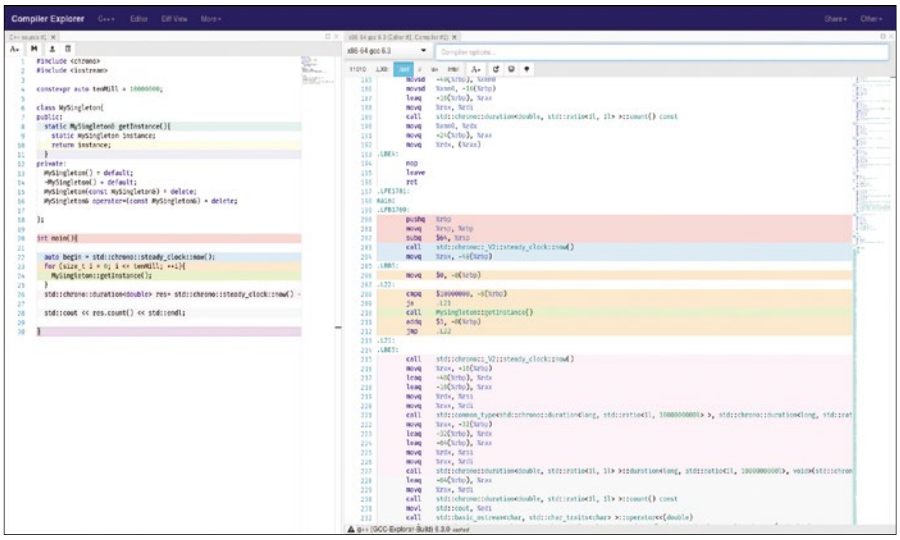
```
Start
Segmentation fault
Finish
```

Figure 3: GCC throws a segfault due to the recursive delegation of constructors.



```
Start
prog.cc:3:15: error: constructor for 'C' creates a delegation cycle [-Wdelegating-ctor-cycles]
  C(double) : C('a') {}  // ill-formed due to recursion
              ^
prog.cc:2:3: note: it delegates to
  C(char) : C(42.0) {}    // ill-formed due to recursion
  ^
prog.cc:3:3: note: which delegates to
  C(double) : C('a') {}  // ill-formed due to recursion
  ^
1 error generated.

1

Finish
```

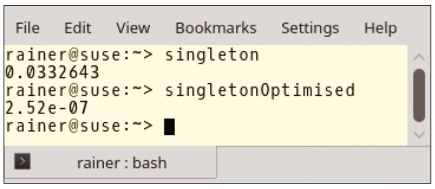
Figure 4: When compiling the program, Clang notices that there is a recursion.



```
#include <chrono>
#include <iostream>

constexpr auto tenMill = 10000000;

class MySingleton{
public:
  static MySingleton& getInstance(){
    static MySingleton instance;
    return instance;
  }
private:
  MySingleton() = default;
  ~MySingleton() = default;
  MySingleton(const MySingleton&) = delete;
  MySingleton& operator=(const MySingleton&) = delete;
};

int main(){

  auto begin = std::chrono::steady_clock::now();
  for (size_t i = 0; i <= tenMill; ++i){
    MySingleton::getInstance();
  }
  std::chrono::duration<double> res= std::chrono::steady_clock::now() -

  std::cout << res.count() << std::endl;

}
```

```
        movsd   -40(%rbp), %xmm0
        movsd   %xmm0, -16(%rbp)
        leaq    -16(%rbp), %rax
        movq    %rax, %rdi
        call    std::chrono::duration<double, std::ratio<1l, 1l> >::count() const
        movq    %xmm0, %rdx
        movq    -24(%rbp), %rax
        movq    %rdx, (%rax)
.LBE4:
        nop
        leave
        ret
.LFE1781:
main:
.LFB1780:
        pushq   %rbp
        movq    %rsp, %rbp
        subq    $64, %rsp
        call    std::chrono::_V2::steady_clock::now()
        movq    %rax, -48(%rbp)
.LBB7:
        movq    $0, -8(%rbp)
.L22:
        cmpq    $10000000, -8(%rbp)
        ja      .L21
        call    MySingleton::getInstance()
        addq    $1, -8(%rbp)
        jmp     .L22
.L21:
.LBE7:
        call    std::chrono::_V2::steady_clock::now()
        movq    %rax, -16(%rbp)
        leaq    -48(%rbp), %rdx
        leaq    -16(%rbp), %rax
        movq    %rdx, %rsi
        movq    %rax, %rdi
        call    std::common_type<std::chrono::duration<long, std::ratio<1l, 1000000000l> >, std::chrono::duration<long, std::rat
        movq    %rax, -32(%rbp)
        leaq    -32(%rbp), %rdx
        leaq    -64(%rbp), %rax
        movq    %rdx, %rsi
        movq    %rax, %rdi
        call    std::chrono::duration<double, std::ratio<1l, 1l> >::duration<long, std::ratio<1l, 1000000000l>, void>(std::chron
        leaq    -64(%rbp), %rax
        movq    %rax, %rdi
        call    std::chrono::duration<double, std::ratio<1l, 1l> >::count() const
        movl    std::cout, %edi
        call    std::basic_ostream<char, std::char_traits<char> >::operator<<(double)
```

Figure 5: **The Explorer compiler highlights source code and associated assembly instructions in color.**



```
rainer@suse:~> singleton
0.0332643
rainer@suse:~> singletonOptimised
2.52e-07
rainer@suse:~> █
```

Figure 6: **The execution time of the singleton calls in seconds.**

instructions. The effect is enhanced even more with the cursor placed in the source code so that Explorer highlights the corresponding assembler instructions (Figure 5).

Explorer's focus on assembly instructions has a number of benefits:

- By comparing the source code with the corresponding assembly instructions, you get a deeper insight into the compilation and optimization process.
- You can discover whether the compiler calculates an expression at compilation time or calls a function inline.
- It becomes clear how the various optimization levels affect the assembly instructions.

The last point is perhaps the biggest added value of the Explorer compiler. Thanks to Explorer, it is possible to understand the often-not-completely-intuitive behavior of the optimizer with a little practice. Consider the example shown in Listing 3.

Listing 3: **Singleton Pattern**

```
01 #include <chrono>
02 #include <iostream>
03
04 constexpr auto tenMill = 10000000;
05
06 class MySingleton{
07 public:
08   static MySingleton& getInstance(){
09     static MySingleton instance;
10     return instance;
11   }
12 private:
13   MySingleton() = default;
14   ~MySingleton() = default;
15   MySingleton(const MySingleton&) = delete;
16   MySingleton& operator=(const MySingleton&) = delete;
17
18 };
19
20 int main(){
21
22   auto begin = std::chrono::steady_clock:: now();
23   for (size_t i = 0; i <= tenMill; ++i){
24     MySingleton::getInstance();
25   }
26   std::chrono::duration<double> res= std::chrono::steady_
     clock::now() - begin;
27
28   std::cout << res.count() << std::endl;
29
30 }
```

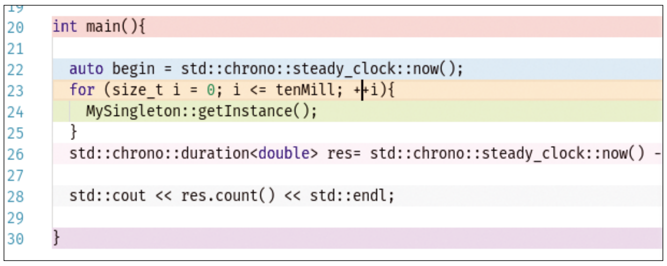
Listing 3 uses the singleton pattern (lines 6 to 18). The big question is how much time it takes to call the singleton 10 million times (line 24). The elapsed time is given in line 28. With GCC, the program is quickly compiled without and with maximum optimization (`-O3`). Figure 6 provides figures for performance.

Surprised? There is every reason to be, because the maximum optimized execution happens far too fast – unless the Optimizer was playing a trick. A look at the Explorer compiler sheds some light on the outcome. Figure 7 shows the source code in the non-optimized variant, and Figure 8 shows the corresponding assembly instructions. Nothing suspicious so far.

Figure 9 is much more suspicious. The singleton call `MySingleton::getInstance();` is not highlighted, which can only mean that there are no appropriate assembly instructions. See Figure 10 – not only are the instructions missing for the singleton call, but they are also missing for the `for-loop`. How is that possible? Since running the `for-loop` has no effect, the optimizer can remove it, which it does, but this leads to an absurd measurement of performance.

Know Your Compiler!

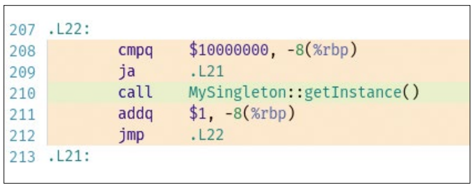
The online compilers offer unbeatable help when it comes to keeping up with the three-year cycle of modern C++ standards, which means that contemporary C++ developers don't have to ask whether they are using the best of all C++ compilers.



```
20 int main(){
21
22   auto begin = std::chrono::steady_clock::now();
23   for (size_t i = 0; i <= tenMill; ++i){
24     MySingleton::getInstance();
25   }
26   std::chrono::duration<double> res= std::chrono::steady_clock::now() -
27
28   std::cout << res.count() << std::endl;
29
30 }
```

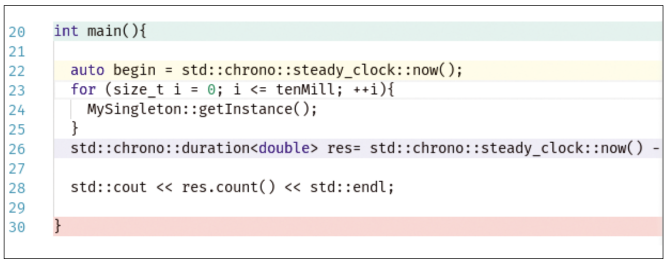
Figure 7: **Source code in non-optimized version.**

In the end, each compiler shows strengths and weaknesses. GCC and Clang implement the C++ core language faster than MSVC. However, MSVC often has the edge when it comes to implementing the C++ library. How can you harness the functional diversity of all three of the big three compilers? With the help of the excellent online compilers. ■■■



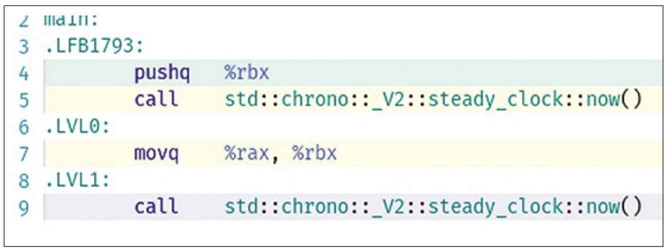
```
.L22:
        cmpq    $10000000, -8(%rbp)
        ja      .L21
        call    MySingleton::getInstance()
        addq    $1, -8(%rbp)
        jmp     .L22
.L21:
```

Figure 8: Assembler instructions for the non-optimized variant.



```
int main(){

  auto begin = std::chrono::steady_clock::now();
  for (size_t i = 0; i <= tenMill; ++i){
    MySingleton::getInstance();
  }
  std::chrono::duration<double> res= std::chrono::steady_clock::now() -

  std::cout << res.count() << std::endl;

}
```

Figure 9: Source code of the optimized version.



```
main:
.LFB1793:
        pushq   %rbx
        call    std::chrono::_V2::steady_clock::now()
.LVL0:
        movq    %rax, %rbx
.LVL1:
        call    std::chrono::_V2::steady_clock::now()
```

Figure 10: Optimized variant assembler instructions.

Info

[1] GCC: *https://gcc.gnu.org*

[2] Clang: *https://clang.llvm.org*

[3] MSVC: *https://docs.microsoft.com/en-us/cpp/*

[4] GNU ARM Toolchain: *https://developer.arm.com/open-source/gnu-toolchain/gnu-rm*

[5] An overview of the C++17 standard, its features, and compiler support: *http://en.cppreference.com/w/cpp/compiler_support*

[6] A parallel STL implements HPX: *http://stellar.cct.lsu.edu/projects/hpx/*

[7] The Sanitizer checks the use of addresses, memory, and threads: *https://github.com/google/sanitizers/wiki*

[8] Windows port of sanitizer: *https://github.com/google/sanitizers/wiki/AddressSanitizerWindowsPort*

[9] Thread Sanitizer: *https://github.com/google/sanitizers/wiki/ThreadSanitizerCppManual*

[10] Arne Mertz's collection of C++ online compilers: *https://arnemertz.github.io/online-compilers/*

[11] Wandbox online compiler: *https://wandbox.org*

[12] Online Visual C++ compiler: *http://webcompiler.cloudapp.net*

[13] Explorer compiler: *https://godbolt.org*

[14] Coliru: *http://coliru.stacked-crooked.com*

[15] Online C++ reference: *http://en.cppreference.com/w/*

Prepare to Jump

LibreOffice 5.4 is the last major release before the big jump to version 6. In addition to a number of updates, you can now sign Writer documents with OpenPGP. *By Heike Jurzik*

Every six months, The Document Foundation publishes a new release of LibreOffice [1]. The versions in the `fresh` branch offer experimental features aimed at users who want to test brand new functions, whereas the `still` versions are for those who value reliability [2].

The new `fresh` version 5.4, the last in the 5.x series before the big jump to version 6.0, is scheduled for early 2018. It comes with new toolbars, improved palettes and filters, and OpenPGP support for signing Writer documents. A test team looked at the new LibreOffice 5.4.0.3 under Debian 9.0 (Stretch, 64-bit).

LibreOffice developers have continued to modernize the appearance of the Office programs. In particular, the cleaned-up default color palette immediately stands out (Figure 1). Instead of colorfully mixed hues, the palette now follows the RYB (red, yellow, blue) color model reduced to 120 tones. A new palette for chart colors (not yet compiled and available as `chart-palettes`) is also included.

The new toolbar for styles stands out. If you prefer to see it at the top rather than in the right sidebar, you can place it with *View* | *Toolbars* | *Formatting*.

The previous 5.3 release was the first to offer a "ribbon," called a notebook bar in LibreOffice, that gathers program functions into groups rather than menus. The feature is still marked as experimental in version 5.4, but you can enable it in *Tools* | *Options* when you check *Enable experimental features (may be unstable)* in the dialog below *LibreOffice* | *Advanced*.

After restarting the program, you can access the new look with *View* | *Toolbar Layout* | *Notebookbar*; then, in *View* | *Notebookbar*, you can decide whether menu entries appear as contextual single (context-dependent content in a single centered toolbar), tabbed (icons grouped by context), or contextual group (fixed File, Clipboard, and Format groups and an Insert group dependent on the object of focus). In version 5.4, the notebook bar now support themes. During tests, the feature did not prove itself to be particularly stable, and LibreOffice often crashed when changing the view.

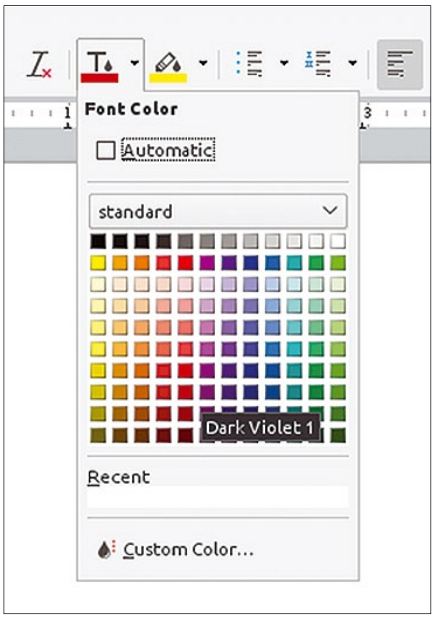


Figure 1: The new standard color palette looks tidier.

Exchange Formats

XML-based documents in ODF and OOXML formats (ODT and DOCX in Writer) are now significantly smaller than the files that proprietary Office suites produce for identical documents, mainly because, according to the developers [3], LibreOffice does not write redundant XML tags to files. With code optimization, Writer documents are up to 90 percent leaner compared with Word documents.

As with any new edition, the programmers improved exchange between third-party formats. LibreOffice 5.4 includes improved support for Windows Enhanced Metafile (EMF) vector graphics created using the ChemDraw software and a revised PDF import feature: The office suite now relies on the `pdfium` library to render inserted images. Innovations in PDF export let Writer and Impress take over embedded videos (Figure 2).

The most interesting feature in the new version is that Linux users can now sign their Writer documents with their

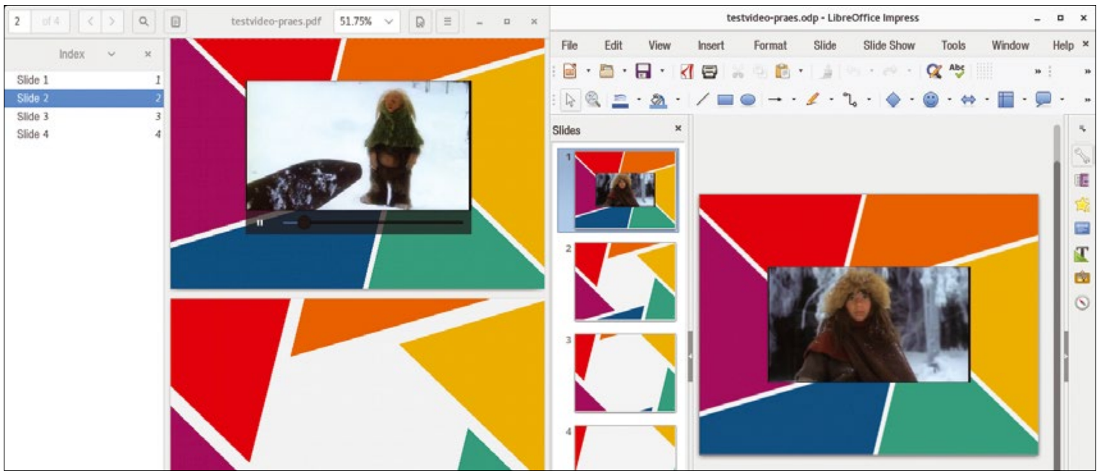


Figure 2: New features for exporting PDFs: The Writer word processor and the Impress presentation tool also export embedded videos.

OpenPGP keys, which guarantees the authenticity of their files regardless of storage location or transmission path. The developers rely on the GnuPG Made Easy API (GPGME) [4].

To sign an ODF file, call *File* | *Digital Signatures* | *Digital Signatures*, click on *Sign Document*, and choose the OpenPGP key in the dialog; Writer sends the public key along with the document.

No Spoofs

If a document signed in this way reaches someone that uses an older version of LibreOffice or an operating system other than Linux, a dialog appears with a note that the signed content does not match the current document signature. After clicking *OK*, Writer opens the file without complaint, but without a signature.

Linux users who also use the new LibreOffice will see a color bar between toolbars and the document, as well as a note that lets you draw conclusions about the signature. As with X.509 certificates, there are three options: valid, invalid, or not verified, in which case, you should not yet trust the key.

After clicking on *Show signatures*, a dialog box appears that not only displays the signatures (Figure 3), but also launches the Certificate Manager. LibreOffice currently supports Kleopatra, Kgpg, and GPA. Cooperation with the Gnome keyring Seahorse did not work in version 5.4.0, but the bug should be corrected in 5.4.2. Alternatively, use `gpg` at the command line to trust a key.

The office suite includes a number of other changes: Writer now adds user-defined watermarks and extends right-click context menus, depending on where the cursor is. In Calc, the developers added extended dialogs and an improved comment function. The spreadsheet application now also shows negative annual figures; the previous version was able to compute, but not display, them.

Impress comes with a new keyboard shortcut for inserting sheets quickly; Math gets additional commands for the context menu; and in Base, the developers improved a number of functions and rearranged the menu structures. The online edition of the office suite is now available with a responsive design for mobile devices and includes a read-only mode for documents.

In the Starting Blocks

The new version was largely stable. The test team only experienced crashes after enabling the experimental notebook bar features. When switching between viewing modes, LibreOffice 5.4 reproducibly bows out. If you can do without the notebook bar concept, you will receive a strong Office suite, with convincing new features and functions.

Teaming up with GnuPG is especially successful. For upcoming version 6.0, the developers are working on integrating additional platforms, especially Windows. It should also be possible to encrypt documents with GnuPG. ■■■

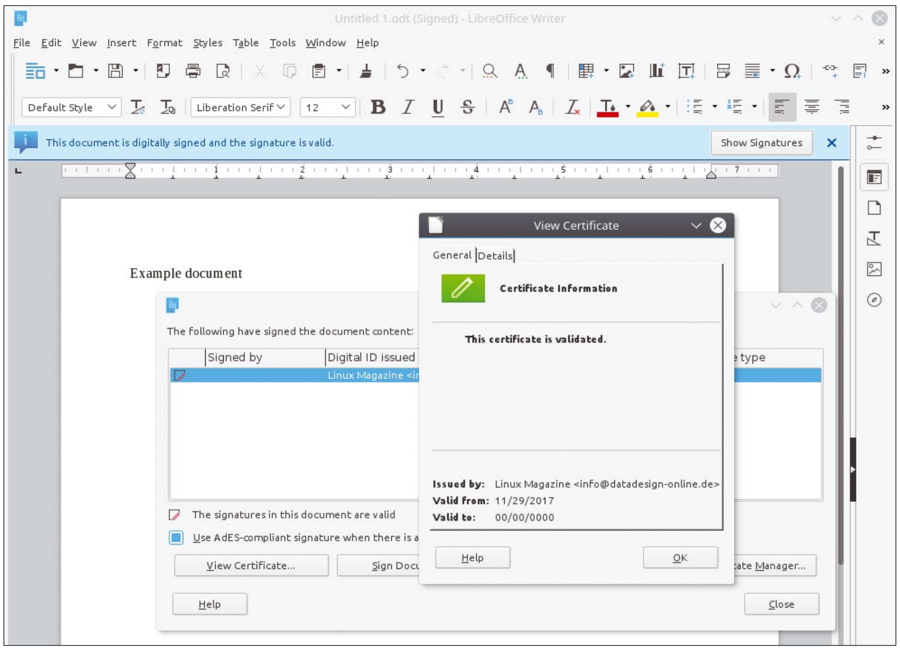
Info

[1] LibreOffice: *http://www.libreoffice.org*

[2] LibreOffice, `fresh` and `still`: *http://www.libreoffice.org/download/release-notes*

[3] Background on file sizes: *https://nextcloud.documentfoundation.org/s/5Oe8guDN0XSS7h8*

[4] GPGME: *https://wiki.gnupg.org/APIs*



This document is digitally signed and the signature is valid.

Example document

The following have signed the document content:

Linux Magazine <in

This certificate is validated.

Issued by: Linux Magazine <info@datadesign-online.de>
Valid from: 11/29/2017
Valid to: 00/00/0000

The signatures in this document are valid

Use AdES-compliant signature when there is a

Figure 3: Linux users can now sign Writer documents with their own OpenPGP key. LibreOffice displays the status in a color bar; the light blue background in the image indicates a valid signature.

What's new in Ubuntu 17.10

Back to the Future

Ubuntu is back to the first letter of the alphabet – and back to the Gnome desktop. But you'll also find lots of new stuff in the latest release. *By Swapnil Bhartiya*

When Mark Shuttleworth announced Ubuntu in 2004, he was extremely passionate about it. Shuttleworth saw the possibility of a third operating system in the consumer space, and many things worked in his favor. Ubuntu managed to create one of the friendliest and most vibrant open source communities.

The success of Ubuntu on the desktop made Shuttleworth even more ambitious about the mobile space, and he thought he could crack the duopoly of iOS and Android with Ubuntu Mobile. He saw a growing trend of smart TVs, and he even felt Ubuntu could offer a platform to TV vendors.

In 2011, Canonical decided Gnome 3 didn't fit into the grand scheme of Ubuntu. The Ubuntu project dropped Gnome Shell (it still used the desktop stack, just not the shell) and created its own shell called Unity that was aligned with Canonical's plans for mobile, tablets, and TVs. At the 2012 Mobile World Congress, Canonical had one of the biggest stalls, where they showcased their technologies.

None of those plans worked out. Canonical seemed to forget that people don't use platforms; they use applications. In 2017, Shuttleworth realized it was time to give up. He stopped investing in consumer-centric products. Projects were shut down; people were let go. The Canonical we knew had changed, for good.

In an interview, Shuttleworth told us that, going forward, Canonical was going to focus on three key areas: servers, cloud, and Internet of Things (IoT); consumer was not their playground. However, desktop was still a critical piece for Canonical. According to the latest Stack overflow survey, the Linux desktop remains the most loved platform by developers (32.9%). It was the second most commonly used platform after Windows (41%).

If you go to any developer conference, Ubuntu is often the only Linux distribution you will see on stage for presentations. Even Microsoft uses Ubuntu at its own events. Linux now accounts for more than 50 percent on Azure cloud, and many of those Linux instance are running Ubuntu.

Ubuntu desktop shares the same codebase with Ubuntu server and other products. Canonical is going to invest in that codebase either way, which means Ubuntu desktop can have a life of its own even if the company now has other priorities.

When Canonical pulled the plug on Unity, it decided to go back to Gnome as the default desktop environment and embrace the Gnome 3 Shell. The Gnome project thus regains the users of one of the largest desktop Linux projects. Canonical gets a fully tested and polished desktop for its users, and users get the super-stable base of Ubuntu. Developers can target integration with one of the major desktop environments without worrying about Unity breaking things.

The Desktop

Ubuntu 17.10 shows that Canonical has not abandoned the Ubuntu desktop. 17.10 is not a Frankenstein's monster where pieces are loosely glued together. Canonical has indeed invested resources in integrating the Gnome 3 Shell with Ubuntu to offer a great out-of-the-box experience. The features you see in this release will be polished and improved for the Ubuntu 18.04 LTS release in April 2018.

Ubuntu engineers made a lot of tweaks to Gnome (on a cosmetic level, without heavily patching or forking anything), so a user upgrading from Ubuntu 17.04 or 16.04 is in for a shock.

Lead Image © sellingpix, 123RF.com

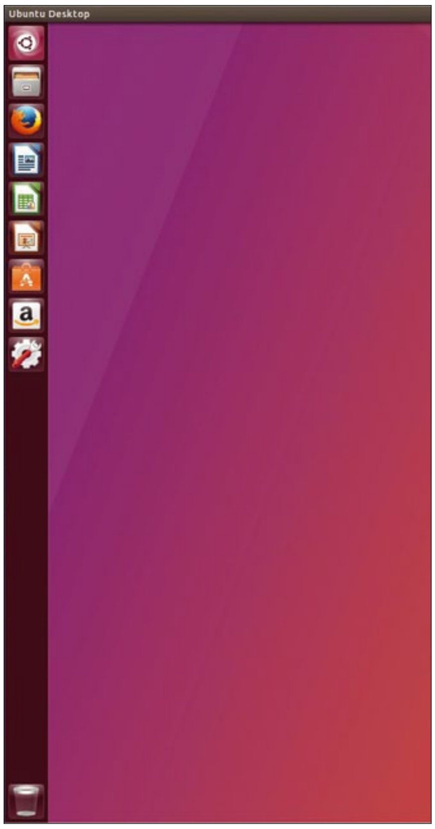


Figure 1: Ubuntu 17.04 placed the Start button in the top left corner of the screen.

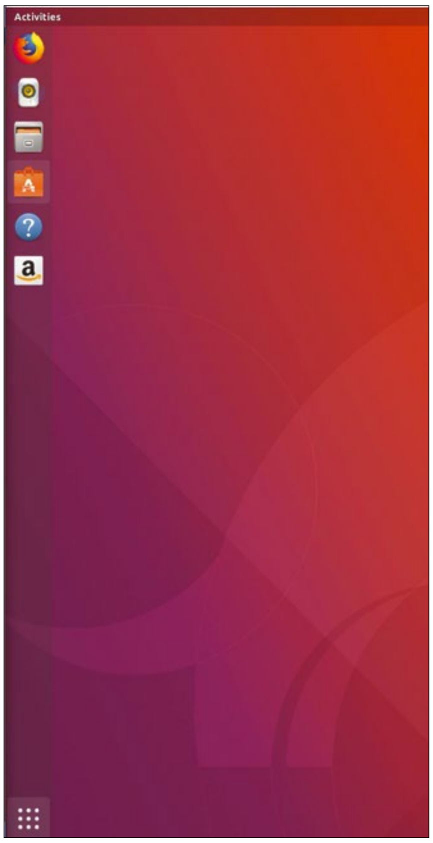


Figure 2: In Ubuntu 17.10, the Start button moves to the bottom, farther from the apps.

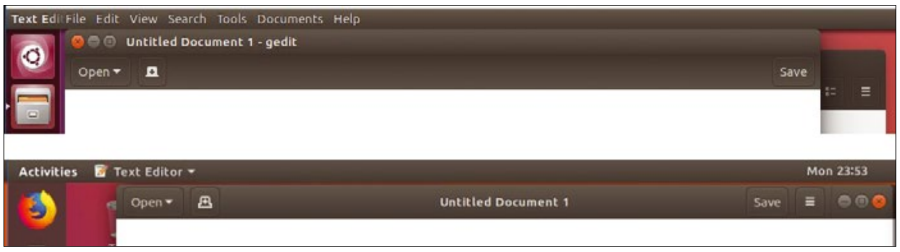

Text Edi File Edit View Search Tools Documents Help
Untitled Document 1 - gedit
Open
Save
Activities
Text Editor
Mon 23:53
Open
Untitled Document 1
Save

Figure 3: Migration to Gnome brings changes to the Ubuntu menubar.

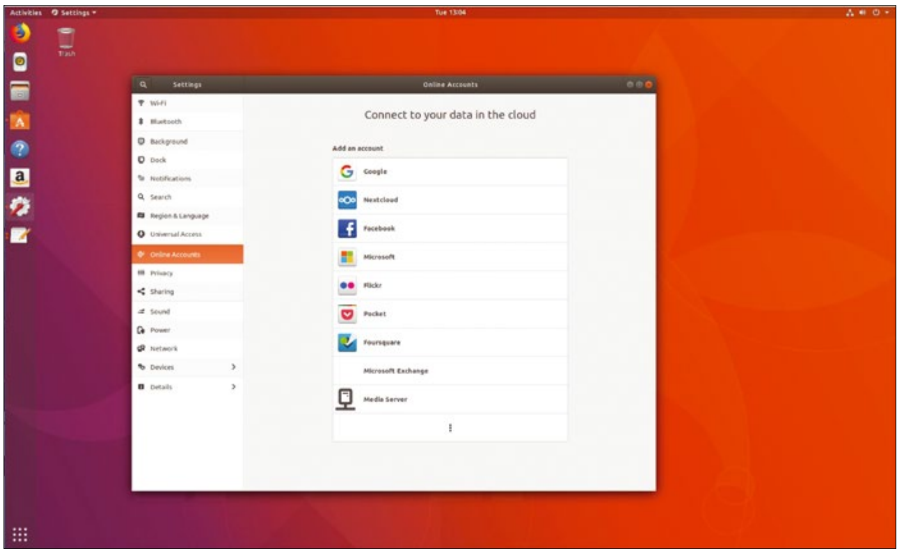


Figure 4: Ubuntu users and enjoy "soft" integration with Google Drive.

One of the first things you'll notice when you boot Ubuntu 17.10 is the *Start* dash launcher has moved from the top in 17.04 (Figure 1) to the bottom in 17.10 (Figure 2). Personally, I do not like the new location. A better design would be to keep all buttons close to each other so users can easily access the launcher and apps. In 17.04, you can open the launcher to open new apps or places and pin most used apps next to it; in 17.10, the launcher has moved to the bottom, far from the apps. It's quite a journey on my 27-inch 4K monitor where I have to go all the way to the bottom and then come back up.

The second minor but significant change is moving windows buttons from the Mac OS-like left side of the windows to the right side.

Unity's menubar behaves differently from Gnome's menubar (Figure 3). Muscle memory testing, auto-hiding menu items are gone and have been replaced with Gnome's over-simplified menus (I still prefer full menu items).

The top bar retains some of the same components – network, sound, power, and a drop-down menu, in addition to date and calendar in the center. Notification behavior has also changed. The older pop-up balloon, which could not be clicked, has been replaced with a fully functional notification system of Gnome.

To ensure better integration with Unity, Ubuntu had either forked or retained older packages from Gnome, like the Nautilus file manager, which has been replaced with the brand new Files file manager. Unfortunately, with the forked version of Nautilus, I couldn't batch rename files in Ubuntu because the older version of Nautilus didn't support it. Now, I can batch-rename all 100 photos that I took at OpenStack Summit.

Gnome also brings pseudo-integration with Google Drive to Ubuntu (Figure 4); you can easily configure the *Online Accounts* feature that remotely mounts Google drive to Ubuntu.

Ubuntu forked Gnome Control Center; now it's back to the brand-new Gnome Center (Figure 5) that has a radically different user interface (UI), which cuts down the number of clicks. It's still growing on me. That's going to be the case with most Ubuntu users. Some people will love these changes, and some won't. The good news is that instead of Unity, which was forcing a

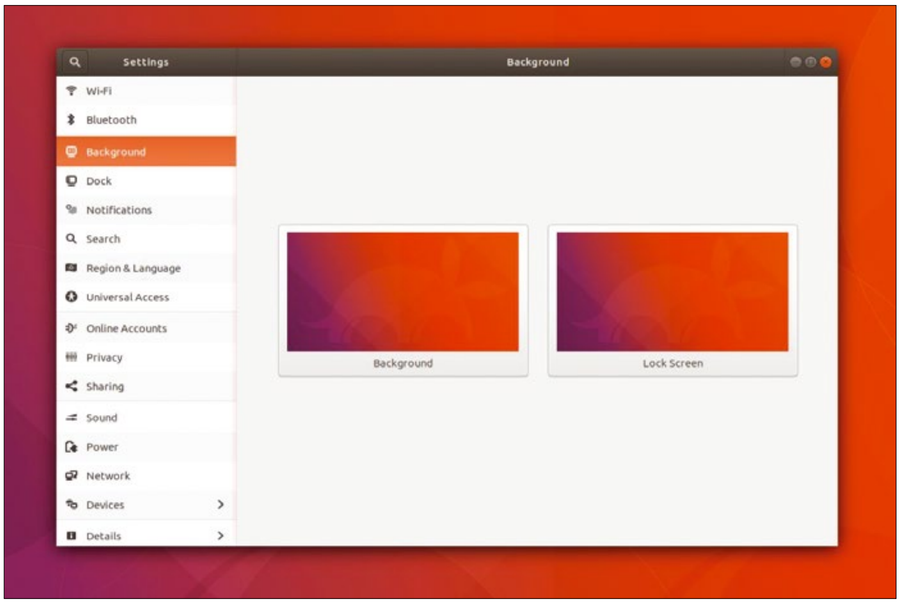


Figure 5: Ubuntu 17.10 gets the redesigned Gnome Center.

mobile UI onto the desktop, Gnome is very much a desktop interface.

Everything is great about Gnome, except for one tiny difference. On my systems, Ubuntu was chewing more RAM with Gnome; almost 10-15 percent more. This additional resource usage might be a concern if you plan to run Ubuntu 17.10 on older hardware.

The Ubuntu Experience

I tested Ubuntu 17.10 on three different machines: Dell Precision 5720 AIO, Dell XPS Developer Edition, and a custom PC running Intel i7-4700K, 32GB memory, and a GTX 1070 Ti card. Ubuntu installed without any hitch on these machines.

The installation process was smooth. As expected, everything worked out of the box on all three machines. Ubuntu even detected and added my Brother HL-L2340 G Series printer that was connected to the wireless network. While Ubuntu was installing, I received a pop-up notification about "a new printer detected." When I checked Gnome Settings, the printer was automagically added, and I was able to start printing. Even on Windows 10 and Mac OS, I have to manually detect and add the printer. So,we can safely say that desktop Linux, especially consumer distros like Ubuntu, have come a long way. It's not automatic. Ubuntu 17.10 has driverless printing with IPP Everywhere, Apple AirPrint, Mopria, and Wi-Fi Direct.

If you travel and have tried to join WiFi networks at airports and hotels, you might have struggled logging in. Ubuntu 17.10 has made it easier to get connected by adding support for captive portals. Those who still listen to music on their laptops, you can now easily switch between built-in audio devices and Bluetooth.

Everything looked good, except for one visible problem: the touch screen. Touch input behaved as a mouse; when I scrolled my finger on a page, instead of scrolling the page up and down, as expected on a touchscreen, it selected the text. Dell offers its own variant of Ubuntu and RHEL that comes with additional packages to support a multi-touch screen. That's one problem I face with every desktop Linux distro out there; there is no out-of-the-box multi-touch support baked in.

The Linux desktop community is working on solving the app distribution problem and fragmentation with solutions like AppImage, Flatpack, and Snaps. It's becoming clear that Flatpack will get wider support, as compared to Snap, which is seen as a Canonical solution. Will there be a standoff between Snap and Flatpack like the previous row surrounding systemd vs. Upstart? Possibly, especially considering that even Ubuntu derivatives like Linux Mint are backing Flatpack.

The big difference between Snap and Flatpack is that Snap is specifically targeting Cloud and the IoT space and therefore is trying to solve a different problem from Flatpack. Canonical might continue to develop Snap for those use cases. This release comes with improved support for Snap. The most interesting Snap feature in this release is the `catkin` Snapcraft plugin that enables Robot Operating System (ROS) snaps for secure, easily updated robots and drones. There are many new mediated secure interfaces available to snap developers, including the ability to use Amazon Greengrass and Password Manager.

Under the Hood

Ubuntu 17.10 comes with Linux kernel 4.13, which comes with support for new hardware and peripherals from ARM, IBM, Dell, Intel, and others.

Wayland is the default display server; Mir is nowhere to be seen in the Ubuntu orbit. But Wayland is new, and performance issues with some of the latest GPUs might give you a hard time. I resorted to not using my $550 GTX 1070 Ti drivers and settled down with integrated graphics. In addition to hardware support, Wayland may also give you trouble with things like screen recording. Ubuntu is offering X.Org as a backup, in case something fails on you. That said, the good news is you won't even notice that you are using Wayland; except for some corner cases, everything else worked fine.

Developers, Developers, Developers

When you look under the hood, most of the focus has been on developer tools and the developer experience. Like every other Linux vendor, Canonical tweaks the kernel. The 17.10 kernel adds support for Opal disk drives and numerous improvements to disk I/O. Namespaced file capabilities and Linux Security Module stacking reinforce Ubuntu's leadership in container capabilities for cloud and bare-metal Kubernetes, Docker, and LXD operations.

Ubuntu 17.10 has settled down with Netplan as the default network configuration manager, eliminating the fragmentation of NetworkManager, ifupdown, and half a dozen network interfaces. Canonical said that Netplan is backwards compatible, enabling interfaces to continue to be managed by tools like NetworkManager, while providing a simple overview of the entire system in a single place. The server and cloud editions of Ubuntu will assign network devices to `systemd-networkd` through Netplan.

Conclusion

Ubuntu is back to being a Gnome distribution. Ubuntu 17.10 offers the best of both worlds – out-of-the-box support for hardware, access to one of the biggest software ecosystems, ease of use, and all the innovation that the Gnome community brings to the table.

If you are running 17.04, it makes perfect sense to jump to 17.10 and start enjoying the new experience; your version won't be supported for longer either way. But if you are on an LTS release, I would suggest waiting for the 18.04 LTS edition, which will iron out most of the issues to prepare for LTS users.

By going back to Gnome, Canonical has freed itself from the unnecessary work it was doing on Unity. Those who don't like Gnome can use KDE Plasma or the Mate desktop environment. What about Unity? There are some independent developers who are trying to keep it alive.

Does it make sense for Canonical to create a Fedora-RHEL-like model where Ubuntu can become a community-driven distribution, whereas server and cloud can be a company-managed product? Shuttleworth has rejected that idea in the past, stating that the unique value that Canonical brings to the desktop Linux world is the same codebase and same product running in both enterprise and consumer space. There is no secret sauce. Everyone uses the same Ubuntu. ■■■

Author

Swapnil Bhartiya is a writer and journalist covering Linux and open source for more than 10 years. He is also a science fiction writer whose stories have been broadcast on Indian radio and published in leading Indian magazines. He founded an open source web magazine while living in Europe. Swapnil currently resides in Washington, DC.

Graphical tools for firewall configuration

Restricted Zone

Setting up a comprehensive firewall with netfilter and iptables is complicated. Graphic user interfaces seek to take the worries out of this demanding task. *By Erik Bärwaldt*

Firewalls under Linux are usually based on the kernel's netfilter system [1], which was introduced in 2001. Nftables [2] is about to replace this system, but until then, iptables [3] remains the configuration helper for the complicated netfilter system and is regarded as the default tool for Linux.

However, configuring iptables is not very intuitive. If you don't regularly use this process, you tend to forget quickly the necessary command-line parameters. Iptables does not make it easy for less experienced administrators to configure the firewall, so several distributions have their own tools. Because of this lack of intuitiveness, running the packet filter at the command line can quickly cause damage by user error.

For this reason, many firewalls now have graphical user interfaces (GUIs), which makes this somewhat cumbersome task easier. In this article, I review four such GUIs: firewalld [4], fwbuilder [5], Gufw [6], and Shorewall [7]. I also looked at the PeerGuardian [8] IP blocker, which is not a conventional firewall (see the "PeerGuardian" box). Not included in this review are configuration environments that are outdated (see the "Not in the Running" box).

Not in the Running

In addition to the GUIs for firewall modules discussed in this article, other configuration environments, such as FireStarter [10], Turtle Firewall [11], or FireFlier [12], can occasionally be found on the Internet. The ncurses program, Vuurmuur [13], has also gained a certain popularity as a Linux firewall management application. What all of these packages have in common is that they have not been maintained for about 10 years, and therefore they do not support – or at least do not fully support – new standards, such as IPv6.

In this article, I have also left out other active professional systems, such as IPFire [14], Untangle NG Firewall [15], and Alpine Linux [16], because they are specialist Linux distributions not based on a standard Linux system.

Criteria

Two of the most important things that professional firewalls need to support are the ability to handle IPv4 and IPv6 and the ability to adapt dynamically. In contrast to a static firewall, not every modification should stop and restart the firewall while interrupting the Internet connection, which is the only way to implement appropriate rules for applications that require specific ports during operation.

Another important evaluation criterion for firewalls is logging. For example, log analyses of packet transfers help the admin set up an Intrusion Detection System (IDS) or Intrusion Prevention System (IPS). Application filters and blacklists also boost security – as long as the admin maintains and updates them regularly.

firewalld

Firewalld [4] has been the default firewall on Red Hat Enterprise Linux (RHEL) since version 7, replacing iptables in this distribution. Although firewalld works with the netfilter system, the software is incompatible with the iptables control model. The firewall, which runs as a daemon, is also found in Fedora and the CentOS RHEL derivative, as well as in the repositories of most common Linux derivatives.

Firewalld supports IPv4 and IPv6; in particular, its zone model stands out. It lets you configure the firewall for different zones, each containing a specific ruleset. The rules are based on the desired or required security level, which is especially advantageous for mobile devices: Depending on the working environment, the user can select the relevant zone, which guarantees a specific level of safety.

PeerGuardian

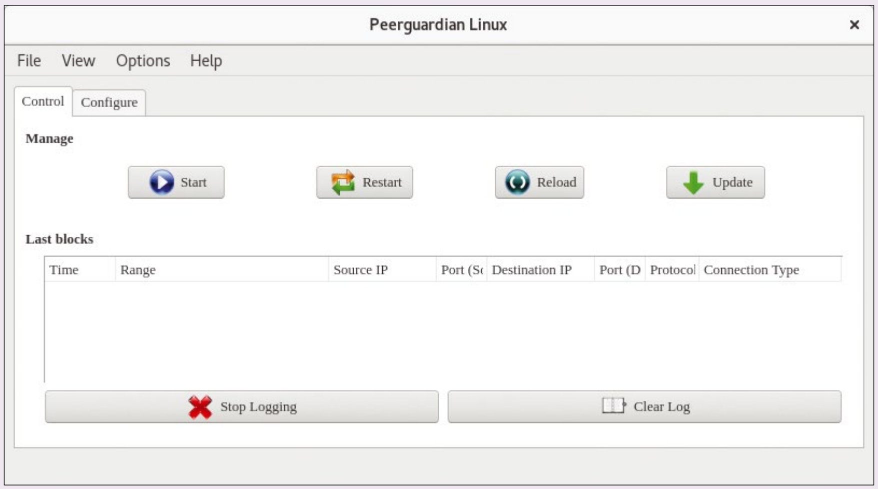


Figure 1: PeerGuardian blocks IP addresses or IP address ranges.

PeerGuardian (Figure 1) [8] is not a conventional firewall, but an application that blocks individual IP addresses or entire address blocks. Originally, the software was designed to prevent peer-to-peer connections under protocols such as BitTorrent or FastTrack from being spied on, but now it also blocks IP addresses that link to websites with criminal content or to spam and phishing sites. It uses the netfilter and iptables rules available on the host. Addresses and address ranges are blocked by the software with predefined blocklists that contain known IP addresses with malware. You can add to these lists.

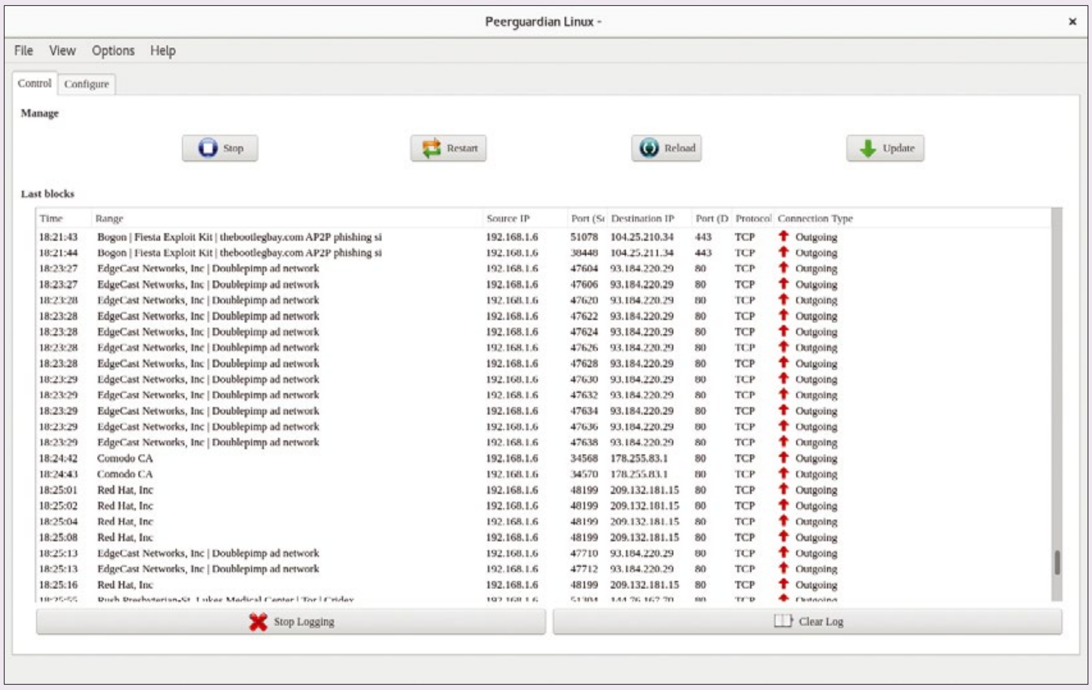
PeerGuardian's source code is available for DIY compilation, as well as from the repositories of some major Linux distributions. The GUI greatly simplifies the handling of blocklists. PeerGuardian lets you block unwanted IP addresses quickly without the need for complex proxy server configuration in intranets.

The program, which is distributed under the GNU GPL, initially comes up with an empty list area in the active *Control* tab of the dialog window showing the session log. The buttons above manage the software. The ready-made blocklists are grouped in the *Configure* tab. Here, in the Whitelist area, you can enter addresses to be released. At the top is an option to start the software at system boot and to update the blocklist automatically.

You can activate the blocklist update intervals as required by checking the address range to be blocked. You can add more websites or areas to the list by pressing the green plus symbol below the blocklist. On first use, you will want to update the lists by pressing the *Update* button in the *Control* tab; then, track the update in a small log window, which you call with *View* | *View pglcmd's log*. Once the updates have been installed, the firewall is enabled by clicking on the *Start* button in the *Control* tab. The log window now gradually fills up with blocked IP addresses, the associated ports, and information about the type of connection (Figure 2).

PeerGuardian offers not only your predefined lists and blocklists, but also externally predefined address collections [9]. They are available on the Internet, divided into categories, some of which can also be purchased for a fee as part of a subscription. You can add the lists to PeerGuardian by copying and pasting the list URL into the add-on dialog; it then regularly updates the blocklists moving forward. If IP addresses that you do not want to block are shown in the history log, you can change it by right-clicking on the entry: Using the context menu, you can temporarily or permanently release both the IP address and the associated port.

If there is too little information available about the blocked address, you can launch a *whois* query from the context menu. The software then displays information in a window, making it easier to decide whether to extend or remove the block.



Time	Range	Source IP	Port (Sc	Destination IP	Port (D	Protocol	Connection Type
18:21:43	Bogon | Fiesta Exploit Kit | thebootlegbay.com AP2P phishing si	192.168.1.6	51078	104.25.210.34	443	TCP	Outgoing
18:21:44	Bogon | Fiesta Exploit Kit | thebootlegbay.com AP2P phishing si	192.168.1.6	38448	104.25.211.34	443	TCP	Outgoing
18:23:27	EdgeCast Networks, Inc | Doublepimp ad network	192.168.1.6	47604	93.184.220.29	80	TCP	Outgoing
18:23:27	EdgeCast Networks, Inc | Doublepimp ad network	192.168.1.6	47606	93.184.220.29	80	TCP	Outgoing
18:23:28	EdgeCast Networks, Inc | Doublepimp ad network	192.168.1.6	47620	93.184.220.29	80	TCP	Outgoing
18:23:28	EdgeCast Networks, Inc | Doublepimp ad network	192.168.1.6	47622	93.184.220.29	80	TCP	Outgoing
18:23:28	EdgeCast Networks, Inc | Doublepimp ad network	192.168.1.6	47624	93.184.220.29	80	TCP	Outgoing
18:23:28	EdgeCast Networks, Inc | Doublepimp ad network	192.168.1.6	47626	93.184.220.29	80	TCP	Outgoing
18:23:28	EdgeCast Networks, Inc | Doublepimp ad network	192.168.1.6	47628	93.184.220.29	80	TCP	Outgoing
18:23:29	EdgeCast Networks, Inc | Doublepimp ad network	192.168.1.6	47630	93.184.220.29	80	TCP	Outgoing
18:23:29	EdgeCast Networks, Inc | Doublepimp ad network	192.168.1.6	47632	93.184.220.29	80	TCP	Outgoing
18:23:29	EdgeCast Networks, Inc | Doublepimp ad network	192.168.1.6	47634	93.184.220.29	80	TCP	Outgoing
18:23:29	EdgeCast Networks, Inc | Doublepimp ad network	192.168.1.6	47636	93.184.220.29	80	TCP	Outgoing
18:23:29	EdgeCast Networks, Inc | Doublepimp ad network	192.168.1.6	47638	93.184.220.29	80	TCP	Outgoing
18:24:42	Comodo CA	192.168.1.6	34568	178.255.83.1	80	TCP	Outgoing
18:24:43	Comodo CA	192.168.1.6	34570	178.255.83.1	80	TCP	Outgoing
18:25:01	Red Hat, Inc	192.168.1.6	48199	209.132.181.15	80	TCP	Outgoing
18:25:02	Red Hat, Inc	192.168.1.6	48199	209.132.181.15	80	TCP	Outgoing
18:25:04	Red Hat, Inc	192.168.1.6	48199	209.132.181.15	80	TCP	Outgoing
18:25:08	Red Hat, Inc	192.168.1.6	48199	209.132.181.15	80	TCP	Outgoing
18:25:13	EdgeCast Networks, Inc | Doublepimp ad network	192.168.1.6	47710	93.184.220.29	80	TCP	Outgoing
18:25:13	EdgeCast Networks, Inc | Doublepimp ad network	192.168.1.6	47712	93.184.220.29	80	TCP	Outgoing
18:25:16	Red Hat, Inc	192.168.1.6	48199	209.132.181.15	80	TCP	Outgoing

Figure 2: PeerGuardian displays blocked IP addresses in a window.

Firewalld also demonstrates its strengths in large IT infrastructures with DMZ integration. In this way, the administrator can configure the firewall to suit the interface. The wireless settings then differ from those on the wired LAN. You can set up the server on the intranet or in a DMZ with different zones to suit your needs.

Firewalld comes with several zones that offer different preset security levels: The palette ranges from the *trusted* zone, which forwards all incoming data packets, through other predefined rulesets, to the *drop* zone, which discards all incoming packets if they do not relate to outgoing packets. The daemon supports its own syntax with which the zones can be managed at the command line.

For less experienced admins who want to set up firewalld as quickly as possible, the software offers a graphical tool primarily designed for Gnome. Firewalld automatically lands on the disk in Fedora, CentOS, and RHEL, but you have to install it on Gnome through its Software utility – the package is called *Firewall*. Alternatively, you can integrate the `firewall-config` tool with:

```
yum install firewall-config
```

After installing, call the GUI by pressing the *Start* button in the graphical installer. Alternatively, enter `firewall-config` in the terminal. Assuming you have logged in as a system administrator, you will be taken to a very clear-cut interface (Figure 3).

A list on the left side of the program window labeled Active Bindings (not shown in this figure) shows all the active connections that physically exist in the system along with the matching zones. To the right of this list are three tabs (*Zones*, *Services*, *IPSets*) supplemented by a smaller window in the lower-right-hand corner that allows changes to other group-specific options. The Configuration drop-down above the tabs determines whether the options are changed permanently (*Permanent*) or temporarily (*Runtime*).

Zones

In the *Zones* tab, you can define the respective packet transfer rules for all existing firewall zones. The *Services*, *Ports*, and *Protocols* tabs contain the most important groups. All tabs show the same options in the basic settings.

The *Services* tab makes services installed on the computer accessible for external access. Depending on the application scenario, it grants access from an intranet or the Internet. You will come across a very extensive selection of available services, which you can enable by checking the boxes. If you log in as a system administrator, firewalld instantly enables the appropriate settings without rebooting.

In the *Ports* tab, you can manually release individual ports or whole port areas for external access to the system. If you want certain protocols to pass through the firewall, you can pick them from the *Protocols* tab. Specifically for IPv4, the *Masquerading* and *Port Forwarding* tabs are intended for setting up the relevant computer system as a gateway for an intranet. However, for this purpose, you have to equip the computer on which the firewall is running with two interfaces. Port forwarding is used to forward ports to the local system or to a remote computer.

For less common services or individual configurations, you can configure firewalld in the *Services* tab. You can use *Ports* to assign port addresses for predefined or manually added services that deviate from the defaults. You can also define source ports and target addresses on request, although these are only available in the *Permanent* configuration setting. Target addresses can be enabled for IPv4 and IPv6.

The last *IPSets* tab lets you define IP address ranges or individual IP addresses for which firewalls grant or block external access. For this purpose, you create the corresponding blacklists and whitelists; the software also takes port numbers and MAC addresses into account.

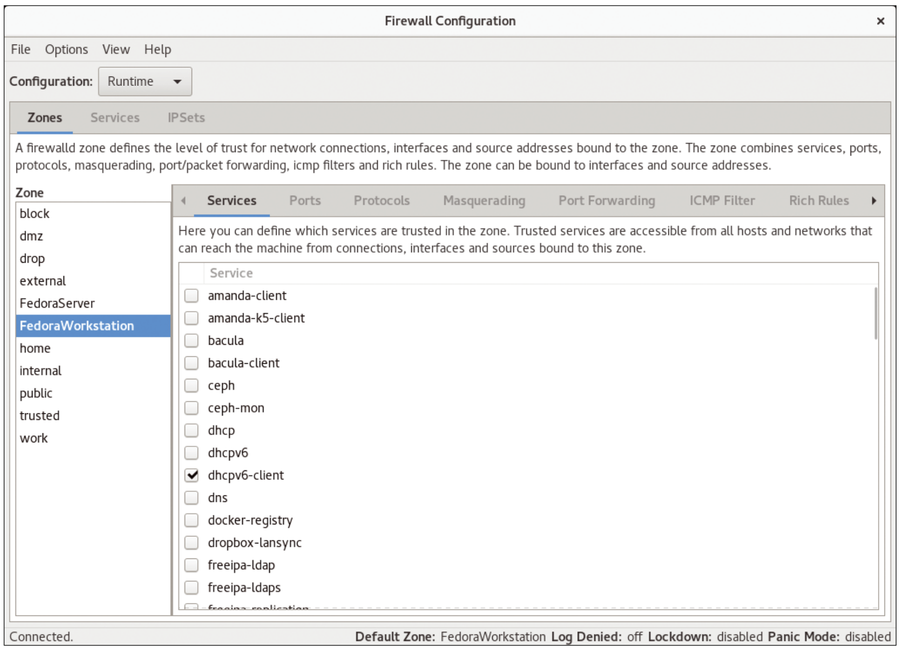
Interplay

Out of the box, firewalld assigns one zone to each of the interfaces physically present in the system. However, a static zone definition is useless for mobile systems that often seek wireless access to the Internet from a wide variety of places. Therefore, you can assign a different zone to each interface at any time. To do this, select *Options* | *Change Zones of Connections* and set up a new connection zone (Figure 4).

Additionally, you can change the default zone for all interfaces in the system at any time by selecting one of the zones offered in the pop-up window in the *Options* | *Change Default Zone* menu. After clicking *OK* and authenticating as an admin, the firewall changes the default zone on the fly.

Panic Mode and Applet

The *Options* | *Panic Mode* setting blocks all connections so that firewalld does not forward any incoming or outgoing packets. An applet installed by the *firewall-applet* package also lets you control the application with a mouse click. The applet automatically sets up shop in the system tray after installation, displaying a red wall icon with a PC behind it. On mouse


Firewall Configuration
File Options View Help
Zones Services IPSets
Services Ports Protocols Masquerading Port Forwarding ICMP Filter Rich Rules

Configuration: Runtime

A firewalld zone defines the level of trust for network connections, interfaces and source addresses bound to the zone. The zone combines services, ports, protocols, masquerading, port/packet forwarding, icmp filters and rich rules. The zone can be bound to interfaces and source addresses.

Zone
block
dmz
drop
external
FedoraServer
FedoraWorkstation
home
internal
public
trusted
work

Here you can define which services are trusted in the zone. Trusted services are accessible from all hosts and networks that can reach the machine from connections, interfaces and sources bound to this zone.

Service
☐ amanda-client
☐ amanda-k5-client
☐ bacula
☐ bacula-client
☐ ceph
☐ ceph-mon
☐ dhcp
☐ dhcpv6
✓ dhcpv6-client
☐ dns
☐ docker-registry
☐ dropbox-lansync
☐ freeipa-ldap
☐ freeipa-ldaps

Connected. Default Zone: FedoraWorkstation Log Denied: off Lockdown: disabled Panic Mode: disabled

Figure 3: The firewalld interface makes iptables easier.

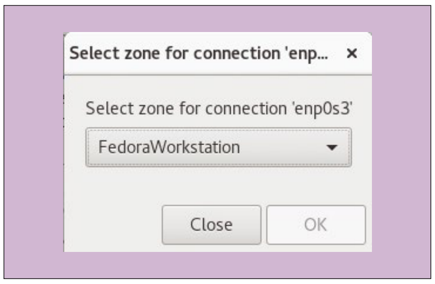

Select zone for connection 'enp... ×
Select zone for connection 'enp0s3'
FedoraWorkstation
Close OK

Figure 4: Thanks to different zones, you can individually adjust the security level of the firewall at any time.

over, it shows the interface used, the default zone, and – if this has changed – the active zone. For WiFi connections, the applet displays the SSID (Figure 5).

Clicking on the applet lets you change the active zone. You can open a small window on the desktop in which to select a new zone without restarting. For the applet to display additional firewall messages (e.g., when zones change or the firewall reloads settings), some distributions also need to change the `/etc/firewall/applet.conf` configuration file. The value of the *notifications* and *show-inactive* options must be set to *true*.

In panic mode, the applet displays an appropriate icon so that you can see that the firewall blocks all packet transfers.

Logging

Contrary to the norm, the firewalld GUI supports virtually no logging functions, which restricts your options for retroactive packet analysis. You can activate logging of all rejected and discarded packets by selecting the *Options | Change Log Denied* entry in the configuration interface and then selecting *all* in the selection field.

fwbuilder

Firewall Builder (fwbuilder) [5] is a dinosaur among the firewall graphical configuration tools with more than 10 years of continuous development history. Accordingly, the software is well positioned on the market and can be found in the repositories of almost all major Linux distributions. Fwbuilder manages firewall systems across platforms and is therefore also suitable for use in heterogeneous environments with Cisco firewalls and BSD's Packet Filter (PF) [17], in addition to iptables.

On Linux, fwbuilder configures iptables with automatic rule validation; it even supports IPv6. The installation adds fwbuilder to the menu structure of the respective desktop. On first launch, you will see two windows: In addition to the configuration window, the routine also calls a smaller Quick Start Guide that introduces the most important functions of the program to new firewall administrators.

In the straightforward configuration window (Figure 6) is a menu at the top with a buttonbar below for quick launching of the most important functions. On the left is a tree view with various objects. On the right, two buttons let you create a new ruleset or import an existing ruleset. A third button calls the system's web browser and opens the Quick Start Guide [18] on the project page.

Wizard

Clicking the *Create new firewall* button opens a dialog that helps you to define new rules with a wizard. The wizard already has templates with useful settings for default firewalls; additional protection rules can be implemented easily.

The objects on the left in the program window contain rulesets and can be organized and extended as object libraries. Interfaces or services also appear as objects. First, you specify a new object name and then tell the routine which firewall – typically iptables for current Linux distributions – and which operating system to expect host-side. Because the firewall selection list correlates with that of the operating systems, the software usually automatically enters the appropriate operating system correctly when you choose a firewall. In the dialog box, you then specify whether you want to use preconfigured rulesets.

Clicking on *Next* takes you to the next dialog. If you want to create the objects manually, the dialog reveals which physical interfaces the system includes. You can search for interfaces via SNMP if it is installed on the host. In the following dialog, define the individual interfaces including the respective mode for IP address assignment. After a final click on *Finish*, the software creates the new object library (Figure 7).

The program window now splits into three areas, with the right pane broken down into a pane for rulesets and another for processing steps, in which you can change individual settings (e.g., for the interfaces). The ruleset is based on iptable syntax; you can extend it by clicking on the green plus symbol in the top left corner of the list. As is usual with iptables, new rules require that all packets are rejected first.

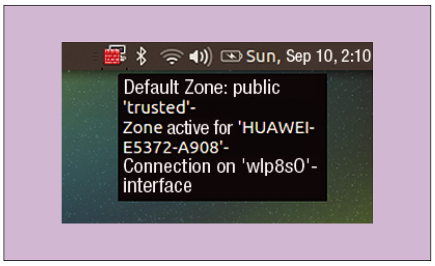

Sun, Sep 10, 2:10
Default Zone: public 'trusted'-
Zone active for 'HUAWEI-E5372-A908'-
Connection on 'wlp8s0'-interface

Figure 5: A firewalld applet gives you a quick overview.

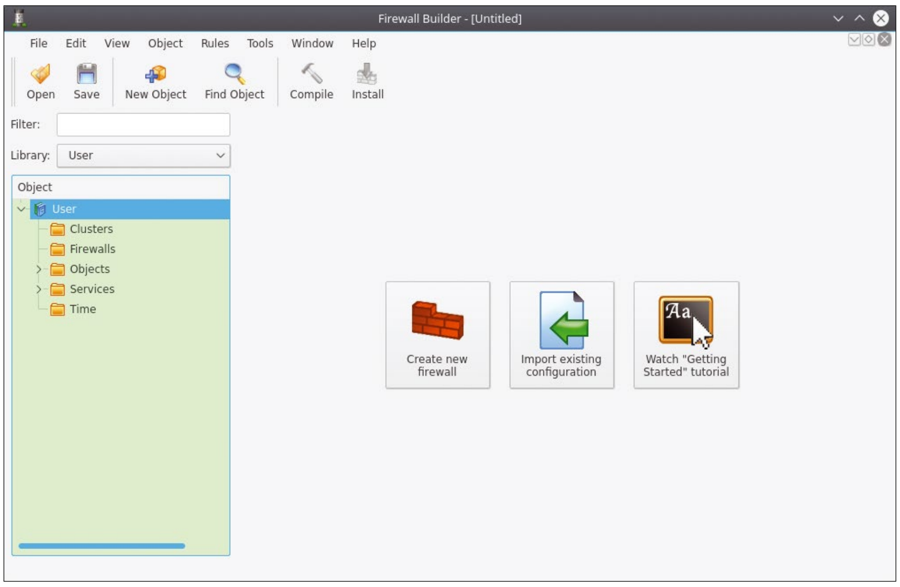
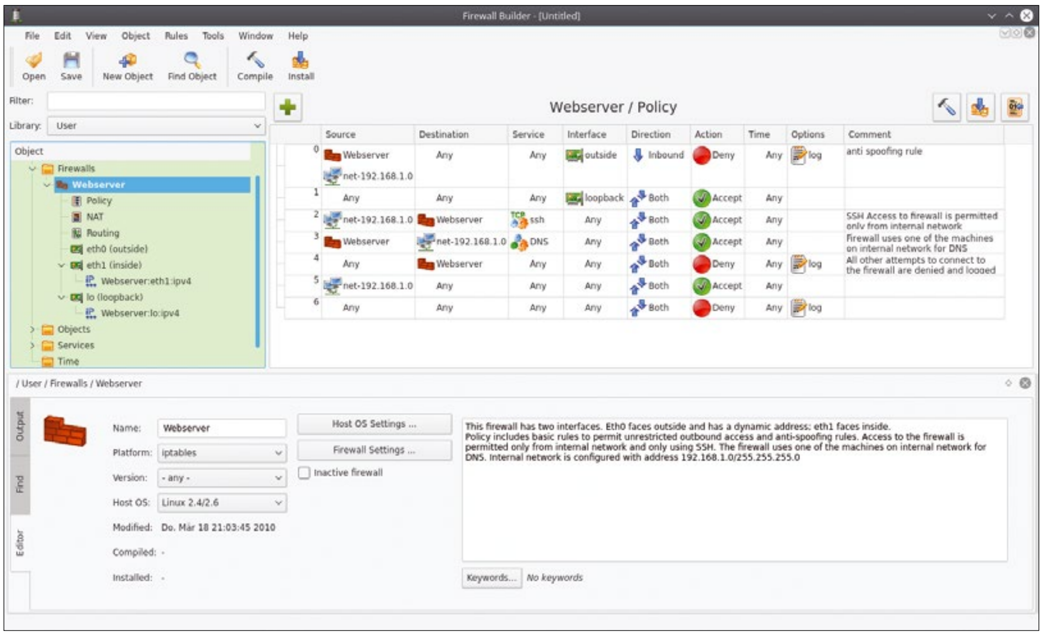

Firewall Builder - [Untitled]
File Edit View Object Rules Tools Window Help
Open Save New Object Find Object Compile Install
Filter:
Library: User
Object
User
Clusters
Firewalls
Objects
Services
Time
Create new firewall
Import existing configuration
Watch "Getting Started" tutorial

Figure 6: Fwbuilder has an intuitive interface.



	Source	Destination	Service	Interface	Direction	Action	Time	Options	Comment
0	Webserver net-192.168.1.0	Any	Any	outside	Inbound	Deny	Any	log	anti spoofing rule
1	Any	Any	Any	loopback	Both	Accept	Any		
2	net-192.168.1.0	Webserver	ssh	Any	Both	Accept	Any		SSH Access to firewall is permitted only from internal network
3	Webserver	net-192.168.1.0	DNS	Any	Both	Accept	Any		Firewall uses one of the machines on internal network for DNS
4	Any	Webserver	Any	Any	Both	Deny	Any	log	All other attempts to connect to the firewall are denied and logged
5	net-192.168.1.0	Any	Any	Any	Both	Accept	Any		
6	Any	Any	Any	Any	Both	Deny	Any	log	

/ User / Firewalls / Webserver

Name: Webserver
Platform: iptables
Version: - any -
Host OS: Linux 2.4/2.6
Modified: Do. Mär 18 21:03:45 2010
Compiled: -
Installed: -
☐ Inactive firewall

This firewall has two interfaces. Eth0 faces outside and has a dynamic address: eth1 faces inside. Policy includes basic rules to permit unrestricted outbound access and anti-spoofing rules. Access to the firewall is permitted only from internal network and only using SSH. The firewall uses one of the machines on internal network for DNS. Internal network is configured with address 192.168.1.0/255.255.255.0

No keywords

Figure 7: Graphically presented rules make it easier to manage the firewall.

You can drag individual objects from the segment with the object libraries and drop them at the desired position in the ruleset. Fwbuilder then automatically adapts the rule to your specifications. If objects are still missing in the library (e.g., if a user adds additional hardware to a host computer), admins can add them at a later time.

Once all rulesets are in place, you need to compile the rules to match the syntax of the respective firewall. The supported host systems range from Linux through various BSD derivatives to Cisco and HP appliances and have very different syntaxes. To proceed, click on the hammer icon in the upper-right corner of the rule list. In the last step, the compiled firewall is transferred to the host system using SSH and SCP (Figure 8). However, if fwbuilder is already running on the host system on which it acts as the firewall, you can click on the *Compile and install this firewall* button.

You can subsequently compare modified or supplemented rules with the original rulesets to filter out inconsistencies. To do this, select *Tools* | *Find Conflicting Objects in Two Files* and specify the files with the rulesets. The software then automatically checks these for inconsistent entries. This is especially useful wherever very complex rule sets and mutual dependencies can cause the admin inadvertently to misconfigure objects when changing the rules.

Gufw

The relatively new Gufw project [6] sees itself as a graphical front end for the Uncomplicated Firewall (UFW) [19], which acts as the default defense for Ubuntu and its derivatives. Gufw and UFW are now available for most of the major Linux distributions and are also included in their software repositories. Like fwbuilder, the dynamic duo UFW and Gufw require netfilter and iptables on the system and can handle both IPv4 and IPv6. Unlike fwbuilder, Gufw does not work across platforms. However, the GUI can be used to manage UFWs on remote computers.

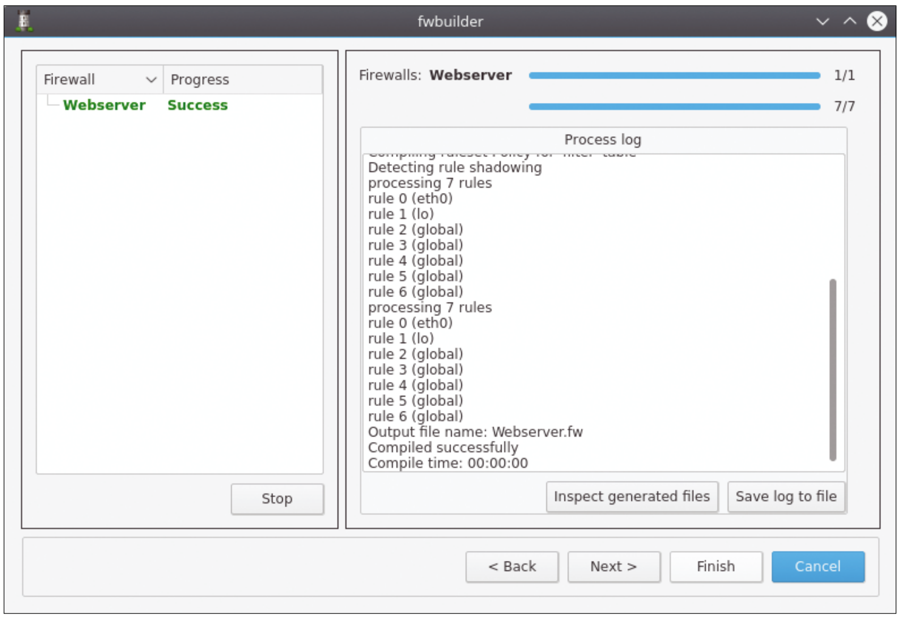
Simple

The Gufw program window is simple and therefore self-explanatory (Figure 9). The firewall is switched on and off by a slider, and, much like firewalld, the system works with profiles that contain different rulesets. This strategy proves to be particularly interesting for users who use Gufw on their laptops, because the device accesses different profiles depending on the type of Internet access.

In the program window, you also use various functions in a table: The *Rules* tab lists the active rules; the *Report* tab shows details of the data transfer. The *Log* tab lists the function log of the firewall in a table and shows the usage history.

Modification

Profiles already exist that can be adapted or supplemented. To do so, click *Edit* | *Preferences* to bring up a dialog in which to create new profiles and modify the corresponding protocol function that individually defines the scope of the history (Figure 10). You can create the rules for the respective profiles in the *Rules* tab in the primary program window.

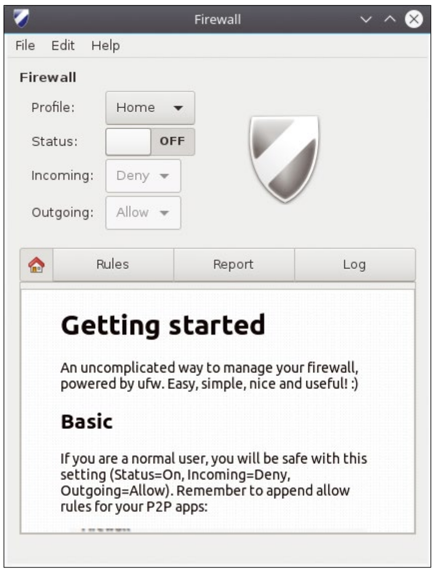


Webserver Success

Firewalls: **Webserver** 1/1
7/7

```
Detecting rule shadowing
processing 7 rules
rule 0 (eth0)
rule 1 (lo)
rule 2 (global)
rule 3 (global)
rule 4 (global)
rule 5 (global)
rule 6 (global)
processing 7 rules
rule 0 (eth0)
rule 1 (lo)
rule 2 (global)
rule 3 (global)
rule 4 (global)
rule 5 (global)
rule 6 (global)
Output file name: Webserver.fw
Compiled successfully
Compile time: 00:00:00
```

Figure 8: Fwbuilder is very flexible in terms of supported host systems.


Firewall

File Edit Help

Firewall

Profile: Home

Status: OFF

Incoming: Deny

Outgoing: Allow

Rules Report Log

Getting started

An uncomplicated way to manage your firewall, powered by ufw. Easy, simple, nice and useful! :)

Basic

If you are a normal user, you will be safe with this setting (Status=On, Incoming=Deny, Outgoing=Allow). Remember to append allow rules for your P2P apps:

Figure 9: The Gufw interface.

A new rule is added by clicking the green plus symbol in the lower-left corner to open a new window. The application filter is immediately noticeable in the window; it predefines useful settings, especially for certain games that require special ports to be enabled for smooth operation. These ports are already stored on the firewall so that you can immediately adopt the appropriate rules.

However, I recommend that you create your own profiles for such applications. When many occur in a default profile, it is a guaranteed security risk, because the firewall keeps specific ports for incoming and outgoing data packets open at all times, even in regular operation.

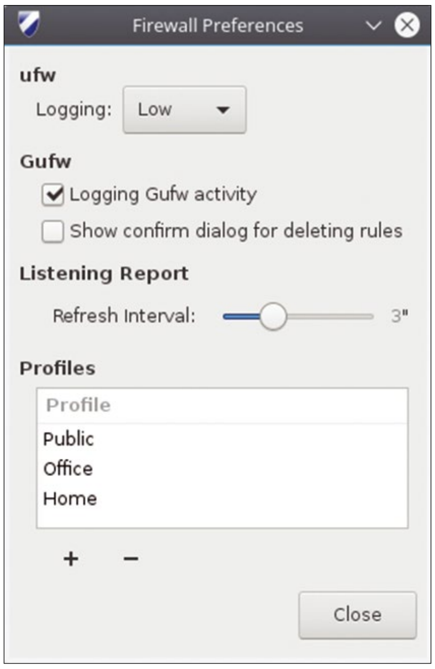
Changing rules is easy, as well: Select the rule to be changed and left-click the button with the gear icon. In a new window, make all the necessary adjustments in a few selection and input fields. When finished, activate the new settings with *Apply* and close the window.

On Record

Gufw displays the firewall's history logs in the *Log* tab. You can save the logs for later documentation and testing purposes, even if only on the clipboard, by clicking on the *Copy to Clipboard* button located below the display area when the *Log* tab is active. Next, paste the log into an editor and save the file. Using the adjacent button, delete logs if necessary.

Shorewall

Shorewall [7] also relies on netfilter and iptables. The graphical management interface for filtering packages is written in Perl; packages are included in most major Linux distributions. Shorewall is suitable both for single workstations and for server systems, which can also be used as gateways with two network connections or in a DMZ with three network interfaces. Shorewall uses Webmin [20] as a GUI or the drakfirewall [21] graphical front end in Mandriva derivatives. This control center module supports configuration in just a few steps (Figure 11).


Firewall Preferences

ufw

Logging: Low

Gufw

☑ Logging Gufw activity

☐ Show confirm dialog for deleting rules

Listening Report

Refresh Interval: 3"

Profiles

Profile
Public
Office
Home

+ −

Close

Figure 10: In Gufw, just a few steps define profiles.

Webmin for distributions outside the Mandriva universe offers many more options for system administration than just configuring the firewall. After installing, you can access it in the browser from the *127.0.0.1:10000* URL. Alternatively, it specifies – if known – the IP address in the local intranet or the hostname.

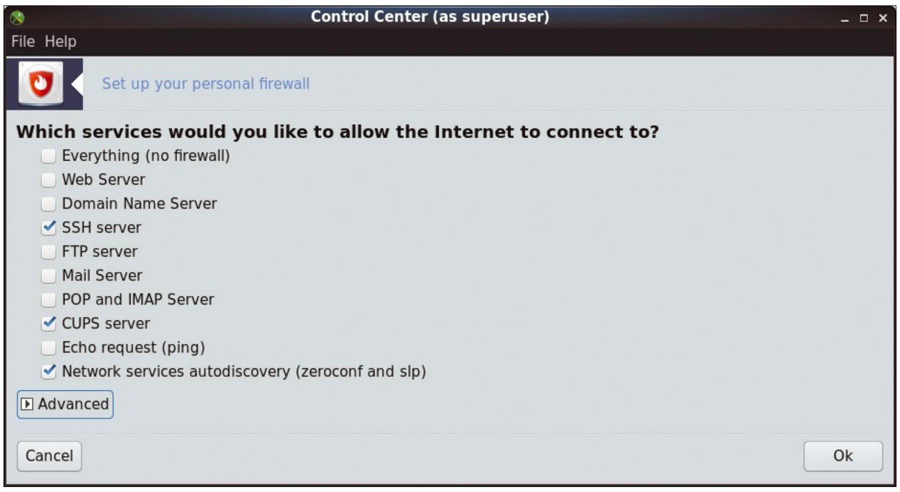
After logging on, you find yourself in the configuration window. In the list area on the left, selecting *Networking | Shoreline Firewall* displays the individual configuration options on the right (Figure 12).

Group Dynamics

The most important options for the existing infrastructure's basic settings can be found in the *Network Zones* and *Network Interfaces* groups. These are joined by the firewall settings options in the *Default Policies*, *Firewall Rules*, and *Blacklist Hosts* groups. The options responsible for routing can be found in the *Routing Rules*, *Additional Routing Providers*, *Masquerading*, and *Static NAT* groups.

You will encounter clear setting dialogs in all configuration groups, which list the available options in tabular form. Below the group selection, you can enable various management options simply by clicking on one of the buttons available there: Clicking on *Apply Configuration* restarts the firewall with your settings.

The *Refresh Configuration* button updates the blacklist and traffic shaping tables, which ensures that the firewall prioritizes certain packets. If you want to reset the firewall settings, click on *Clear Firewall*. The *Stop Firewall* button blocks access from all hosts – with the exception of those excluded from the whitelist. The *Show Status* button tells you about the current operating status of Shorewall.


Control Center (as superuser)

File Help

Set up your personal firewall

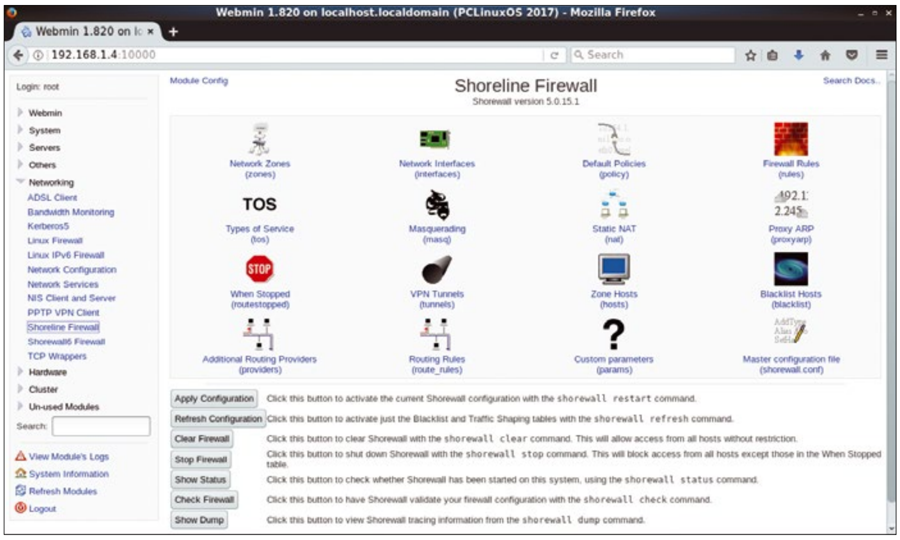
Which services would you like to allow the Internet to connect to?

☐ Everything (no firewall)
☐ Web Server
☐ Domain Name Server
☑ SSH server
☐ FTP server
☐ Mail Server
☐ POP and IMAP Server
☑ CUPS server
☐ Echo request (ping)
☑ Network services autodiscovery (zeroconf and slp)

Advanced

Cancel Ok

Figure 11: With Mandriva-based distributions, Shorewall can be set up with just a few mouse clicks.



Click this button to activate the current Shorewall configuration with the `shorewall restart` command.
Click this button to activate just the Blacklist and Traffic Shaping tables with the `shorewall refresh` command.
Click this button to clear Shorewall with the `shorewall clear` command. This will allow access from all hosts without restriction.
Click this button to shut down Shorewall with the `shorewall stop` command. This will block access from all hosts except those in the When Stopped table.
Click this button to check whether Shorewall has been started on this system, using the `shorewall status` command.
Click this button to have Shorewall validate your firewall configuration with the `shorewall check` command.
Click this button to view Shorewall tracing information from the `shorewall dump` command.

Figure 12: Webmin offers extensive configuration options for Shorewall.

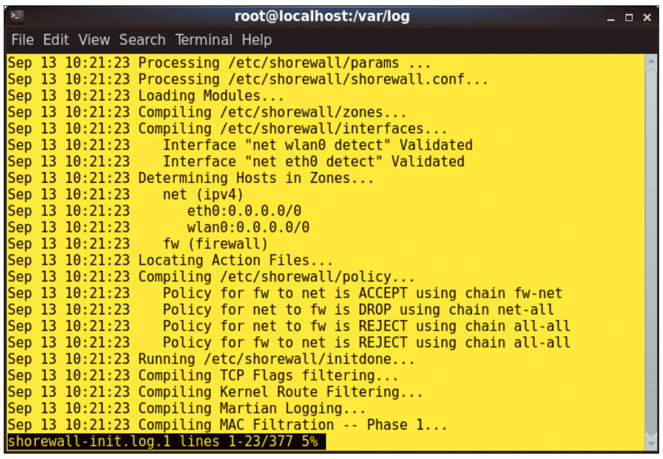
Table 1: An Overview of Firewall GUIs

Feature	firewalld	fwbuilder	Gufw	Shorewall
Requirements	Netfilter	Netfilter and iptables	Netfilter and iptables	Netfilter and iptables
Cross-Platform	No	Yes	No	No
Remote Host	No	Yes	Yes	Yes
IPv4/IPv6	Yes/Yes	Yes/Yes	Yes/Yes	Yes/Yes
Zone Model	Yes	No	Yes (profile)	Yes
Chain and Control Model	No	Yes	Yes	Yes
Dynamic	Yes	No	No	No
Application Integration	No	No	Yes	Limited
Logging	Limited	No	Yes	Yes
Wizard	No	Yes	Limited	No
Primary Application	Server	Server, desktop, cluster, appliance	Desktop	Server, desktop, appliance

The *Check Firewall* button enables a consistency check: Many extensive iptables-based rules and regulations contain errors because of their complexity. They can be detected and fixed before being exploited by malicious attackers.

Logging

Shorewall can log the data traffic and stores the protocols in the `/var/log/` directory. Alternatively, you can view the logfiles in Webmin with the *View Module's Logs* option (Figure 13).



```
Sep 13 10:21:23 Processing /etc/shorewall/params ...
Sep 13 10:21:23 Processing /etc/shorewall/shorewall.conf...
Sep 13 10:21:23 Loading Modules...
Sep 13 10:21:23 Compiling /etc/shorewall/zones...
Sep 13 10:21:23 Compiling /etc/shorewall/interfaces...
Sep 13 10:21:23    Interface "net wlan0 detect" Validated
Sep 13 10:21:23    Interface "net eth0 detect" Validated
Sep 13 10:21:23 Determining Hosts in Zones...
Sep 13 10:21:23    net (ipv4)
Sep 13 10:21:23       eth0:0.0.0.0/0
Sep 13 10:21:23       wlan0:0.0.0.0/0
Sep 13 10:21:23    fw (firewall)
Sep 13 10:21:23 Locating Action Files...
Sep 13 10:21:23 Compiling /etc/shorewall/policy...
Sep 13 10:21:23    Policy for fw to net is ACCEPT using chain fw-net
Sep 13 10:21:23    Policy for net to fw is DROP using chain net-all
Sep 13 10:21:23    Policy for net to fw is REJECT using chain all-all
Sep 13 10:21:23    Policy for fw to net is REJECT using chain all-all
Sep 13 10:21:23 Running /etc/shorewall/initdone...
Sep 13 10:21:23 Compiling TCP Flags filtering...
Sep 13 10:21:23 Compiling Kernel Route Filtering...
Sep 13 10:21:23 Compiling Martian Logging...
Sep 13 10:21:23 Compiling MAC Filtration -- Phase 1...
shorewall-init.log.1 lines 1-23/377 5%
```

Figure 13: Shorewall logfiles are also very helpful.

Conclusions

The firewall GUIs discussed here are all suitable for securing IT infrastructures. However, the individual tools focus on different user groups and scenarios: Gufw is somewhat out of the ordinary, because it primarily targets desktop systems. PeerGuardian is not a classic firewall interface but is limited to working with blocklists. The software does not let you implement complex rule constructs, but it is useful as a firewall extension.

Fwbuilder, firewalld, and Shorewall primarily target server systems and therefore have far more features. Fwbuilder is also suitable for use in heterogeneous environments with various server operating systems and even with some manufacturers' appliances. The software can compile appropriate configuration files depending on a firewall's syntax. On the other hand, Shorewall can be configured with Webmin and thus managed from remote computers.

Thus, you are spoiled for choice (for an overview of features, see Table 1). In the end, your choice depends on your distribution and network security requirements. ■■■

Info

[1] Netfilter: *http://www.netfilter.org*

[2] Nftables: *https://netfilter.org/projects/nftables/*

[3] Iptables: *https://netfilter.org/projects/iptables/*

[4] firewalld: *http://www.firewalld.org*

[5] fwbuilder: *http://www.fwbuilder.org*

[6] Gufw: *http://gufw.org*

[7] Shorewall: *http://shorewall.org*

[8] PeerGuardian: *https://sf.net/projects/peerguardian/*

[9] PeerGuardian blocklists: *https://www.iblocklist.com/lists*

[10] FireStarter: *https://sf.net/projects/firestarter/*

[11] Turtle Firewall: *http://turtlefirewall.sourceforge.net*

[12] FireFlier: *http://fireflier.sf.net*

[13] Vuurmuur: *https://www.vuurmuur.org/trac/*

[14] IPFire: *http://www.ipfire.org*

[15] Untangle NG Firewall: *https://www.untangle.com/untangle-ng-firewall/*

[16] Alpine Linux: *https://alpinelinux.org*

[17] BSD PF: *http://www.openbsd.org/faq/pf/*

[18] Quick Start Guide: *http://www.fwbuilder.org/4.0/quick_start_guide.shtml*

[19] UFW: *https://wiki.ubuntu.com/UncomplicatedFirewall*

[20] Webmin: *http://www.webmin.com*

[21] drakfirewall: *https://doc.mageia.org/mcc/5/en/content/drakfirewall.html*

Neural networks learn from mistakes and remember successes

Car NOT Goat

The well-known Monty Hall game show problem can be a rewarding maiden voyage for prospective statisticians. But is it possible to teach a neural network to choose between goats and cars with a few practice sessions? *By Mike Schilli*

Here's the problem: In a game show, a candidate has to choose from three closed doors; waiting behind these doors is a car, which is the main prize, a goat, and yet another goat (Figure 1). The candidate first picks a door, and then the presenter opens another, behind which there is a bleating goat. How is the candidate most likely to win the grand prize: Sticking with their choice or switching to the remaining closed door?

As has been shown [1] [2], the candidate is better off switching, because then they double their chances of winning. But how does a neural network learn the optimal game strategy while being rewarded for wins and punished for losses?

Human Model

The input and output data must be professionally manipulated – as is always the case with machine learning. An artificial intelligence (AI) system is not a cauldron into which you throw problems and then ready-made solutions just bubble up. In fact, AI algorithms only solve a small number of precisely defined problems.

To solve the problem, the algorithm used in this article is a multilayer neural network that takes three input parameters: the door the candidate selected, the door the presenter opened, and the remaining locked door.

In the perceptron's hidden middle layer, each artificial neuron is connected to each input neuron of the first layer. Even though a neural network does not quite work like a human brain, you can still interpret these massive links as a reflection of the human design. In turn, each of the hidden layer's inner neurons fires pulses to all the output layer's neurons.

Carrot and Stick

In the training phase, we want the network to learn a strategy from played game shows to predict those that indicated winning the prize on the output neuron, depending on the current door constellation. In a few thousand rounds, the script feeds the three input parameters into the AI system and compares the door value at the output with the actual door leading to the car. If the system has predicted the correct door, it is rewarded. If it is wrong, it has to adjust its neurons' parameters via a feedback mechanism.

In AI jargon, the training runs are called episodes. It is often helpful not to adjust the neuron parameters with every dataset, but instead just after a batch of input values. This saves computing time and prevents the system from balancing the weights in wild swings, which often leads to unstable conditions that don't converge into a solution.

1,000 Shows Recorded

Listing 1 records the results of 1,000 game shows in which the prize is placed behind a random door, and then the presenter opens a door that neither leads to the main prize nor is already open. Game results go to a file in CSV format (Figure 2).

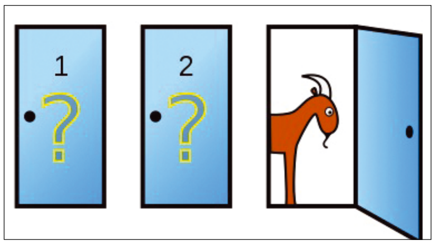


Figure 1: Monty Hall problem on Wikipedia.

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.

It numbers the doors from `0` to `2` and logs the indices of the following doors in each line in this order: the door that is chosen by the game show candidate, the door that the presenter opens, the remaining door, and the one leading to the prize.

For example, if the neural network encounters the `[1,2,0,0]` combination, such as in the first line in the file displayed in Figure 2, it knows that the candidate has chosen the second door (index 1), the presenter has then opened the third (index 2), and the first is still closed (index 0). The main prize was randomly hidden and ended up behind the first door (index 0). With these parameters, to win the game, the neural network must pick the first door.

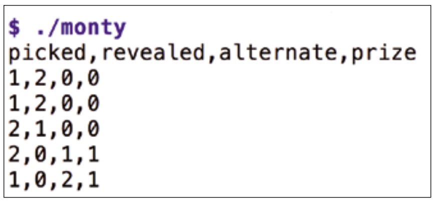
Listing 1 defines two classes, `Door` for individual doors and `Show` for a world with three doors and the rules of the TV show. `Door` objects are initialized either with or without a main prize; line 15 places the prize behind the first door and then lets line 16 shuffle the doors, so that the prize randomly ends up somewhere. Using the `pick()` method from line 22, `randrange()` simulates the candidate picking a random door. The `Show` object remembers the selected door's index in the `picked` instance variable.

Listing 1: monty

```
01 #!/usr/bin/env python3
02 from random import shuffle, randrange
03
04 class Door(object):
05   def __init__(self,prize):
06     self.prize = prize
07
08 class Show(object):
09   def __init__(self):
10     self.picked   = None
11     self.revealed  = None
12     self.alternate = None
13
14       # hide prize behind random door
15     self.prizes = [1,0,0]
16     shuffle(self.prizes)
17
18     self.doors = []
19     for prize in self.prizes:
20       self.doors.append(Door(prize))
21
22   def pick(self):
23       # candidate picks a door
24     idx = randrange(0,len(self.doors))
25     self.picked = idx
26
27       # moderator reveals another door
28     for idx,door in \
29         enumerate(self.doors):
30       if door.prize or \
31           self.picked == idx:
32         continue
33       if self.revealed is None:
34         self.revealed = idx
35       # determine remaining door
36     for idx,door in \
37         enumerate(self.doors):
38       if self.picked != idx and \
39           self.revealed != idx:
40         self.alternate = idx
41
42   def prize_idx (self):
43     for idx,door in \
44         enumerate(self.doors):
45       if door.prize:
46         return idx
47
48 print("picked,revealed,alternate,prize");
49
50 for i in range(1000):
51   show = Show()
52   show.pick()
53   show.reveal()
54
55   print("{0},{1},{2},{3}".format(
56     show.picked, show.revealed,
57     show.alternate,show.prize_idx()))
```

Presenter in a Bind

The presenter then has to open another door in the `for loop` starting at line 28, but must not reveal the main prize. In the `revealed` attribute, the object stores this door's index. The remaining third door's index is then saved in the `alternate` attribute. The `for` loop starting in line 50 iterates over 1,000 game shows, and the `print()` statement on line 55 outputs their results in CSV format. This is, line by line for each show, the indices of the candidate door, the presenter door, the remaining door, and the winning door.



```
$ ./monty
picked,revealed,alternate,prize
1,2,0,0
1,2,0,0
2,1,0,0
2,0,1,1
1,0,2,1
```

Figure 2: A random generator generates results of game shows and outputs them in CSV format for a subsequent training session with a neural network.

One-Hot Encoding

If the AI apprentice employs a neural network and feeds in individual shows as 3-tuples, each paired with a one-part result tuple in the training phase, it won't produce satisfying results because door indices aren't really relevant as numerical values; instead, they stand for categories, each door representing a different category. The AI expert transforms such datasets before the training run into categories using one-hot encoding. If a dataset provides values for *n* categories, the one-hot encoder shapes individual records as *n*-tuples, each of which has one element set to `1`, with the remaining elements set to `0`.

Figure 3 shows an example of how an input series like `[2,1,2,0,1,0]` is converted into six one-hot-encoded matrix rows. The code in Listing 2 uses the `to_categorical()` function from the `np_utils` module of the *keras.utils* package to accomplish this. To return from one-hot encoding back to the

Listing 2: onehot

```
#!/usr/bin/env python3

from keras.utils import np_utils
import numpy

X = numpy.array([2,1,2,0,1,0])
print("org=", X)

onehot=np_utils.to_categorical(X)
print("onehot=", onehot)

a=onehot.argmax(1)
print("back=", a)
```

Listing 3: learn

```
01 #!/usr/bin/env python3
02 from keras.models import Sequential
03 from keras.layers import Dense
04 from keras.utils import np_utils
05 import numpy
06
07 data = numpy.loadtxt("shows.csv",
08         delimiter=",", skiprows=1)
09 X = data[:,0:3]
10 Y = data[:,3]
11
12 categories=np_utils.to_categorical(Y)
13
14 model = Sequential()
15 model.add(Dense(10, input_dim=3,
16                 activation='relu'))
17 model.add(Dense(3, activation='relu'))
18 model.add(Dense(3, activation='sigmoid'))
19
20 model.compile(loss='binary_crossentropy',
21               optimizer='adam')
22 model.fit(X, categories, epochs=100,
23           batch_size=100, verbose=0)
24
25 test_data = numpy.array(
26   [[0,1,2], [0,2,1], [1,0,2],
27    [1,2,0], [2,0,1], [2,1,0]
28   ])
29
30 pred = model.predict(test_data)
31
32 for (idx,row) in enumerate(test_data):
33 for (idx,row) in enumerate(test_data):
```

original value later, use the `argmax()` method provided by `numpy` arrays.

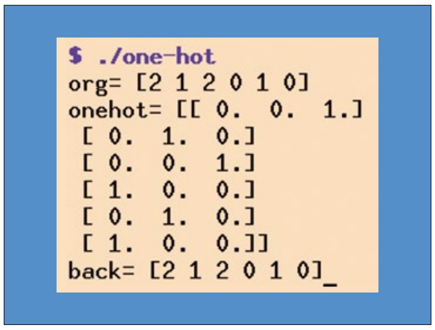
Machine Learning

Armed with the input values in one-hot format, the three-layer neural network method defined in Listing 3 can now be fed with learning data. Important: The network also encodes the output values according to the one-hot method and therefore not only needs a single neuron on its output, but three of them, because both the training and, later on, the predicted values are available as 3-tuples, each of them indicating the winning door as a 1 in a sea of zeros.

The saved training data generated by Listing 1 in `shows.csv` is then read by Listing 3 in line 7. The first three elements of each line are the input data of the network (candidate door, presenter door, alternative door), and the last item indicates the index of the door to the main prize.

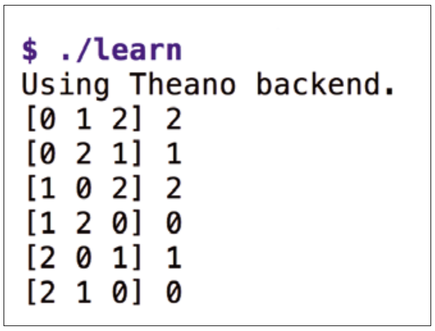
Line 12 transforms the desired output values into categories in one-hot encoding; lines 14 to 18 build the neural network with an entry layer, a hidden layer, and an output layer. All layers are of the `Dense` type; thus, they are networked in a brain-like style connecting with all elements of adjacent layers. The `Sequential` class of the Keras package holds the layers together. Line 20 compiles the neural network model; Listing 3 specifies the error function as `binary_crossentropy` as the learning parameter and selects the `adam` algorithm as the optimizer, which specializes in categorization problems.

In the three-layer model, 10 neurons receive the input data in the input layer and `input_dim=3` sets the data width to 3, since it consists of 3-tuples (values for three doors). The middle layer has three neurons, and the output layer also has three. The latter is, as mentioned above, the one-hot encoding of the results as categories.



```
$ ./one-hot
org= [2 1 2 0 1 0]
onehot= [[ 0.  0.  1.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]
 [ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 1.  0.  0.]]
back= [2 1 2 0 1 0]_
```

Figure 3: One-hot converts values into categories that set one value per tuple to 1.



```
$ ./learn
Using Theano backend.
[0 1 2] 2
[0 2 1] 1
[1 0 2] 2
[1 2 0] 0
[2 0 1] 1
[2 1 0] 0
```

Figure 4: The network has learned that the alternative door offers the most lucrative chance of winning.

Acid Test

The training phase starts in line 22 by calling `model.fit()`. It defines 100 iterations (`epochs`) and a batch size of 100, which defines that only after 100 training values should the neural network adjust its inner weights with the collected information. From line 25, the script visualizes whether the training was successful or not: For all possible door combinations, line 30 calls the `predict()` method to pick a door according to what the network has learned so far. Lo and behold, Figure 4 actually shows that the computer selects the alternative door each time, thus letting the candidate switch to increase their chances of winning in the most optimal way, as the mathematical proof also shows.

This is remarkable, because the network does not know the mathematical correlations, but instead only learns from empirically obtained data. The input data are even somewhat ambiguous, because switching only leads to success in two thirds of all cases. If you forge the input data and hide the prize behind the alternate door every time, the network's internal success metrics rise all the way to 100 percent, and then the network is absolutely sure.

Even if we feed real-world data into the network and the candidate loses in a third of all cases using the switch method, the network optimizes its approach and ends up switching most of the time, with the occasional outlier. Also, if you vary the parameters of the network, for example, the number of epochs or number of neurons per layer, results might vary. As always with these kinds of problems, it's just as much art as science with lots of wiggle room to train an artificially intelligent system successfully. ■■■

Info

[1] "Calculating Probability" by Michael Schilli, *Linux Magazine*, issue 165, August 2014, pg. 60, *http://www.linux-magazine.com/Issues/2014/165/Calculating-Probability*

[2] Monty Hall problem: *https://en.wikipedia.org/wiki/Monty_Hall_problem*

[3] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/magazine/207/*

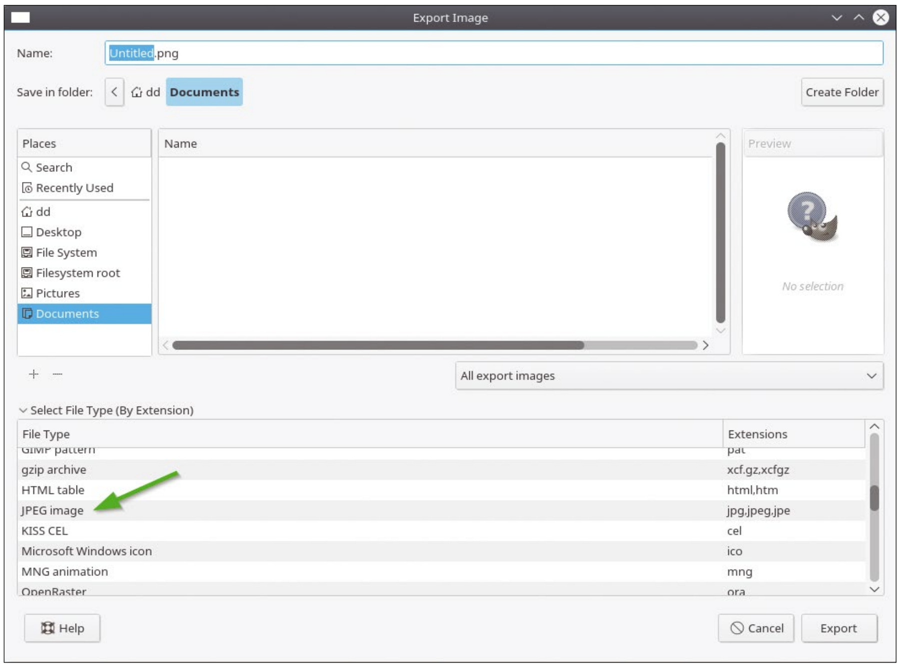
Compress image files with Guetzli

Small Wonders

The Guetzli image optimizer by Google developers produces smaller images than JPEG while maintaining the same quality, but it requires a powerful computer with a large working memory. *By Karsten Günther*

Images play an important role, especially on the Internet. Web hosting providers and users have a keen interest in efficient use of storage space and reduction of load times. With a need to budget its resources wisely, Google therefore set its developers the task of compressing image data.

The result is the Guetzli project – Guetzli being the name for a candy or cookie in the Alemannic German dialect. Under certain, frequently occurring conditions, Guetzli can reduce JPEG images by about a third compared with previous methods. Smaller file sizes then automatically result in quicker website load times.



Export Image

Name: Untitled.png

Save in folder: dd Documents — Create Folder

Places
Search
Recently Used
dd
Desktop
File System
Filesystem root
Pictures
Documents

Name — Preview: No selection

All export images

Select File Type (By Extension)

File Type	Extensions
GIMP pattern	pat
gzip archive	xcf.gz,xcfgz
HTML table	html,htm
JPEG image	jpg,jpeg,jpe
KISS CEL	cel
Microsoft Windows icon	ico
MNG animation	mng
OpenRaster	ora

Help — Cancel — Export

Figure 1: Gimp can store JPEG images in different variants.

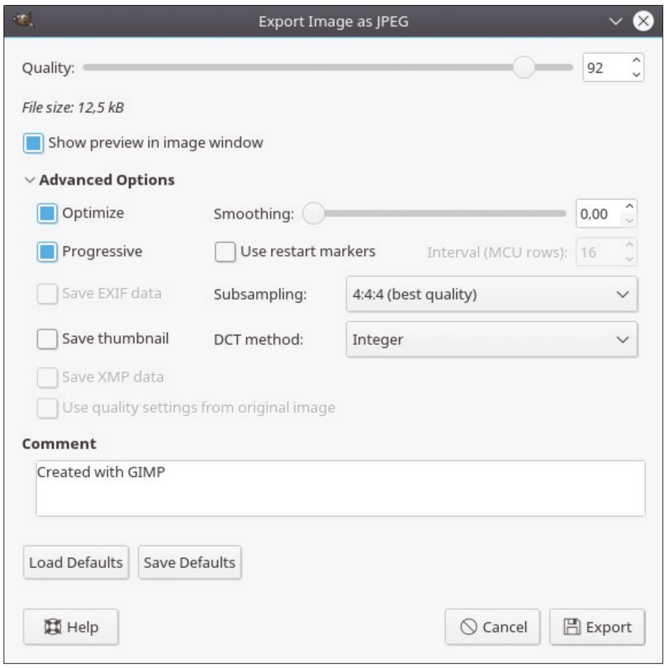
Optimizing JPEGs in Gimp

When exporting images to the JPEG format in Gimp (*File | Export As*), under the *Select File Type (By Extension)* drop-down (Figure 1), select *JPEG image* or, if you have installed the corresponding plugin, *JP2 image*.

In the following Export Image as JPEG dialog, the best approach is to choose *Show preview in image window* so that Gimp computes the file size of the exported images (Figure 2). It takes a bit of time for larger images, but it can help you find optimum parameters. Gimp displays several coding options under the *Advanced Options* drop-down.

In addition to the *Quality* slider, the most important correcting variables are *Subsampling* and *Smoothing*. *Quality* is a fairly intuitive parameter: How far you can reduce it largely depends on the specific image. *Smoothing* causes moderate softening that is often pleasing to the human eye.

Under *Subsampling*, you can directly tweak the algorithm and define the quality of the results. Better quality with the *4:4:4 (best quality)* option costs computing time and results in larger files. Computing time can be influenced by the *DCT method*, as well: *Fast Integers* is the quickest option, with only minor quality loss.



Export Image as JPEG

Quality: 92

File size: 12,5 kB

[✓] Show preview in image window

Advanced Options

[✓] Optimize — Smoothing: 0,00

[✓] Progressive — [ ] Use restart markers — Interval (MCU rows): 16

[ ] Save EXIF data — Subsampling: 4:4:4 (best quality)

[ ] Save thumbnail — DCT method: Integer

[ ] Save XMP data

[ ] Use quality settings from original image

Comment

Created with GIMP

Figure 2: The Advanced Options section manages many details when you export JPEG images.

The Chronicles of JPEG

For a long time, JPEG has been the trusted standard for image compression, but a closer look reveals that the algorithm sometimes produce rather unpleasant artifacts that are difficult to remove. Lossy algorithms remove details from images that are barely visible to the human eye to create larger uniform structures that can then be better compressed, whereas the lossless algorithms, such as JPEG2000 or JPEG-LS [1] generate larger image files. Some variants of Libjpeg [2] support the lossless method, so you should be able to use them under Linux with the appropriate software.

Dropbox made the first advance in JPEG algorithms by introducing Lepton [3], the image compressor, which retroactively compresses JPEGs. However, hardly any program can display the resulting images; to display them, you first have to convert back to the original JPEG. Gimp also can optimize JPEGs (see the "Optimizing JPEGs in Gimp" box). Although it is a fairly lengthy manual process, Gimp's scripting additions can automate the process.

Command-Line Goodies

The `guetzli` command implements the method from the command line. Calling the program without parameters shows the few, extremely simple options (Listing 1).

The `--quality` switch is the most important parameter. The developers have tried to maintain compatibility with the quality parameters used by Libjpeg (the most common JPEG converter) to ensure compatibility. To avoid visible artifacts, Guetzli only allows values of `--quality 84` or higher.

Practical use of Guetzli is extremely simple. Calling Guetzli with the following pattern converts an image into a Guetzli JPEG:

```
$ guetzli --quality <nn> <input> ⮐
  <output>.jpg
```

Because Guetzli completely loads images for editing into main memory, you can also overwrite the input file, if necessary:

```
guetzli bild.jpg bild.jpg
```

The program does not output any warnings.

Input files can only be JPEG or PNG formats; any other format will result in the *Error reading JPG data from input file* error message. Guetzli expects images with the sRGB color profile and a gamma of 2.2 – a common default for cameras. The tool ignores any other color profile, so you have to convert the output image, if necessary, to prevent incorrect colors. For a PNG with an alpha channel, Guetzli converts the transparent areas to black.

As well as Guetzli works, it has some downsides: First, it requires significantly more computing power than the algorithms provided by Libjpeg. Second, the RAM requirement is unusually high, at around 300MB per megapixel. Thorough analysis costs time and space. Third, you might have to build Guetzli yourself (see the "Build Your Own Guetzli" box).

Psychovisual Model

It seems amazing that there are still ways to improve such well-established

Listing 1: Guetzli Options

```
$ guetzli
Guetzli JPEG compressor. Usage:
guetzli [flags] input_filename output_filename

Flags:
  --verbose    -- Print a verbose trace of all attempts to standard output.
  --quality Q  -- Visual quality to aim for, expressed as a JPEG quality value.
  --memlimit M -- Memory limit in MB. Guetzli will fail if unable to stay under
                  the limit. Default is 6000 MB
  --nomemlimit -- Do not limit memory usage.
```

Build Your Own Guetzli

Not all common distributions have Guetzli in their package sources. You will find the latest version on GitHub [4], where the developers are constantly working on the code. Recently, they reduced memory usage by 16 bytes per pixel – quite important in light of Guetzli's high memory requirements. Arch Linux recommends installing the Git version in the AUR repository.

A Guetzli dependency is the current version of `libpng-devel`. However, when building, the developers do not rely on known methods; instead, they use Bazel [5], which is rarely used under Linux and might initially require some installation work. Subsequently, the

```
bazel build -c opt //:guetzli
```

command is sufficient. The source code provides the required "recipe."

methods as JPEG coding. All JPEG algorithms have ways to reduce data and use a (run-time) compression method to filter out and remove parts of the image that are not visible to the human eye.

Guetzli uses a new "psychovisual model," as the developers explain: "Guetzli tries to balance minimum loss and file size by using a search algorithm that seeks to overcome the difference between the psychovisual modeling of the JPEG format and the psychovisual model of Guetzli." Guetzli's algorithm, referred to as a "human visual processing system model" for the detection of essential components of images is based on a method called Butteraugli, which is also available as a separate tool [6].

The special trick of the applied Guetzli method lies in the visual masking of image information by a nearby second (strong) stimulus. In some aspects, it is similar to the Blur mask, the best of Gimp's sharpening filters. In this case, the algorithm accentuates an edge in the image by a second edge in the immediate vicinity. The Blur mask and Guetzli are oriented toward human visual perception and do not necessarily generate mathematically accurate results.

Because the models used to detect relevant image information differ, even with JPEG converters, the algorithms based on the models also differ and work effectively in different ways. It is useful to remove the "correct" parts of images to obtain a small file size and good visual quality.

The relatively universal methods based on visual perception are referred to as being "perceptual." However, they differ in many details. You will find detailed information about the Guetzli algorithm in the Google developer's blog [7], where the programmers show by example that environmental artifacts (ringing artifacts) in Guetzli are fewer than in other algorithms.

The developers regularly experiment with several images to check how well the algorithm works. They report a test in which more than 75% of the subjects rated the images generated with the Guetzli algorithm better than those produced by Libjpeg. However, occasional reports from users criticize the quality of Guetzli images, but these statements are not very convincing because the images that were used are often missing and the criticism therefore cannot be understood.

However, Guetzli goes one step further when compressing the data with a "quantization stage of compression" method that reduces the volume of data [8]. In recent years, this method has been developed and optimized continuously by chief developer Jyrki Alakuijala, which has led to a whole series of programs that handle the recompression process.

Other Programs

Also in the Guetzli GitHub repository under `tools` is the `guetzli-compare.py` heat map program that is created when compiling and can be used to display the differences between the original image and the edited version. The Butteraugli program is also part of Guetzli, in a broader sense, and can be built and installed separately. As far as quantization is concerned, in the Guetzli environment, you will find a number of tools that optimize PNG images and even reduce the size of ZIP and Gzip archives.

In 2013, the developers first presented an algorithm to the public that became known as Zopfli [9]. As with ZIP, Gzip, and PNG, it supports the DEFLATE output format, but does so more thoroughly, which costs more computing power and makes the process particularly interesting for once generated, often read and distributed files. Using this method, the Gimp images are reduced better than with BZIP2; however, the version created by XZ Utils [10] is often even smaller.

The Zopfli suite implements several programs (Table 1). The most universal, `advdef` (the AdvanceCOMP deflate compression utility), is used to recompress existing images. The current implementation requires a large amount of memory, which limits the size of the input data. The `-z` switch activates recompression. The `-0` to `-4` parameters control (increasingly) the degree of recompression.

With `-i <number>`, you can influence the number of iterations and, therefore, the optimization indirectly; the default is 15 rounds. Higher values improve the outcome for the `-3` and `-4` modes but require more computing power. Through `-f` (force), Advdef creates the output files, even if they are bigger than the input files. The `-h` option explains which options and parameters control the processing.

In 2015 the developers provided the public with a further development in Zopfli [11]: a variant of LZ77 and Huffman codings that again promises better results. The only program for compression is `bro`.

Table 1: Zopfli Programs

Program	Function
`advdef`	Recompress PNG, MNG, GZ, TGZ, and SVGZ
`advzip`	Compress ZIP
`advpng`	(Re)compress PNG
`advmng`	Compress MNG (animated PNGs)

Conclusions

Guetzli and its relatives show that the opportunity for innovations can still be found in image editing. In addition to special visual optimizations are the useful methods developed for quantization. In particular, operators of websites that use JPEG or PNG images should check to see whether and to what extent the programs presented here are suitable for their purposes. ■■■

Info

[1] JPEG-LS: *https://jpeg.org/jpegls/index.html*

[2] Libjpeg: *https://github.com/thorfdbg/libjpeg*

[3] Lepton: *https://github.com/dropbox/lepton*

[4] Guetzli: *https://github.com/google/guetzli*

[5] Bazel: *https://bazel.build/*

[6] Butteraugli: *https://github.com/google/butteraugli*

[7] Guetzli announcement: *https://research.googleblog.com/2017/03/announcing-guetzli-new-open-source-jpeg.html*

[8] Guetzli information: *https://designmodo.com/guetzli/*

[9] Zopfli: *https://github.com/google/zopfli*

[10] XZ Utils: *https://tukaani.org/xz/*

[11] Brotli: *https://github.com/google/brotli*

The sys admin's daily grind: Smorgasbord

Quarrying

Sys admin columnist Charly Kühnast has an electronic note box in which he collects ideas and small snippets of code. He calls it his "quarry" and is taking this opportunity to offer up some collectors' items to regular readers. *By Charly Kühnast*

My life as a geologist started at a time before Git. Originally, I collected code in my "quarry" if I suspected I would need it again one day. Meanwhile, I hoard artifacts from configuration files, keyboard shortcuts that I can't remember for various shells, and names of tools I want to try.

Usually, none of them appear on this page, because each note contains no more than a few lines of text. That's a pity, so today I'm serving up a wildly mixed selection of notes in the hope that everyone will find something new.

The first note is a configuration snippet for Postfix. Many readers will be familiar with email featuring subject lines straight out of hell:

AW:Re:AW:Answer:Re:AW: <Subject>

The following line, built into Postfix's `header_checks.cf`, shortens the mess to a civilized *re: <Subject>*.

```
/^Subject:\s*((Re|AW|Answer):\s*)+↩
  (.*)$/REPLACE Subject: Re: $3
```

Sys admins live in the shell, mostly Bash. There, they often have to do things with root privileges. Switching there, be it permanently (`su`) or just once (`sudo`), is annoying. The Bash alias

```
alias iddqd='sudo su -'
```

not only saves a few characters, but also proves that its initiator is, first, a nerd with dubious humor and, second, no longer a young man. (The sequence *iddqd* enabled God Mode in the Doom shooter game in 1993).

The next alias does things in a different, but also smart, way:

```
alias but='history -s sudo ↩
  $(history -p \!\!) && sudo ↩
  $(history -p \!\!)'
```

It is medicine for the "Permission denied" disease that every forgotten `sudo` reliably triggers and is fixed as soon as you type `doch` (of course!). The shell then runs the failed command again, but preceded by `sudo`. Typing `sudo!!` would have the same effect, but that requires three more keystrokes and is less cool.

Properly Appended

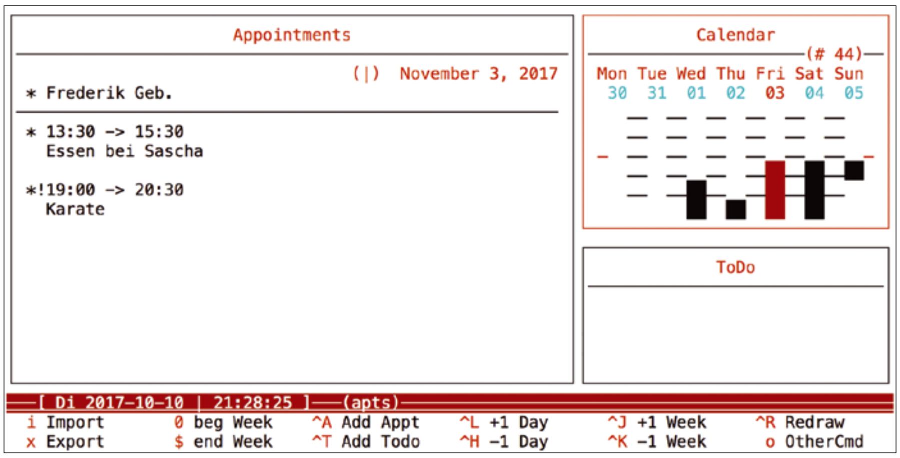
Searching and replacing in the last entry of the Shell history helps laziness win an important battle. For example, if you typed `tail /var/log/syslog`, but actually wanted to see the end of `/var/log/mail`, then appending

```
^syslog^mail^
```

replaces the wrong substring. Bash runs the modified command immediately.

Ah! My "tools" list also includes `calcurse`, a console-based calendar available in almost every distribution. It imports third-party calendars in the common ICS format, supports scripting, and provides information on upcoming appointments in several ways (Figure 1).

I have just found another piece of paper that says: "Change the note box to SVN." It's from 2006. ■■■



```
                Appointments                              Calendar
                                                  ---------------(# 44)--
                        (|)  November 3, 2017      Mon Tue Wed Thu Fri Sat Sun
* Frederik Geb.                                     30  31  01  02  03  04  05

* 13:30 -> 15:30
  Essen bei Sascha

*!19:00 -> 20:30
  Karate
                                                            ToDo

---[ Di 2017-10-10 | 21:28:25 ]---(apts)---
 i Import      0 beg Week   ^A Add Appt   ^L +1 Day   ^J +1 Week   ^R Redraw
 x Export      $ end Week   ^T Add Todo   ^H -1 Day   ^K -1 Week    o OtherCmd
```

Figure 1: Thanks to calcurse, Charly didn't miss his Friday noon lunch appointment and put on his black belt punctually at 7pm.

Author

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

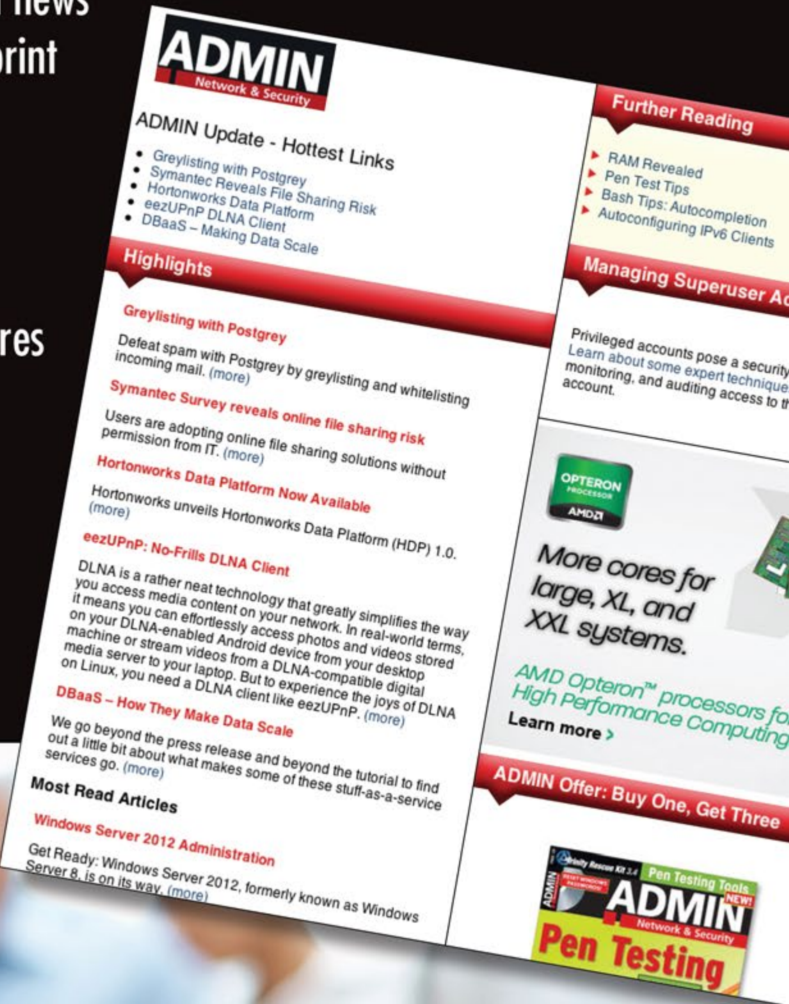
Too Swamped to Surf?

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

Subscribe today to our free ADMIN Update newsletter and receive:

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

ADMIN Network & Security



ADMIN Network & Security

ADMIN Update - Hottest Links

- Greylisting with Postgrey
- Symantec Reveals File Sharing Risk
- Hortonworks Data Platform
- eezUPnP DLNA Client
- DBaaS – Making Data Scale

Highlights

Greylisting with Postgrey

Defeat spam with Postgrey by greylisting and whitelisting incoming mail. (more)

Symantec Survey reveals online file sharing risk

Users are adopting online file sharing solutions without permission from IT. (more)

Hortonworks Data Platform Now Available

Hortonworks unveils Hortonworks Data Platform (HDP) 1.0. (more)

eezUPnP: No-Frills DLNA Client

DLNA is a rather neat technology that greatly simplifies the way you access media content on your network. In real-world terms, it means you can effortlessly access photos and videos stored on your DLNA-enabled Android device from your desktop machine or stream videos from a DLNA-compatible digital media server to your laptop. But to experience the joys of DLNA on Linux, you need a DLNA client like eezUPnP. (more)

DBaaS – How They Make Data Scale

We go beyond the press release and beyond the tutorial to find out a little bit about what makes some of these stuff-as-a-service services go. (more)

Most Read Articles

Windows Server 2012 Administration

Get Ready: Windows Server 2012, formerly known as Windows Server 8, is on its way. (more)

Further Reading

- RAM Revealed
- Pen Test Tips
- Bash Tips: Autocompletion
- Autoconfiguring IPv6 Clients

Managing Superuser Ac

Privileged accounts pose a security
Learn about some expert technique
monitoring, and auditing access to th
account.

OPTERON PROCESSOR AMD

More cores for large, XL, and XXL systems.

AMD Opteron™ processors fo
High Performance Computing

Learn more >

ADMIN Offer: Buy One, Get Three

Pen Testing Tools NEW! ADMIN Network & Security Pen Testing

© Yuri Arcurs, fotolia.com

www.admin-magazine.com/newsletter

FLAC: The premier digital audio codec

Lossless Listening

With a little effort, you can create digital audio files with CD quality sound. *By Bruce Byfield*

Many music listeners content themselves with the MP3 format. However, for dedicated audiophiles, the preferred digital format is the Free Lossless Audio Codec (FLAC) [1]. FLAC files are supported by most hardware and music players, as well as editors like Audacity. However, creating FLAC files or even ripping to FLAC on Linux requires installation of the *flac* package for converting to and from FLAC. The package is available for most distributions but is not always installed by default.

Developed by the Xiph.Org Foundation [2], which also provides the reference implementation for the OGG formats [3], FLAC is preferred by music lovers for numerous reasons. To start with, it is licensed under the New BSD license [4], unencumbered by patents, and does not support digital rights management. In fact, the codec's website specifically states that "there is no intention to add any copy prevention methods." Also, being free-licensed, it is easy to convert to other formats and does not require vendor lock-in for users to continue to have access to a file's data.

Just as important are the features of FLAC files. If you compare the same track in MP3 (`.mp3`) and FLAC (`.flac`) formats, you will notice that the FLAC file can be as much as 250 percent larger. The main reason for this difference is that, while both formats compress audio information, MP3 is a lossy format [5], whereas FLAC is lossless [6]. In other words, when compressing audio information, MP3 discards content and replaces it with approximations that are acceptable to the ear, and FLAC does not. A lossy format might be preferable when transmitted over limited bandwidth or acceptable when played in a noisy environment, such as on a busy street, but if your priority is the faithfulness of sound reproduction, then a lossless format like FLAC offers better playback. A decade ago, lossy formats were popular because storage was limited on handheld music players. Today, with the availability of 250GB microSD cards and players with high-end digital-to-analog converters (DACs) [7], lossless formats like FLAC are becoming more popular than ever. Independent music labels, for instance, routinely offer FLAC downloads per fanbase requests.

Additionally, FLAC offers other advantages. For example, among lossless formats, FLAC offers the best ratio of compression to speed in decoding for play [8]. The information to decode each frame is contained within the frame, rather than being dependent on other frames. In its OGG FLAC format (`.oga`), unlike other lossless formats, FLAC thereby limits the damage to the frame in which transmission errors occur. Still another advantage is that FLAC supports tags, album art, seek tables, and cue sheets. Free-licensed, high-quality, and efficient, FLAC sets a standard that no other audio format currently in widespread use can match.

FLAC Encoding and Decoding

The `flac` command installs with an incomplete man page. To get a summary of the command's options, type `flac -h`. To get a full explanation of available options, type `flac -H`. Once a system has the *flac* package installed, the command structures used in a shell can also be added to audio editors like K3B, which are a front end to the command, to act as the default command for ripping CDs.

Author

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

Photo by Alphacolor 13 on Unsplash

To encode a file to FLAC or decode a `.flac` file to another format, the basic command structure is:

```
flac [<general-options>] ⮐
  [<encoding/format-options>] ⮐
  [FILE1] [FILE2]
```

Unless the `--decode` (`-d`) option is specified, `flac` assumes it is encoding to FLAC. The source file(s) must be in another lossless format (e.g., PCM WAVE (WAV), RF64, AIFF, AIFF-C, or a raw audio file); otherwise, the conversion would be pointless, because `flac` cannot add information that was discarded in making a lossy file.

During processing, `flac` lists any warnings or errors, the size of the output file, and the compression ratio achieved. The default is to create a `.flac` file, but the command can be made to create an Ogg FLAC file with `--ogg` or another lossless format with the options `--force-raw-format`, `--force-aiff-format`, `--force-rf64-format`, and `--force-wave64-format`. You will need to use an audio editor or converter to create a copy of a `.flac` file in a lossy format. Somewhat surprisingly, adding other options does not change this basic output to any degree unless a warning or error occurs (Figure 1).

Whether you are encoding or decoding, any number of files can be converted at the same time by adding them at the end of the command in a space-separated list. By default, the `flac` command writes each input file to a new file with the same name but a `.flac` extension. If you want to change the name of a single file, add `--output-name=FILENAME` (`-o=FILENAME`). Multiple files can be renamed and prefixed with the option `--output-prefix=STRING`. An output string can also be used for outputting to a different directory, so long as the string ends with a forward slash (/).

When encoding, you also have the option of using `--verify` (`-V`), which decodes while encoding and compares the two results to check for errors. Warnings do not stop processing, but when decoding, you can add `--decode-through-errors` (`-F`) to change the `flac` default behavior of stopping decoding when an error occurs. Be aware, though, that decoding through errors can result in missing samples or silent sections in the output file. If you choose, you can also add the general option `--delete-input-file` when encoding or decoding. Better yet, for safety, combine `--delete-input-file` with `--warnings-as-errors` (`-w`), which will stop the creation of the output file but leave the input file untouched, because it is only deleted when the output file is successfully created.

Most CDs consist of a single file, typically in WAV format, with individual tracks marked by seek points. Unlike audio editors like abcde [9], `flac` cannot rip the CD file into separate tracks in a single operation. However, you can use the `--until` option to stop the creation of an output file at a certain point; `--until` can be completed by a track number, so that `--until=2` includes only the first two tracks in the output file. Alternatively, if you have an album cover or a CD label that lists the length of songs, you can complete the option by specifying a time using the *mm:ss:ss* format (minutes:seconds:fraction of a second), so that `--until=03:58:00` would limit the output to just under the first four minutes of the CD file. You can use the same completions with the `--skip` option to begin the output file at a particular track.

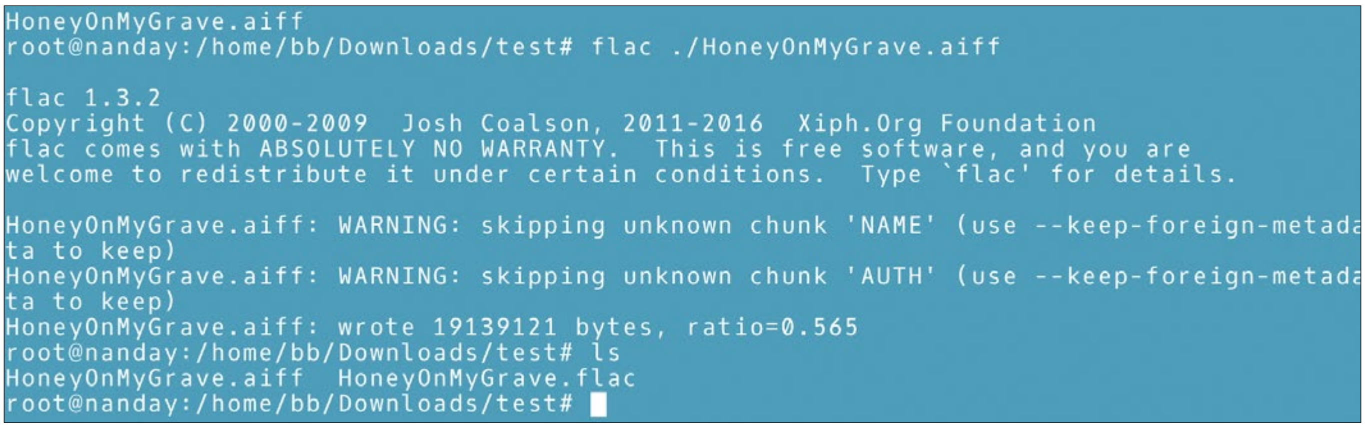
However, the most important options are probably the compression options. The `flac` command includes nine compression levels, the lowest being `--compression-level-0` (`-0` or `--fast`) and the highest being `--compression-level-8` (`-8` or `--best`). All these compression levels are collections of several other options with different settings, including

- `--blocksize` (`-b`)
- `--mid-side` (`-m`)
- `--adaptive-mid-side` (`-M`)
- `--rice-partition-order` (`-r`)

plus some of the Tukey functions available with `--apodization` (`-A`). For example, according to the `-H` option, compression level 0 is synonymous with `-l 0 -b 1152 -r 3`. However, this jargon probably means as little to most users as it does to me (without further research) and takes longer to enter. For most users, most of the time, what matters is that the compression level can reduce the size of `.flac` files by almost a third and that the degree of compression makes no observable difference when a `.flac` file is played back on local equipment, but might be noticeable when streaming – something that would require experimentation even if you were using the individual options. For most users, the compression level options should usually be enough.

Other Options

For many users, the basic options for encoding and decoding `.flac` files is all they need. However, some users may choose to use options for pictures and metadata tags.



```
HoneyOnMyGrave.aiff
root@nanday:/home/bb/Downloads/test# flac ./HoneyOnMyGrave.aiff

flac 1.3.2
Copyright (C) 2000-2009  Josh Coalson, 2011-2016  Xiph.Org Foundation
flac comes with ABSOLUTELY NO WARRANTY.  This is free software, and you are
welcome to redistribute it under certain conditions.  Type `flac' for details.

HoneyOnMyGrave.aiff: WARNING: skipping unknown chunk 'NAME' (use --keep-foreign-metada
ta to keep)
HoneyOnMyGrave.aiff: WARNING: skipping unknown chunk 'AUTH' (use --keep-foreign-metada
ta to keep)
HoneyOnMyGrave.aiff: wrote 19139121 bytes, ratio=0.565
root@nanday:/home/bb/Downloads/test# ls
HoneyOnMyGrave.aiff  HoneyOnMyGrave.flac
root@nanday:/home/bb/Downloads/test# █
```

Figure 1: The `flac` command's default behavior is to encode. The command gives solutions for any warnings or errors, in case you want to recreate the file.

Table 1: Picture Types

Type	Description
0	Other
1	32x32p “file icon” (PNG only)
2	Other file icon
3	Cover (front)
4	Cover (back)
5	Leaflet page
6	Media (e.g., label side of CD)
7	Lead artist/lead performer/soloist
8	Artist/performer
9	Conductor
10	Band/orchestra
11	Composer
12	Lyricist/text writer
13	Recording location
14	During recording
15	During performance
16	Movie/video screen capture
17	A brightly colored fish
18	Illustration
19	Band/artist logotype
20	Publisher/studio logotype

Users who want to recreate digitally the LP or CD album experience may choose to store pictures as part of a `.flac` file. These pictures will display while the `.flac` is playing. The full command structure for a picture is:

```
--picture=TYPE MIME-TYPE DESCRIPTION ↵
  PIXEL-WIDTHxHEIGHTxCOLOR-DEPTH ↵
  COLORS FILE
```

Only the file needs to be specified. The picture types are listed in Table 1. If some parts of the specification are optional, FLAC uses its defaults; the type, for instance, defaults to `3`, which is the front cover. Other options, such as MIME type, dimensions, or colors will be read from the file if not specified. Multiple pictures can be added to the file, although only one each of types `1` and `2`.

The `.flac` files use the same system for metadata tags as Xiph.org’s Ogg formats. When encoding to `.flac` format, users can use the option `--keep-foreign-metadata` to preserve the tags used by other systems, such as the one used in WAV or AIFF files. Unless `--no-utf8-convert` is also used, existing metadata will be converted to UTF-8, which is usually not a problem but might conceivably cause some unexpected characters to display. Both these options will affect all tags specified in the command after them.

You can add additional tags during encoding with the option `--tag=FIELD=VALUE` (`-T`), using one option for each new tag. After you create a `.flac` file, you can use `metaflac`, a utility installed with the FLAC codec, to add tags using the option `--set-tag=NAME=VALUE` or to remove them with `--remove-first-tag=NAME` or `--remove-all-tags`. The `metaflac` command can also perform many of the functions of `flac`, including importing a picture with `--import-picture-from=` and suppressing the conversion of tags to UTF-8 with `--no-utf8-convert`.

Worth the Wait

As the compression example shows, the FLAC codec offers many more options, many of which are of interest mainly to audio experts. However, the Xiph.org Foundation takes its obligation as a reference implementation seriously, and the `.flac` format is easy for anyone to begin to use.

Of course, like any command, `flac` requires some effort to master. However, if music is important to you, the results are worth the effort: digital files with CD quality sound. Once you have heard the difference that a lossless format like FLAC can make, you may never be satisfied with MP3 files again. ■■■

Info

[1] FLAC: *https://xiph.org/flac/index.html*

[2] Xiph.org Foundation: *https://www.xiph.org/*

[3] Ogg formats: *https://xiph.org/ogg/*

[4] New BSD license: *https://opensource.org/licenses/BSD-3-Clause*

[5] Lossy compression: *https://en.wikipedia.org/wiki/Lossy_compression*

[6] Lossless formats: *https://en.wikipedia.org/wiki/Lossless_compression*

[7] DACs: *https://en.wikipedia.org/wiki/Digital-to-analog_converter*

[8] Compression to decoding speed: *https://xiph.org/flac/comparison.html*

[9] abcde: *http://www.andrews-corner.org/linux/abcde/abcde_lossless.html*

MakerSpace

Open hardware makes science education fun

Brown Dog Gadgets

Brown Dog Gadgets is making science education more accessible and affordable with open hardware. *By Bruce Byfield*

Free software and the maker movement have always been closely connected to education. In the last few years, open hardware has started to strengthen this connection. A case in point is Brown Dog Gadgets [1], a small company dedicated to making science accessible. The company's latest crowdfunded project is Crazy Circuits [2], kits of Lego-compatible circuit boards and other electronic components and a growing array of projects ranging from what founder Joshua Zimmerman describes as an "evil Cylon pumpkin" to basic Arduino programming (Figure 1).

A few yeas ago, Zimmerman was a middle school science teacher in Milwaukee, Minnesota. "My classroom had a really small budget – it was my salary. I was always looking for fun science projects for my students and myself. I would find projects on Instructables.com and YouTube and then implement them in my classroom and after-school science club. I was using the open source projects that were out there, and then I started posting them back on Instructables.com with better instructions and teaching points assigned to them."

After a while, other teachers started asking for parts and kits. "I started putting parts in bags, and that kind of snowballed on over a couple of years," says Zimmerman. "The tipping point was when I was working a 10-hour school day, [and] then an eight-hour evening shipping things and answering emails. I was getting maybe five hours' rest, and not doing well at either job. It came to a point where I had to choose one or the other, and I was upset at my current school at that point." Zimmerman chose his growing business, working out of his parents' basement, mailing kits and developing projects to use with the kits.

Zimmerman's first crowdfunding project in 2013 was for solar panels. The campaign reached its goals – but just in time for the market prices to drop as solar panels became widely available. Solar features are an important part of Brown Dog products, but its solar panels were unsellable and were finally thrown out in 2017.

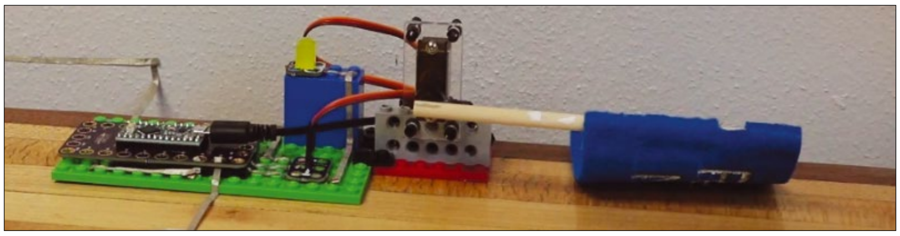


Figure 1: An Arduino programming project.

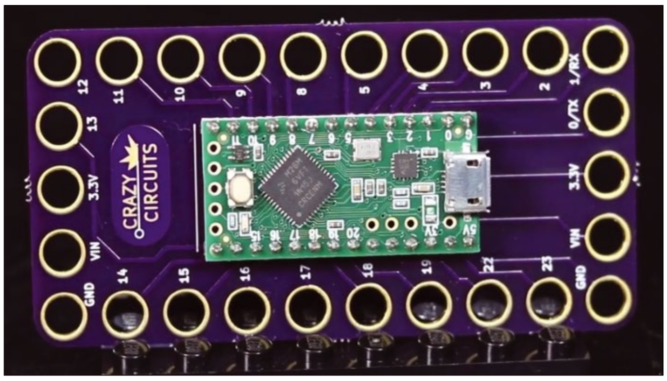


Figure 2: Crazy Circuits consist of color-coded, pre-soldered electronic parts.

Today, Brown Dog Gadgets still sells off-the-shelf components and will rent the use of its laser cutter and other machines for $40 to local customers – about half what is usually charged in Milwaukee. The company avoids doing design work, which Zimmerman describes as "the black hole of customer service. We really don't have the man hours to do that stuff, and most people don't want to pay for it." Increasingly, the company focuses on educational products: the best-selling Bristlebots, "small robots made by combining a toothbrush head and vibrating motor"; its arts and craft supplies, including paper circuits, conductive paint, and thread; and, increasingly, Crazy Circuits. "We try to be a budget brand," Zimmerman says. "We're cheaper, and we give you more for the same price."

Explaining Crazy Circuits, Zimmerman notes, "Most students and most teachers don't want to do a project on a breadboard with LEDs and jumper wires." By contrast, Crazy Circuits chips (Figure 2) do everything possible to simplify the use of electronics. Parts are pre-soldered, because "teachers either don't want to do soldering or don't have the resources in the classroom to adequately manage safety and liability issues." Besides, as Zimmerman points out, projects that require soldering are already widely available online. Parts are also color-coded and come with note cards showing the part and, on the back, an example of how the part might be wired in a simple circuit. A reference card, Zimmerman reasons, is easier to use than finding information online.

However, the main appeal of Crazy Circuits components is their compatibility with Legos (Figure 3). It took Brown Dog Gadget six months to drill the holes so that parts would fit on Lego blocks and to develop custom conductive tape that would stand up to wear and tear, but Zimmerman considers the effort worth his time. "Lego is ubiquitous," he says. "Everyone knows Lego. It's the largest toy company in the world, over the twice the size of its rivals, and well-branded with licensed products. Teachers can take advantage of those licensed projects by making things like a Star Wars X-Wing with wings that open and basic sound effects." Even Lego clones can be useful – for example, Zimmerman has started using steel blocks from a clone to designate push points.

Science Arts and Crafts

The Brown Dog Gadgets' home page promises "a passion for creating fun." Talking with Zimmerman, it soon becomes clear that the slogan is more than just words. Rather, it describes the company's entire approach to science education.

"From a teaching perspective, we try to mix new ideas with things people are already familiar with," he says, "making it 95 percent familiar. It's really science arts and crafts. I mean, every student by the second grade has learned at home or in the classroom to use scissors. It's what kids do. Well, adding conductive tape and LEDs to it isn't adding a whole lot of new things for them." In addition, familiarity is important for teachers, because the reality of public education is that "a lot of teachers are newbies to

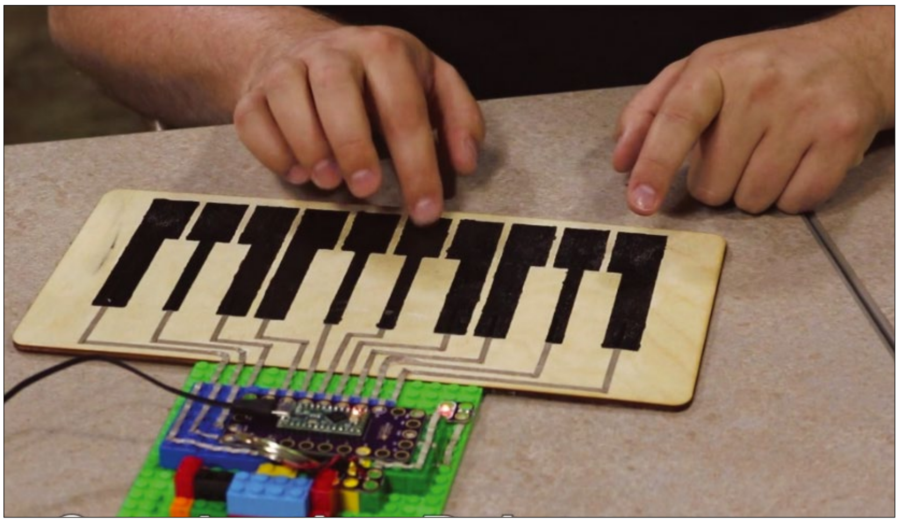


Figure 3: Crazy Circuits components are Lego-compatible.

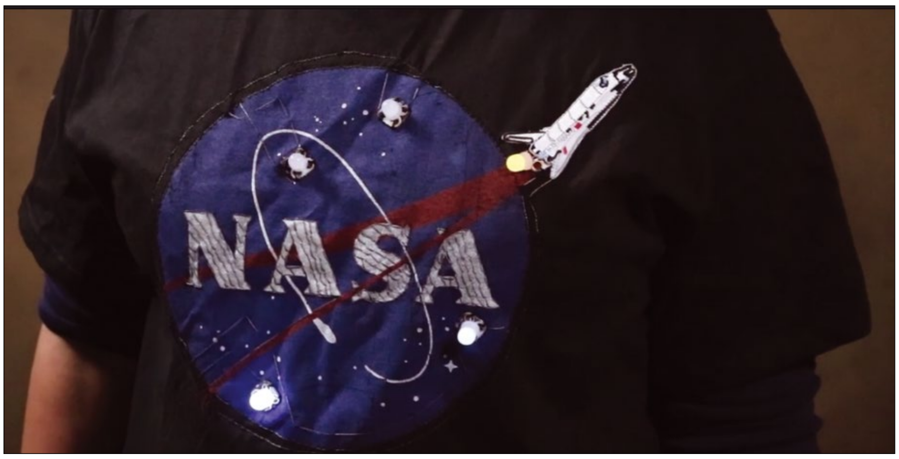


Figure 4: Electronic T-shirts are one of Brown Dog Gadget's more common projects.

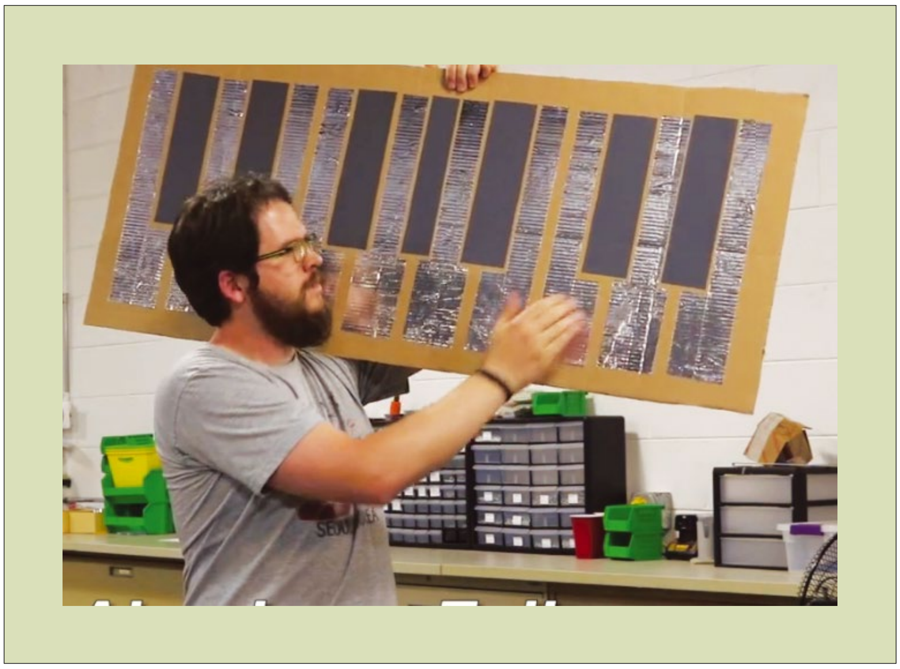


Figure 5: Using conductive tape to produce a playable keyboard.

their subjects, too, and if the teachers don't feel comfortable with something, they're not going to do it."

For example, a common science project is to create the game Simon [3], which requires players to echo increasingly more complex sound patterns. "It's just four buttons and four LEDs on an Arduino board," Zimmerman says, "but by putting it on Lego or on a T-shirt [Figure 4], you add that extra novel touch to it. It turns a breadboard project into a smart interactive kit. Or you can do the same project on a wall with conductive paint, so you're touching conductive paint points as opposed to push buttons, and suddenly the look and feel makes it engaging and interesting. From a teaching perspective, we try to mix new ideas with things that people are already comfortable with."

Zimmerman cites his mother as an example of the attitudes this approach is intended to overcome. "She's an amazing sewer and an amazing quilter and sews up a storm in her retirement, but she's like, 'I could never do your T-shirt projects,' and I say, 'Mom, I'm doing the most basic sewing known to man. I stink at sewing. All I'm doing is sewing on special buttons – that's all I'm doing. I know you can sew a button on, and I know you can use thread.' It's just adding that idea and presenting it in a way that is not frightening or threatening to somebody, especially an adult who is set in their ways."

Beyond this entry-level approach, Brown Dog Gadgets also offers projects for more advanced users. Often, he says, teachers get overambitious and set themselves impractical goals, such as trying to teach robot programming in a single semester. Instead, Zimmerman focuses on encouraging familiarity with programming. For example, instead of programming from scratch, Zimmerman includes projects that only require minor editing of existing code, such as creating a piano with conductive tape (Figure 5) and then changing the note played. "It's something new, but doesn't overwhelm them. Keep things simple; then let people have fun," he advises.

Educational Sharing

When Zimmerman is not filling orders, he spends his time talking at teaching conferences. Next year, he also hopes to start presenting at Lego events. His time is also spent writing an open source curriculum for electronics, working with other small resellers, and posting the results on GitHub [4]. The curriculum is designed to work with other off-the-shelf components, not just those manufactured by Brown Dog Gadgets (Figure 6). What matters is being accessible. "Teachers swap secrets," he says. "Reputation is huge. If something is working for them, and it's fun for them and fun for their classes, they tell everybody. That's the type of advertising that I could never afford. One teacher talking to other teachers on Facebook means far more than one Facebook post from me."

As a result of these efforts, a community is starting to form around people working with Brown Dog's components and projects. "I'm always amazed when people send me ideas, because half the time I'm like, 'That's a great idea. How can I do that?' Or, 'Ooh, that gives me another idea.' We're all about what people can do with our projects and our resources." ■■■

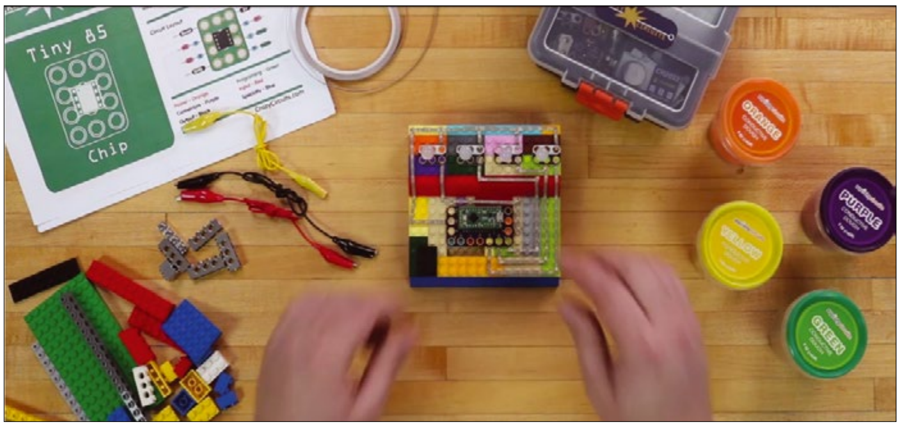


Figure 6: Conductive dough is another engaging way to bring electronics into the classroom.

Info

[1] Brown Dog Gadgets: *https://www.browndoggadgets.com*

[2] Crazy Circuits: *https://www.crowdsupply.com/brown-dog-gadgets/crazy-circuits*

[3] Simon: *https://en.wikipedia.org/wiki/Simon_(game)*

[4] Brown Dog Gadgets GitHub site: *https://github.com/BrownDogGadgets/CrazyCircuits*

MakerSpace

Raspberry Pi aids conservation work

Fox Hunt

As a countermeasure to predators of rare ground-breeding birds, live traps are monitored by a microcontroller and a Raspberry Pi. *By Bernhard Bablok and Lothar Hiller*

In a nature reserve in northern Germany, foxes and martens threaten rare ground-breeding birds. Although the park supervisor catches the predators with live traps, checking the traps is time consuming and expensive in this virtually inaccessible area. Between 40 and 50 traps have to be monitored, which means long and mostly unnecessary inspections.

The use of modern technology would minimize the considerable effort involved in regularly checking the traps by having a microcontroller and Raspberry Pi team up to monitor the traps electronically and send an automatic alarm when one of the traps is triggered. However, the path from this idea to its implementation took longer than originally thought.

One of the challenges was the lack of mobile Internet in the sparsely populated area, although text message services are available. A previous article [1], in which one of us described how to set up a 3G hotspot with a Rasp Pi and a UMTS [2] stick, was the basis for the collaboration on this project.

The Idea

The basic idea is simple: The PIC16F690 microcontroller with nanoWatt Technology uses a sensor, such as a ball switch or magnetic contact, to monitor the state of the trap and uses the PIC internal analog-to-digital converter (ADC) to monitor the battery voltage. When the trap is closed or the battery voltage is too low, the microcontroller stimulates a connected Rasp Pi and signals the status. The Rasp Pi then sends a text message to the supervisor over a connected UMTS stick, and the supervisor then transports the captured animal away or replaces the battery.

For the status of the trap, the Rasp Pi uses two GPIOs as input (one for the latch and one for the battery). At first glance, the script required seems very simple, because reading GPIOs and sending text messages can be implemented quickly, thanks to the large software base in Raspbian. However, the devil is in the details, because the solution has to be very robust: You would not want a captive animal to die in a trap because text messaging fails.

The Rasp Pi therefore not only has to read out the signals from the PIC but also report back the status. To do this, the setup requires two more GPIOs: one pin for the status and a second that indicates a shutdown to the PIC, which then switches off the power and saves the battery. In the event of a fault, the PIC stimulates the Pi after five minutes for

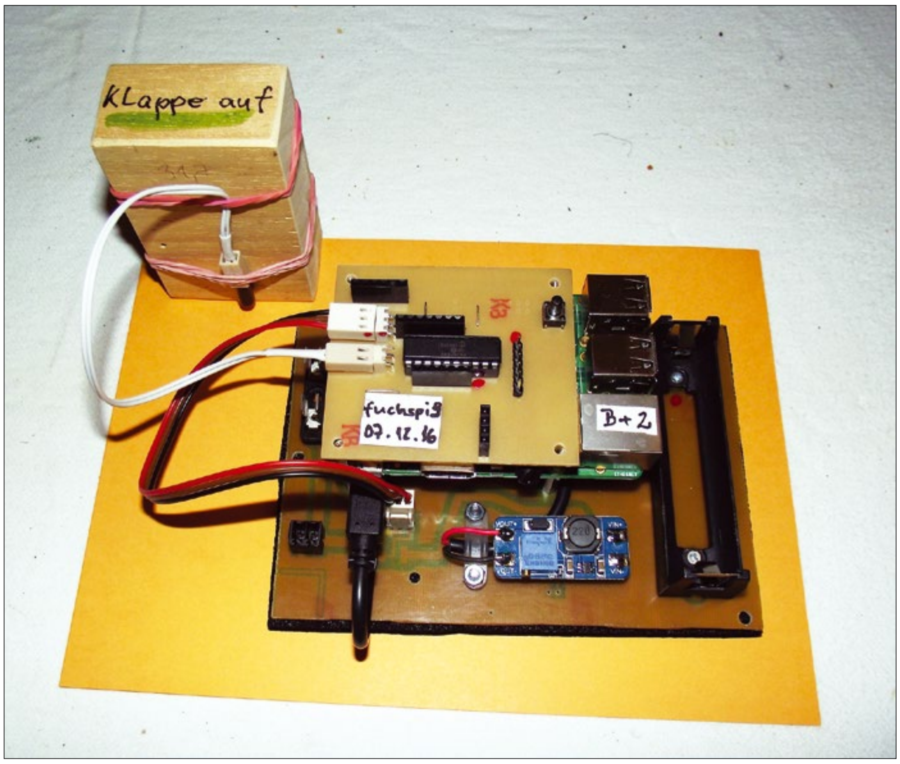


Figure 1: Prototype: The experimental setup with a connected Rasp Pi and microcontroller; Klappe auf = trap open.

another attempt. After a successful text message transmission, the PIC remains inactive until a reset, which the supervisor must handle.

An optional component is a real-time clock (RTC). Because the Rasp Pi does not make contact with a network, it cannot update its time. The RTC provides the right time for each boot, which means that all log messages have the correct time stamp. The text messages now also contain the transmission time in addition to the case number and the case or power status.

Microcontroller

The PIC16F690 is located on a carrier board that plugs directly into the Rasp Pi GPIO strip (Figure 1). The diagram and the parts list, as well as all programs, are on GitHub [3]. (English versions are available on the *Linux Pro Magazine* anonymous FTP site [4].)

The PIC16F690 by Microchip was chosen because it works with particularly economical nanoWatt Technology. In sleep mode, a watchdog timer monitors the PIC so that the power consumption from the battery is only about 1-1.5µA. Among other things, the controller has 18 I/O pins, an internal oscillator, a 10-bit ADC with 12 channels, flash memory for 4096 words, 256-byte SRAM, and interrupt on change (IOC) at ports A and B. Transfer of data to a 4x16 LCD uses software I2C and the PCF8574 I/O expander. This solution allows transmission at up to 100kHz; the PIC can then be operated at a maximum of 4MHz. This data transfer variant saves I/O pins.

The PIC is set to sleep mode for about five minutes after configuration. It then wakes up and controls the trapdoor sensor and battery voltage. If everything is OK, it will switch to sleep mode again. In this case, the Rasp Pi is left out. If no animal is trapped, a battery charge is sufficient for about a half a year in this setup.

Only when an event occurs (trap closes, battery dies) does the Rasp Pi boot and send a text message. The Pi reports a transmission failure to the PIC, which "remembers" it. After another five minutes, the Rasp Pi starts resending the text and does so repeatedly until the transfer is successful. The system then returns to the idle phase.

The chip is programmed with assembler in a special single-step mode. The information is forwarded by software I2C and pluggable auxiliary boards and is visualized on an LCD with four lines of 16 characters each (Figure 2).

The PIC is connected to the Rasp Pi through four GPIOs (Figure 3). GPIOs 17 and 27 inform the Rasp Pi of the status of a trap, which in turn sets GPIO 22 to HIGH during program startup and to LOW in the event of an error. The shutdown signal comes through GPIO 10. After the value changes to HIGH, the PIC gives the Rasp Pi another minute to shut down and then turns off the power (Figure 4). If the Rasp Pi takes too long (e.g., because a text message is pending), the PIC still strictly regulates the current. The Pi loses no data, apart from some log messages.

However, the PIC alone is not enough for the circuit. Another critical component is the boost converter (Figure 5),

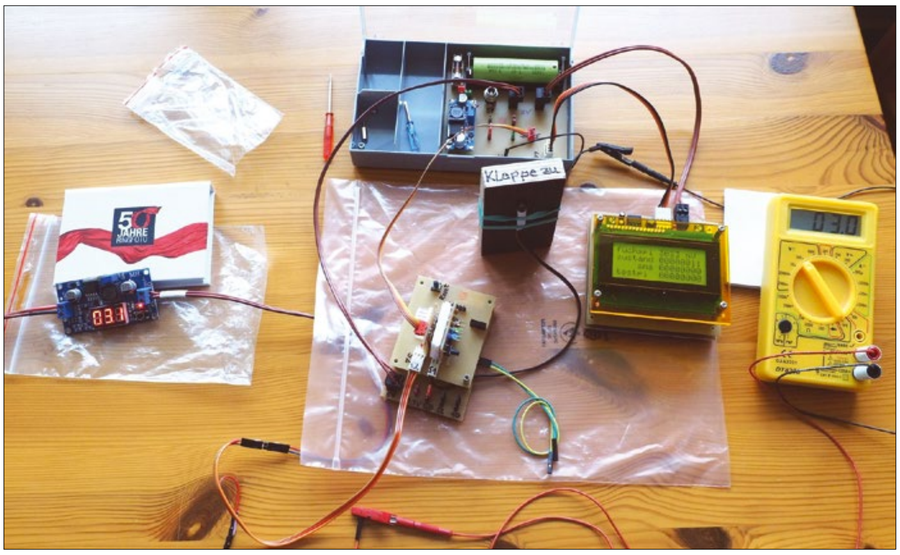
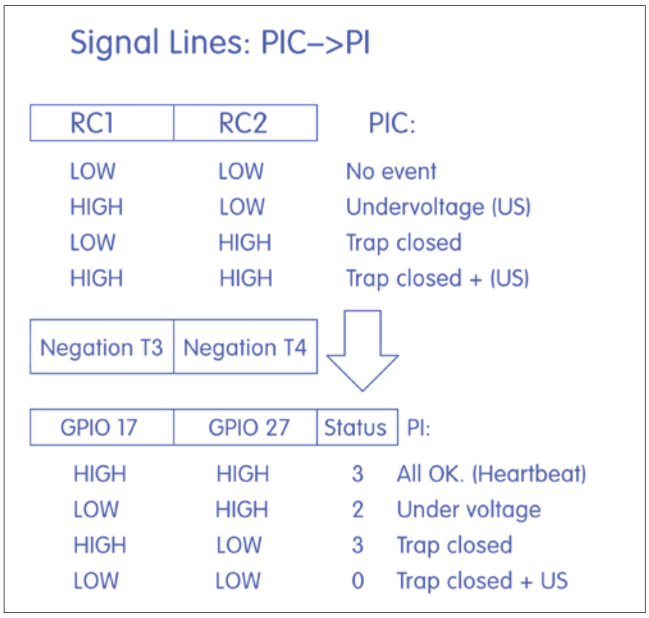


Figure 2: A segment display allows debugging of the setup; Klappe zu = trap closed.

which uses the 3.0-4.2V battery voltage to generate the 5V required by the Rasp Pi. First, the XL6009 switching regulator module (up to 4A) was used. However, because this was no longer reliable enough in the lower voltage range of 3-3.5V, the ML8205 (2A) module was given preference, which runs smoothly down to 3V. For test purposes, a switch mode power supply connected with an adjustable step-up converter replaces the lithium polymer (LiPo) battery, which means it is possible to stimulate undervoltage without problem.



Signal Lines: PIC–>PI

RC1	RC2	PIC:
LOW	LOW	No event
HIGH	LOW	Undervoltage (US)
LOW	HIGH	Trap closed
HIGH	HIGH	Trap closed + (US)

Negation T3	Negation T4

GPIO 17	GPIO 27	Status	PI:
HIGH	HIGH	3	All OK. (Heartbeat)
LOW	HIGH	2	Under voltage
HIGH	LOW	3	Trap closed
LOW	LOW	0	Trap closed + US

Figure 3: The signal lines from the PIC to the Rasp Pi; US = UMTS stick.

The pull-up/pull-down resistors integrated in Rasp Pi seem to be some of the least known components of the small computer and proved to be a particular challenge. Although the GPIOs are well documented, with many tutorials about connecting with external pull-up or pull-down resistors, hardly any articles about the internal resistors can be found.

These pull-ups can be configured using the WiringPi library. Once set up, Rasp Pi saves the status, but it can no longer be read out. The data is probably stored in an EEPROM-like cell that cannot cope with arbitrary write operations. However, none of this is documented and is one downside of the Rasp Pi hardware not being completely open.

Rasp Pi Tasks

When the PIC switches on power, the Rasp Pi boots and works directly on its tasks: reading GPIOs, searching for modem, setting PIN code, sending text message, informing PIC, and shutting down again. Ideally, this should take no more than two minutes. However, the implementation has to intercept various error situations.

The `/etc/rc.local` file starts the `/usr/local/sbin/fuchsfalle.sh` processing script at the end of the boot process. It then monitors in a loop to see whether the script stops again. After a configurable amount of time, it terminates it regardless of its status, informs the PIC about the failure, and shuts down the system. The Rasp Pi tries to prevent the hard shutdown caused by the PIC.

The search for the modem, the setting of the PIN code, and the sending of text messages all make several attempts in a loop, which proves to be a useful precautionary measure in practice. Sometimes it takes a while for the UMTS stick modem to become available. From time to time, setting the PIN or sending text messages also requires several attempts.

The Rasp Pi behavior is controlled by the central configuration file `/boot/fuchsfalle.cfg`. The storage location was chosen because it is located in the first FAT-formatted partition and can therefore also be accessed from Windows. However, on the Windows side, you will need an editor that also saves the file in UTF-8 format, which is common on Linux. Notepad++ does a good job of this.

Special Features

The sysfs interface under `/sys/class/gpio` was originally used for the configuration, as well as for reading and writing the GPIOs. However, when the problems with the internal pull-ups described above arose, we switched to the *wiringPi* package, which provides a command-line tool and thus performs all tasks without problem from scripts. The following commands are used to configure the pull-ups:

```
$ sudo gpio -g mode 17 up
$ sudo gpio -g mode 27 up
```

In addition to `up` are the `down` and `tri` switches. The latter probably

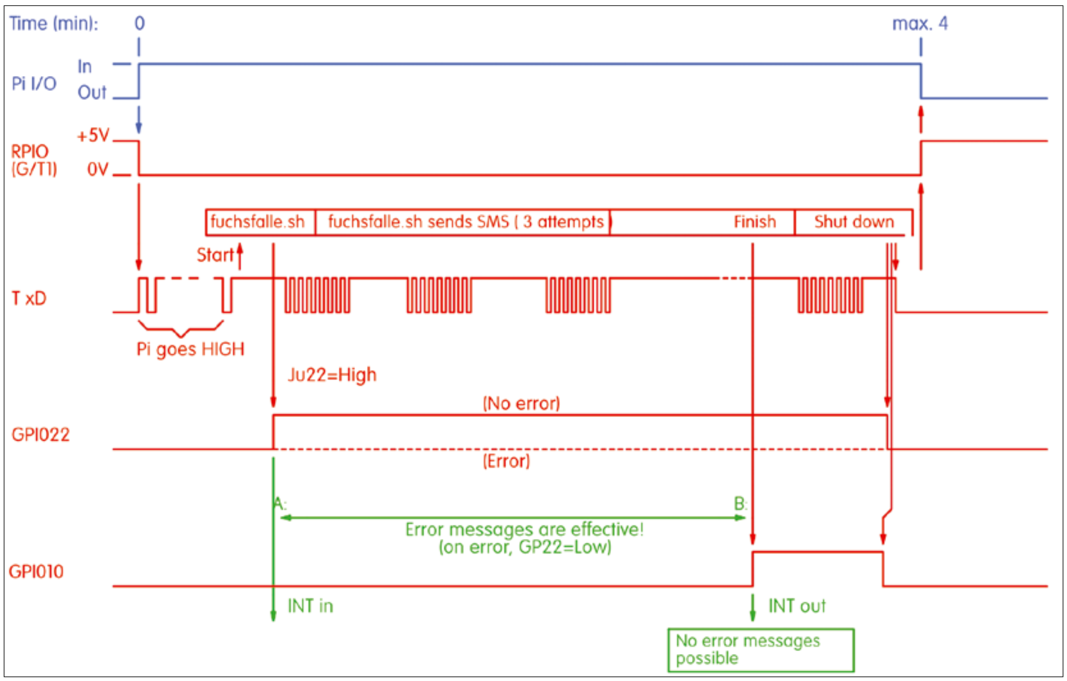


Figure 4: The intended schematic for the fox trap.

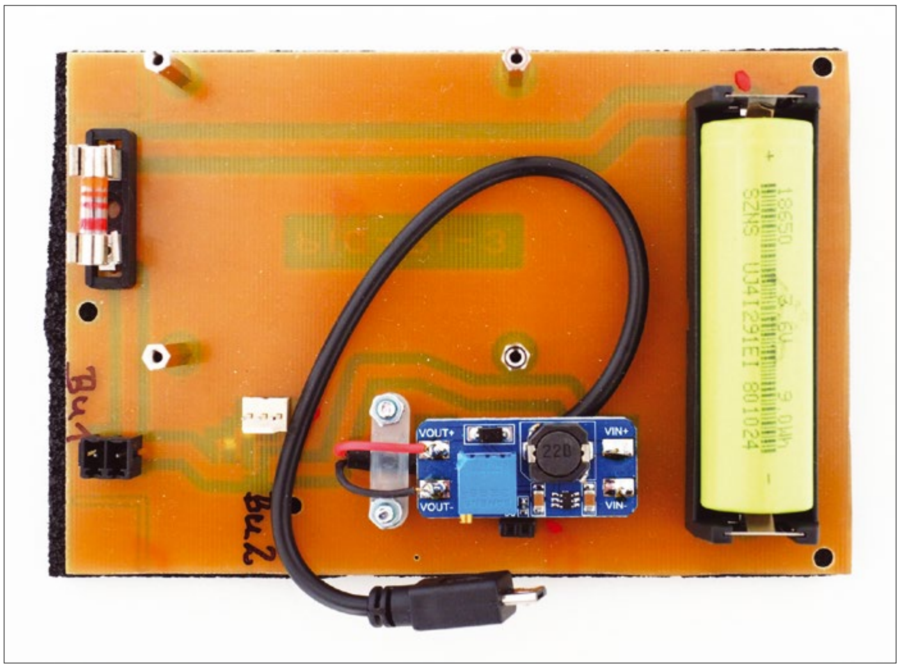


Figure 5: The carrier board with battery and boost converter.

stands for "trigger" and switches off the internal resistors. However, it is not possible to read the status with software.

In addition to GPIOs, other challenges need to be solved by the software. Even if it only runs for a few minutes, the system needs to save as much energy as possible. The Rasp Pi Zero is considered unbeatable in this respect, but because UMTS sticks draw a large amount of power, it is not guaranteed that this arrangement will work with all sticks. Turning off the screen and network helps all Rasp Pis conserve energy. An interface-free system on a chip (SoC) also poses no problem in production because communication with the PIC takes place through GPIOs.

The various power-saving techniques were implemented with a separate system service, reconfigured in the central configuration file, which advantageously allows the individual techniques to be switched on and off as required.

Consistent logging is another feature. To ensure that as little information as possible is lost in the event of a power outage, a sync function ensures a secure write to the SD card after every written record and also reduces the possibility of filesystem corruption.

Installation

A Raspbian Lite installation with a number of additional packages, like *wiringPi* for the GPIOs and *gammu* for sending text messages, is sufficient, but because

the installation involves a number of manual steps, the Raspbian Netinstaller [5] comes into play by automating the entire installation process and ensuring uniform, reproducible, and consistent systems.

The project files are located in the appropriate folders, and the configuration files for the installer are also updated directly in the project. A script downloads the Netinstaller and integrates the files from the project into its own image. After booting, the system then retrieves all necessary files and subsequently configures the system. After installing, all that remains is configuring for the specific trap.

Conclusions

At present, the system is sufficiently advanced to start beta testing, with plans for improvements already in place. For example, the Rasp Pi could send a text message at regular intervals, even without triggering it as a heartbeat message, so that the supervisors can have an overview of all traps. However, this would mean increased capacity requirements with regard to the LiPo battery.

Since the beginning of 2016, small SIM modules like the SIM800L [6] have been around that can be connected to the Rasp Pi through the GPIO. These modules offer an interesting alternative to the UMTS sticks currently used in the project in terms of price and technology. Problems with unrecognized sticks under Linux are then a thing of the past.

The mode switch from a CD-ROM to a modem device, which is necessary under Linux, turned out to be particularly problematic with these sticks, some of which only work under Wheezy and jessie-backports, but not under Jessie. Others work with Jessie without problem, but when switching to the USB modeswitch backport, the system suddenly no longer recognizes it. The software installation then depends on an external component.

Another advantage is that the SIM800 modules draw power directly from the LiPo battery, which means the module even works on first-generation Rasp Pis or the Pi Zero without problem. Additionally, the entire USB complex of the Rasp Pi 2/3 models can be switched off, which saves even more energy.

Regardless of the challenges on the hardware and software sides, many administrative and logistical problems have yet to be solved. For example, you would have to produce the printed circuit board (PCB) in small batches. The many required SIM cards also present a problem; because, ideally, the Rasp Pi only sends a handful of SMS messages per year, prepaid cards are best suited for this purpose. However, managing 40 to 50 contracts is a challenge and, in the worst-case scenario, could lead to false positives.

Finally, financing all the traps is still a problem. Purely from the electronics side, each costs about EUR100, if you produce the PCBs yourself. Another matter of concern is the cost of construction for the concrete pipe traps (Figure 6).

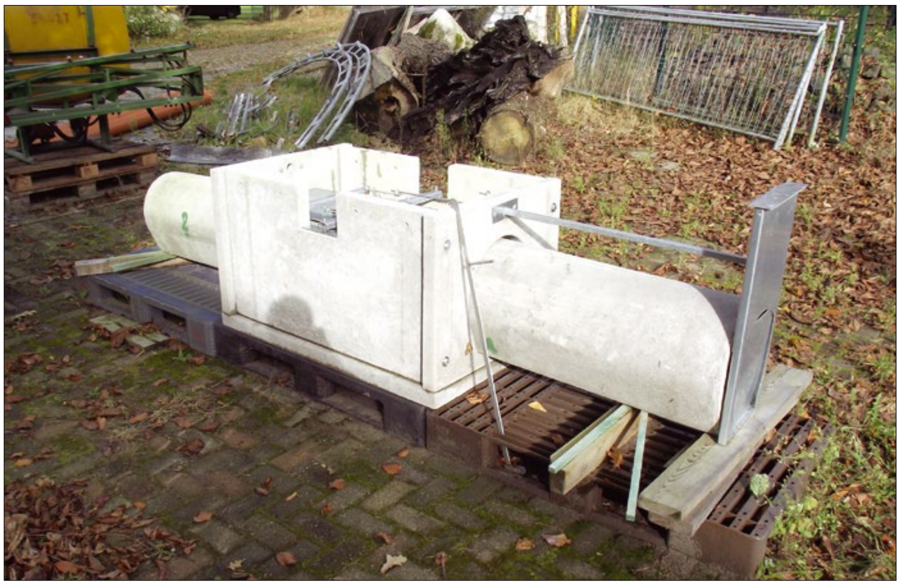


Figure 6: The robust concrete pipe traps will provide a long service life.

The circuit and schematic developed for the fox trap easily can be transferred to other projects that work autonomously and only occasionally transfer data. Because a text message has up to 160 characters, it is suitable for more than just a simple status report, and more information could be distributed across multiple text messages.

If GPRS at least is available in the intended area of application, an Internet connection with the UMTS stick is more suitable for the transfer of large amounts of data. Setting up the stick for this purpose is described in a previous RPG article [7]. ■■■

Info

[1] "Using SMS" by Bernhard Bablok, *Raspberry Pi Geek*, issue 21, 2017, pg. 82, *http://www.raspberry-pi-geek.com/Archive/2017/21/Sending-and-receiving-an-SMS-with-the-Raspberry-Pi*

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

[3] Fuchsfalle [Fox trap] on GitHub: *https://github.com/bablokb/fuchsfalle*

[4] English instructions: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/207/*

[5] Minimal Raspbian Netinstaller for Rasp Pi: *https://github.com/FooDeas/raspberrypi-ua-netinst*

[6] SIM800L: *https://www.amazon.com/SIM800L-Quad-band-Breakout-Interface-Raspberry/dp/B06Y6Q7MFB*

[7] "Rasp Pi 3G Hotspot" by Bernhard Bablok, *Raspberry Pi Geek*, issue 15, 2016, pg. 40, *http://www.raspberry-pi-geek.com/Archive/2016/15/Creating-a-3G-hotspot-with-the-Raspberry-Pi*

The Authors

Bernhard Bablok (*mail@bablokb.de*) works at Allianz Services SE as a SAP HR developer. When he's not listening to music, cycling, or walking, he enjoys Linux, programming, and SBCs.

Lothar Hiller is a retired intelligence operations engineer. He is a passionate electronics enthusiast with a soldering iron that is almost always hot. Hiller programs microcomputers like PICs in assembler, and since 2015, he has been working with Raspberry Pis in the home applications field.

MakerSpace

Integrate keypads and gamepads into your next project

In Control

The Python evdev library offers a simple way to connect input devices, even if you don't know the key bindings. By Pete Metcalfe

Keypads and gamepads can be good solutions for projects that have requirements for simplicity, small form factor, or more rugged operation. Companies like Sony (PS2/PS3) and Logitech, and even makers of generic clones, offer a number of keypads and gamepads for USB wired or wireless products (Figure 1).

Although the Python Pygame library offers a generic keyboard and joystick interface, I found the *python-evdev* library to be a little simpler and more straightforward.

python-evdev

The *python-evdev* library [1] provides bindings to the generic input event interface in Linux. The *evdev* interface passes events generated in the kernel, typically located in `/dev/input/`.

To install *python-evdev* on a Debian-compatible operating system, enter:

```
sudo apt-get install python-evdev
```

Not all keypad and gamepad buttons are mapped consistently, so a simple test program is needed to find the keycodes (Listing 1) [2]. For most small systems,

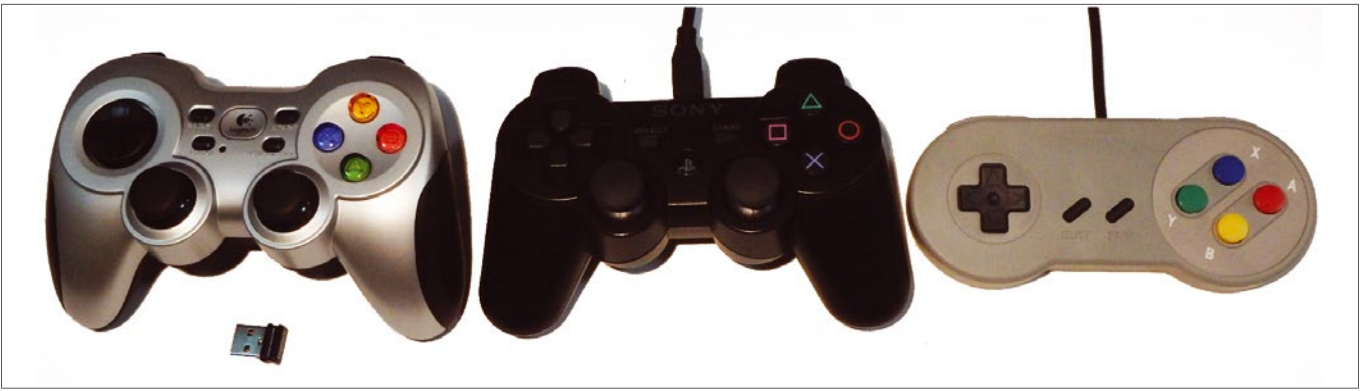


Figure 1: Wired and wireless keypads and gamepads.

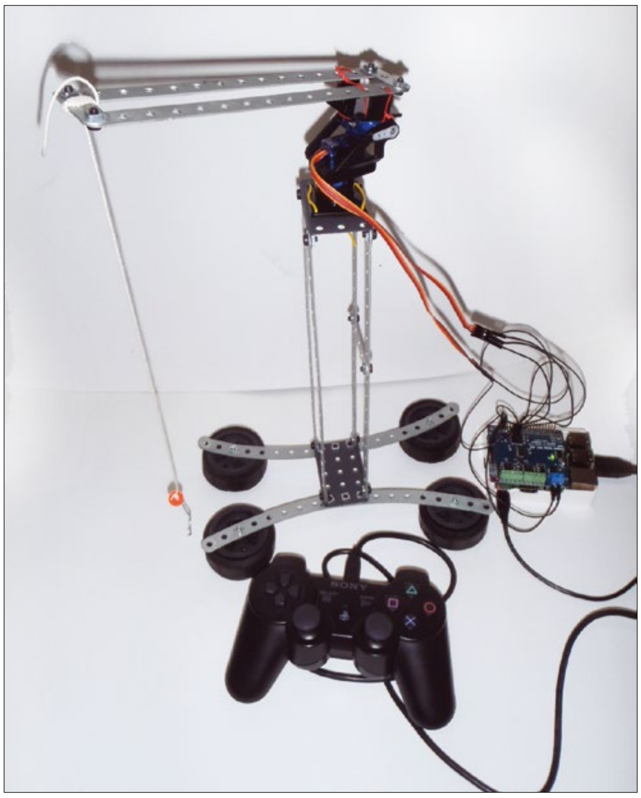


Figure 2: A Raspberry Pi crane with a gamepad controller.

Listing 1: keys_evdev.py

```
from evdev import InputDevice, categorize, ecodes
gamepad = InputDevice('/dev/input/event0')

for event in gamepad.read_loop():
    if event.type == ecodes.EV_KEY and event.value == 1:
        try:
            keyevent = categorize(event)
            print (keyevent) # comment to remove key messages

        except:
            print ("Problem key pressed...")
```

Listing 2: Test Program Output

```
key event at 1488557515.714249, 295 (BTN_BASE2), down
key event at 1488557517.491448, 294 (BTN_BASE), down
key event at 1488557519.594712, 293 (BTN_PINKIE), down
```

the controller's USB connection will be on `/dev/input/event0`; on larger systems, `event1` or `event2` might be the input device. When a key is pressed, the test program outputs the time, keycode, and an alias for the key binding (Listing 2).

At a minimum, most keypads or gamepads will have the buttons: A, B, X, Y, Start, Select, Left Shoulder, and Right Shoulder. The code in Listing 3 captures these buttons.

Summary

With the sample code provided here, you have a starting point for your next kiosk, robotics, or game project; for example, my daughter and I used the *evdev* interface with a Raspberry Pi to control a two-axis crane (Figure 2). ■■■

Author

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

Info

[1] python-evdev: *http://python-evdev.readthedocs.io/en/latest/#*

[2] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/207*

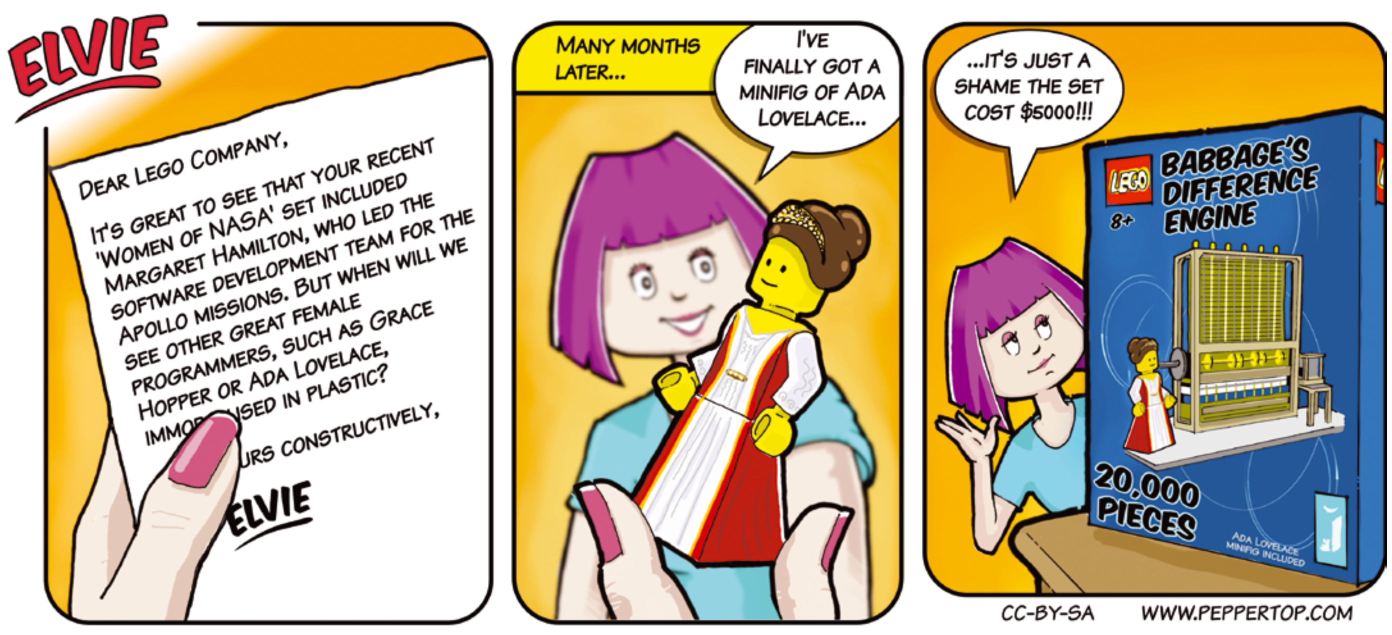
Listing 3: gamepad_evdev.py

```
from evdev import InputDevice, categorize, ecodes
gamepad = InputDevice('/dev/input/event0')

for event in gamepad.read_loop():
    if event.type == ecodes.EV_KEY and event.value == 1:
        try:
            keyevent = categorize(event)
            if keyevent.keycode == "BTN_THUMB":
                print ("Right - 'A' button")
            if keyevent.keycode == "BTN_THUMB2":
                print ("Bottom - 'B' button")
            if 'BTN_TRIGGER' in keyevent.keycode:
              # ['BTN_JOYSTICK', 'BTN_TRIGGER']
                print ("Top - 'X' button")
            if keyevent.keycode == "BTN_TOP":
                print ("Left - 'Y' button")
            if keyevent.keycode == "BTN_TOP2":
                print ("Left shoulder button")
            if keyevent.keycode == "BTN_PINKIE":
                print ("Right shoulder button")
            if keyevent.keycode == "BTN_BASE3":
                print ("SELECT button")
            if keyevent.keycode == "BTN_BASE4":
                print ("START button")

            print (keyevent) # comment to remove key messages

        except:
            print ("Problem key pressed...")
```

If you work at the command line, you are never far from the trusty `cd` (change directory) command. But seriously, is `cd` as good as it gets? The ever-inventive Linux community is always looking for something better – even for something as simple as directory navigation. Meet bd, autojump, and Fasd, alternative navigation tools for the command line. We also explore SystemRescueCd, a specialty Linux designed as a system recovery tool, and Paul Brown continues studying the FFmpeg multimedia framework he began last month, showing how you can use FFmpeg to generate video streams on the fly.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE


ELVIE
DEAR LEGO COMPANY,
IT'S GREAT TO SEE THAT YOUR RECENT 'WOMEN OF NASA' SET INCLUDED MARGARET HAMILTON, WHO LED THE SOFTWARE DEVELOPMENT TEAM FOR THE APOLLO MISSIONS. BUT WHEN WILL WE SEE OTHER GREAT FEMALE PROGRAMMERS, SUCH AS GRACE HOPPER OR ADA LOVELACE, IMMOR ISED IN PLASTIC?
URS CONSTRUCTIVELY,
ELVIE
MANY MONTHS LATER...
I'VE FINALLY GOT A MINIFIG OF ADA LOVELACE...
...IT'S JUST A SHAME THE SET COST $5000!!!
LEGO
BABBAGE'S DIFFERENCE ENGINE
8+
20,000 PIECES
ADA LOVELACE MINIFIG INCLUDED

CC-BY-SA WWW.PEPPERTOP.COM

NEWSANALYSIS

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

Opinion

Unknown Others

Open source is for you, yes, but it's also for unknown others. BY SIMON PHIPPS

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

Being close to an open source project, it's easy to imagine that everyone sees the project the way you and your fellow community members do. This especially applies to the corporate sponsors of a single-company project; anticipating use by competitors, they often want to apply controls to who can use the code.

A core objective of software freedom is to ensure that the code can be used not only by your collaborators, but by unknown others with undisclosed goals. All OSI-approved licenses ensure everyone is permitted to use software for any purpose without further permission, delivering this core objective.

Random code liberation leading to unexpected application (aka "innovation") has always been and will remain a hallmark of open source. Borrowing portions of great code – from elegantly executed algorithms to useful libraries to entire components – is an intended mode of exercise for software freedom and not an artifact. Leaving it available is essential.

The same provisions that allow code reuse also enable the crucial pressure release valve of open source: the fork. The ability to take the code and do something the original author or the current community don't want is an essential freedom, not an unwanted side effect. Indeed, it is the origin of many of the most significant moments in open source.

It was a fork that rescued OpenOffice.org from corporate neglect, giving us LibreOffice. A fork allowed ForgeRock to rescue Sun's identity management software from abandonment, thus saving huge investments in its deployment and creating a highly valued "unicorn" startup in the process. The MariaDB fork is keeping the MySQL project focused on community. Even the Firefox browser was a kind of fork from Mozilla, albeit a strategic one.

A core objective of software freedom is to ensure that the code can be used not only by your collaborators, but by unknown others with undisclosed goals.

Making open source code freely available to unknown others is thus axial and not tangential to open source. That's why I get extremely concerned by anything that wants to be seen as "open source" but still tries to lock out the outsiders, the rebels, and the aliens. Attempts to do this range from the crude, like using a "time-locked" license that only becomes open source after a significant delay for "monetization," to more subtle approaches, like requiring an account to access the source repository and then only allowing paying customers to have an account.

The code may be under an open source license, but software freedom is not present if accessing or using it requires being or knowing an insider. None of this is theoretical; indeed, ForgeRock and MariaDB are themselves playing these games despite their origin story being rooted in software freedom.

So remain skeptical when software freedom is abridged or diminished in pursuit of a business model or "safety." Whatever that's called, it's not open source. ■■■

MADDOG'S DOGHOUSE

Your own personal cloud

BY JON "MADDOG" HALL

Jon "maddog" Hall is an author, educator, computer scientist, and free software pioneer who has been a passionate advocate for Linux since 1994 when he first met Linus Torvalds and facilitated the port of Linux to a 64-bit system. He serves as president of Linux International®.

Two months ago, I introduced in this column Subutai, the open source peer-to-peer cloud software created by OptDyn. Since that time, people have asked how they might use Subutai in their work and at home.

Many people store their data in commercial clouds, like Amazon, Google, or Microsoft Azure. While this can give them access to their data from many different machines, both inside and outside their facilities (whether it be home or work), it also means that their communications costs can go up if they are accessing their data a lot from outside of the cloud infrastructure.

Imagine if you are working at home storing your personal music, pictures, and other data in one of these commercial clouds. You pay for the Internet bandwidth to transfer the data to the cloud, and you pay for the bandwidth again to access it in the future. To fix this problem you might, as many people do, set up a network-attached storage (NAS) system at home or at work and access your data directly when you are in the local environment. Unfortunately, this also means that your data may not be easily available when you are not in the same network as the NAS system. By setting up your own cloud with Subutai and using it in conjunction with your NAS system, you can access your data from the NAS without paying your Internet provider for the bandwidth – and typically at much faster speeds than the Internet provider can sell you. When you go outside of the local network, the Subutai peer-to-peer cloud software still allows you to access your data transparently.

One of the other ways people use commercial clouds is to back up their data. For small amounts of data, this service may be free, but often those data costs quickly increase, and you end up paying a monthly charge.

If both you and a trusted friend or relative have a NAS system at home, you could set up a relationship in which you do backups to their NAS system and they do backups to yours. In case of disaster (fire, theft, flood), you can take a new disk to their home, rapidly copy your data to that disk, and restore all of your files to your new NAS system. If time is not an issue, you could do this restoration over the Internet without even going to their home.

Another usage would be to create a Live flash memory stick of your favorite GNU/Linux distribution with persistent data and Subutai incorporated. Having booted your Live distribution and set up Subutai, you now have access to your cloud environment. When you shut down the system, all of your data is kept in your private Subutai cloud, and the persistent storage keeps your Subutai settings, ready for the next time. For travelers like me, it might mean that all I have to bring with me is that flash stick, and I can use almost any laptop or desktop in the world for doing my work. My data is safe in my private cloud and away from the prying eyes of customs agents.

For telecommunications companies (Telcos) that felt "bypassed" by the giant cloud companies, Subutai offers the chance to use open source cloud software, giving Telco customers the opportunity to utilize spare resources. The same is true for web-hosting companies or other Internet service providers (ISPs).

For companies, including governments and organizations such as hospitals, Subutai allows consolidation of individual internal resources, which cuts down on their external telecommunication and cloud facility costs. They may not stop using commercial clouds altogether, but they may use them a lot less. For more "sensitive" information, such as customer banking and health information, they may instead use a Subutai private cloud.

The first time I went to Brazil, I visited the University of Sao Paulo (USP) and saw the first Beowulf supercomputer (a high-performance parallel computing cluster) in "real life." The main use of Beowulf was for diagnosing mammograms to detect breast cancer. The program running on a SPARC workstation would take 20 hours, but on the Beowulf system, it would take only 10 minutes, which meant that the woman would probably still be at the hospital when the results were known, and either further testing or an operation could be scheduled.

The university did not intend for each hospital to buy and maintain a Beowulf system, but to spread the load out over all the laptops, desktops, and small servers in the hospital to get the same effect. Subutai would make this task trivial, and if even one woman's life was saved by timely information, it would be worthwhile. ■■■

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

Audio editor

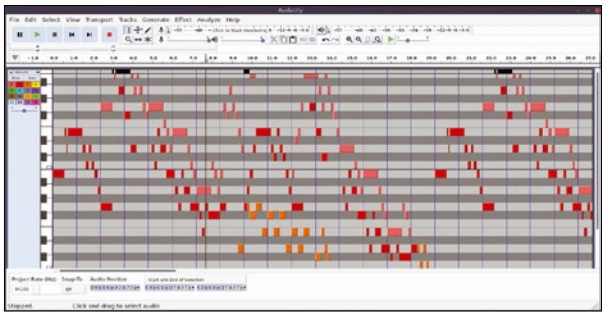
Audacity 2.2.0

Audacity is the audio editing equivalent of Gimp. It's an open source cornerstone in both its breadth of functionality and as a flag bearer for the capabilities of open source. Like Gimp, Audacity shows that Free Software is a viable alternative to costly proprietary software, in that it can be used as a drop-in replacement for costly alternatives. Before Audacity, there wasn't even a low-cost alternative. If you wanted to edit and process audio with any precision, then musicians, podcasters, audio engineers, and desktop audio tinkerers had to either spend money or use much more primitive tools. Audacity has changed all that, and you can now find it on the Mac OS and Windows desktops of successful producers, alongside those of us still producing audio and playing with analog synthesizers on Linux.

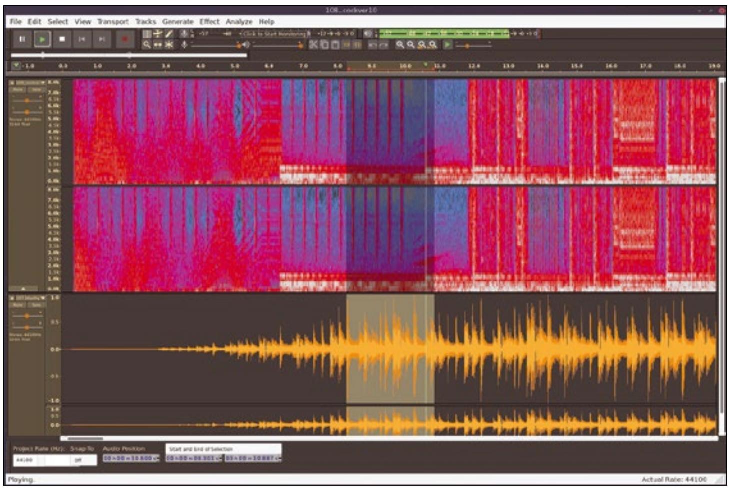
But Audacity isn't perfect. In particular, its user interface has more in common with the late 1990s than the 21st century. Its audio processing also suffers from being from the same era, allowing you to apply an audio effect only after selecting a range, listening to a preview, and clicking *Apply*, rather than letting you create your own virtual effects rack that you can change in real time. Outside of these shortcomings, Audacity allows you to edit audio right down to the individual samples, from simple mono recordings to multitrack audio on different lanes recorded at high bit depth and high frequencies.

Thankfully, this major update attempts to redress some of Audacity's shortcomings, in particular, by incorporating four user interface themes designed for the Dark Audacity project, alongside links to the help documentation embedded within many of the application's dialogs. The help links go hand-in-hand with a fully revised manual, which will help many new users understand the sometimes dark art (and relatively inaccessible world) of audio editing. The new themes work well in the dark environments favored by many recent digital audio workstations and look much fresher than the bright grey of the original. Another major addition is MIDI playback and editing. Although MIDI files are used to produce sound, they're completely different from audio files. They're analogous to vector images versus pixel data, containing note, pitch, and instrument data rather than the raw samples.

To hear MIDI, the data needs to be sent to a MIDI-interpreting synth (e.g., FluidSynth. Audacity then lets you edit MIDI files just like raw audio files, selecting ranges and loops, and editing to time points and signatures – even alongside audio if you open another file. This is a brilliant and rather cutting-edge feature, and it would be fantastic if these kinds of developments signaled a new direction for Audacity – one that includes parallel real-time audio effects previews and a user interface built around a more modern toolkit. But as with everything to do with Audacity, these are small criticisms when compared with the unrivaled editing power it offers in the realm of audio editing.



MIDI data files can now be edited just like audio files, which is a unique and useful feature if you work with MIDI sequencers.



This release finally makes some user interface concessions, with the inclusion of a new set of dark and high-contrast themes.

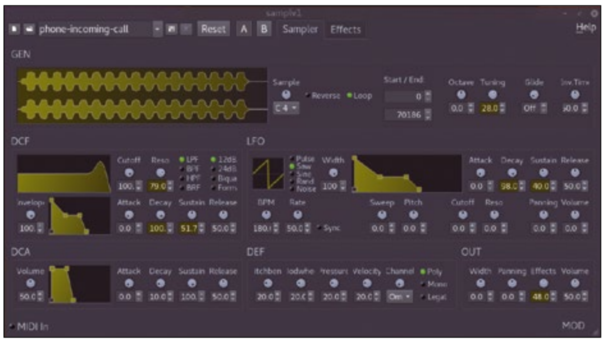
Project Website
http://www.audacityteam.org/

Audio sampler

samplv1

Samplv1 is just one of four audio synths released together as the Vee One Suite. This is a great collection of sound generators built around the same Qt toolkit and design elements, and they each share many of the same processors, such as the filters and envelopes, used to shape the sound. They can each be used standalone or as a plugin, and they share the same preset management and color scheme. But they also have their own unique sound. That's because they each implement different sound engines. Samplv1 is an old-school sampler that lets you take your own source material, usually a short recording of an instrument or percussive strike, and process it in such a way that the output sounds excellent across many octaves and playing styles. You could take a tinny piano sample, for example, and use samplv1 to convert it into your own mid-90s house piano for your latest pop hit. Other instruments in the suite include a drum kit sampler, a classic subtractive synthesizer, and an additive synthesizer (covered below).

A sampler is an audio production mainstay, whether for faking illness at school, providing beats, playing back samples of an expensive synth, or sampling your pots and pans. And while there are few samplers for Linux, very few attempt to combine a sampled audio source with the general synthesizer elements that augmented many samplers in the 1990s and 2000s. Those that do, such as the filter and the low-frequency oscillation (LFO), add an analog character to the source material. It's these elements that make samplv1 sound so good. Even poorly recorded samples or the content of your `/usr/share/sounds` directory can be used as creative sources, alongside drum loops and the usual fare, all of which make samplv1 an essential add-on to any Linux music workstation.


Samplv1 lets you play back your samples in stereo, processed through some excellent sounding modulators and filters.

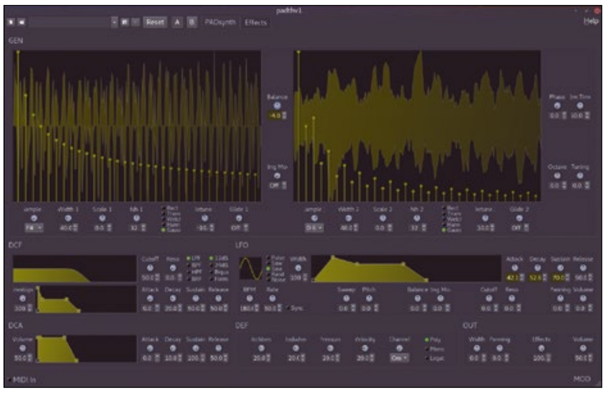
Project Website
https://samplv1.sourceforge.io/

Additive synth

padthv1

The prosaically named padthv1 is an additive audio synthesizer. Additive in this sense means the audio complexity is engineered by combining sine waves to modulate samples, creating a complex series of harmonics and dynamics that can be played at various musical pitches. It's perfect for chime, choral, and pad sounds, common in the mid-1980s, and is in many ways the opposite of the subtractive analog synthesis found in the majority of synths. At the center of this synth's sound is the PADsynth algorithm, which might already be familiar to you. Developed by Paul Nasca, the same algorithm is used in the brilliant ZynAddSubFX to generate similar sounds. Padthv1, however, has a much more advanced Qt-based user interface and runs either as a standalone JACK application or as an LV2 plugin suitable for embedding within your favorite audio software.

Alongside a filter, the low-frequency oscillator, the envelope generator, and the effects that you'd find on almost any synth are in the two GEN sections at the top of the window. These house the additive section that generates the sound piped into the rest of the synth. The controls are identical for them both, letting you select between the waveform you want to use for the basis of the sound, the amplitude for each harmonic, and various scale and width settings that change the spread and tuning of the harmonics. It sounds complicated, but it actually "sounds" really good, even when simply changing values. Additive sound generation like this also sounds completely different from any other sound source. A balance knob adjusts the mix between the two GEN sections, and a ring mod effect can be added to create a more metallic and robotic sound, helping padthv1 hold a unique position in the Linux synth pantheon.


As with all Vee One Suite processors, the level of control and visual feedback is exceptional, especially for an open source synth.

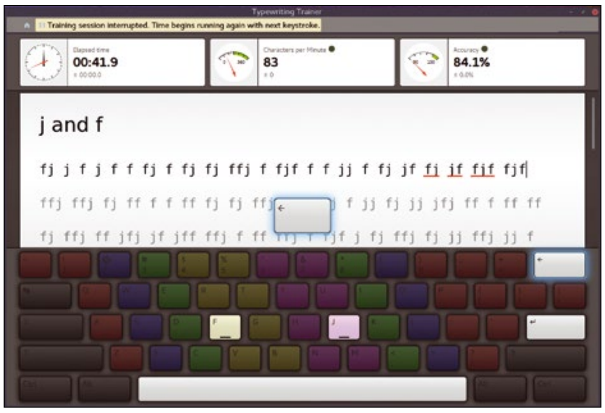
Project Website
https://padthv1.sourceforge.io/

Typing tutor

KTouch

Keyboardio is a wonderful, paradigm-shifting, ergonomic keyboard that was successfully crowdfunded two and a half years ago. As the video that accompanied the original campaign proclaimed when discussing making such a keyboard, "How hard can it be?" But it ended up being very hard. During those years, Kaia Dekker and Jesse Vincent have provided a compelling book's worth of excellent updates, flew to China more times than I can remember, and produced endless prototypes, as well as their own offspring! Years after the idea took shape and was subsequently funded, the wooden keyboards are finally making their way to hacker households. Which creates a new problem: How do you relearn to type on a keyboard with vertically aligned keys, no cursor control, two butterfly shaped keybeds, and a vastly different layout? You go back to basics.

KTouch is a brilliant tool that teaches you the basics of touch typing through a series of tutorials that unlock in stages as you improve. Unlike the many websites that hope to do the same, KTouch is very responsive and features a more game-like user interface. Speed dials show characters per minute and accuracy, for example, and the main view itself adapts to the keys being focused on. You start typing simple left-hand/right-hand combinations, such as *fj f f j j*, which are repeated, reflected, and isolated to the point at which it hopefully becomes muscle memory before moving on to more complex typing. It works brilliantly. Even if you type for a living, spending a few minutes a day will improve your speed, accuracy, and posture. It's also essential when learning how to use a weird wooden keyboard delivered at great cost several years after the original order.


Typewriting Trainer
Training session interrupted. Time begins running again with next keystroke.
Elapsed time
00:41.9
Characters per Minute
83
Accuracy
84.1%
j and f
fj j f j f f fj f fj fj ffj f fjf f f jj f fj jf fj jf fjf fjf
ffj ffj fj ff f f ff fj fj ffj f jj fj jj jfj ff f ff ff
fj ffj ff jfj jf jff ffj f ff jf j fj ffj fj jj ffj jj f

If you're suffering from a repetitive stress injury, one of the best things to do is learn how to touch type and adopt a more balanced posture.

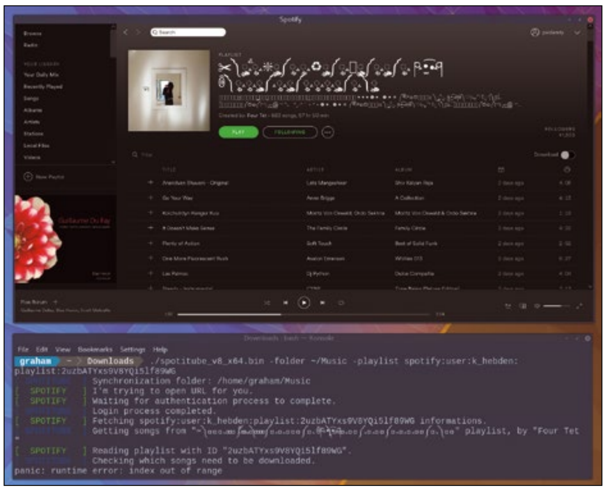
Project Website
https://www.kde.org/applications/education/ktouch

Music downloader

Spotitube

What this little tool does may be legally ambitious, depending on where you're located, but it's such a clever idea that it's worth covering. Spotitube can be asked to parse a Spotify playlist and look for each and every track from that playlist on YouTube. It then uses the awesome *youtube-dl* script to grab a copy of that track before then transcoding it into an audio format that you can use locally on your Linux machine. Of course, this process could well involve copyright infringement if the music isn't licensed to allow this format shifting, but it also takes into account the reality of how many people use YouTube, especially for old, rare, or unavailable material.

What's most interesting, however, is that this script can be used to legitimately bypass the lock-in you often experience with services like Spotify. If, for example, you've used Spotify for a number of years, it's likely you'll have built complex and infinitely edited playlists and collections to access from within the application itself. This is one of Spotify's best features, as is the ability to subscribe to and follow playlists created and curated by other people, often skilled musicians themselves. This script helps you extract these playlists and use them however you see fit, hopefully rebuilding the playlist with the specific tracks you buy from a DRM-free service. The script also offers lots of control over how the conversion takes place, including optional normalization, embedded lyrics, and the generation of an M3U playlist file. Perhaps its cleverest trick is being able to select the correct song on YouTube, using nothing more than the keywords it has learned from Spotify and its own algorithm (and your own Spotify account).



```
graham  ~  Downloads  ./spotitube_v8_x64.bin -folder ~/Music -playlist spotify:user:k_hebden:
playlist:2uzbATYxs9V8YQi5lf89WG
                 Synchronization folder: /home/graham/Music
[  SPOTIFY   ] I'm trying to open URL for you.
[  SPOTIFY   ] Waiting for authentication process to complete.
                 Login process completed.
[  SPOTIFY   ] Fetching spotify:user:k_hebden:playlist:2uzbATYxs9V8YQi5lf89WG informations.
                 Getting songs from "[illegible]" playlist, by "Four Tet
"
[  SPOTIFY   ] Reading playlist with ID "2uzbATYxs9V8YQi5lf89WG".
                 Checking which songs need to be downloaded.
panic: runtime error: index out of range
```

Spotitube harnesses the power of *youtube-dl* and FFmpeg to free your playlists from Spotify.

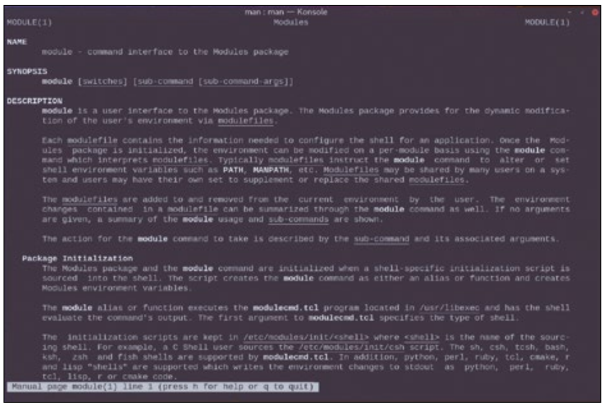
Project Website
https://github.com/streambinder/spotitube

Environment switcher

Environment Modules 4

With many of us now working from the same terminal in lots of different environments, one local configuration no longer fits all use cases. You might want a local build environment, for example, using environment variables, configuration files, and paths tailored for a specific version of GCC targeting a specific architecture, or you might work with lots of virtual machines locally before taking your scripts and applications to a high-performance computing platform. If you want to use your local configuration on these systems, you need some way to change the environment quickly. One option is to create different users for each of these, to copy all your configuration files back and forth before relogging into the same account, or to source each file before every switch. Environment Modules is a better way.

As implied by its name, Environment Modules enable you to create blocks of environmental variables and aliases and make them easily applicable and changeable. You could create one for the GCC example mentioned earlier, or for paths pointing to your local set of scripts. Because each would be a separate module, Environment Modules lets you pick and choose which parts to activate when you need to use them. You could start the session with GCC and a local `bin` in the path, for instance, before switching the latter module to point at a `bin` on a remote machine or network. Also worth noting is the excellent man page that tells you everything you need to know to get started, which could be confusing after installation, because the `module` command won't be in your path until you've configured your system. All it takes is a few commands and a little time to split up your current configuration, and you'll be able to add and remove environment configuration options dynamically and atomically, as and when you need to.



```
                                  man : man — Konsole
MODULE(1)                              Modules                              MODULE(1)

NAME
       module - command interface to the Modules package

SYNOPSIS
       module [switches] [sub-command [sub-command-args]]

DESCRIPTION
       module is a user interface to the Modules package. The Modules package provides for the dynamic modifica-
       tion of the user's environment via modulefiles.

       Each modulefile contains the information needed to configure the shell for an application. Once the  Mod-
       ules  package is initialized, the environment can be modified on a per-module basis using the module com-
       mand which interprets modulefiles. Typically modulefiles instruct the module  command  to  alter  or  set
       shell environment variables such as PATH, MANPATH, etc. Modulefiles may be shared by many users on a sys-
       tem and users may have their own set to supplement or replace the shared modulefiles.

       The modulefiles are added to and removed from the  current  environment  by  the  user.  The  environment
       changes  contained  in a modulefile can be summarized through the module command as well. If no arguments
       are given, a summary of the module usage and sub-commands are shown.

       The action for the module command to take is described by the sub-command and its associated arguments.

   Package Initialization
       The Modules package and the module command are initialized when a shell-specific initialization script is
       sourced  into the shell. The script creates the module command as either an alias or function and creates
       Modules environment variables.

       The module alias or function executes the modulecmd.tcl program located in /usr/libexec and has the shell
       evaluate the command's output. The first argument to modulecmd.tcl specifies the type of shell.

       The  initialization scripts are kept in /etc/modules/init/<shell> where <shell> is the name of the sourc-
       ing shell. For example, a C Shell user sources the /etc/modules/init/csh script. The sh, csh, tcsh, bash,
       ksh,  zsh  and fish shells are supported by modulecmd.tcl. In addition, python, perl, ruby, tcl, cmake, r
       and lisp "shells" are supported which writes the environment changes to stdout  as  python,  perl,  ruby,
       tcl, lisp, r or cmake code.
 Manual page module(1) line 1 (press h for help or q to quit)
```

Replace all those `if` statements in your Bash configuration file with modular environment variables and aliases.

Project Website
http://modules.sourceforge.net/

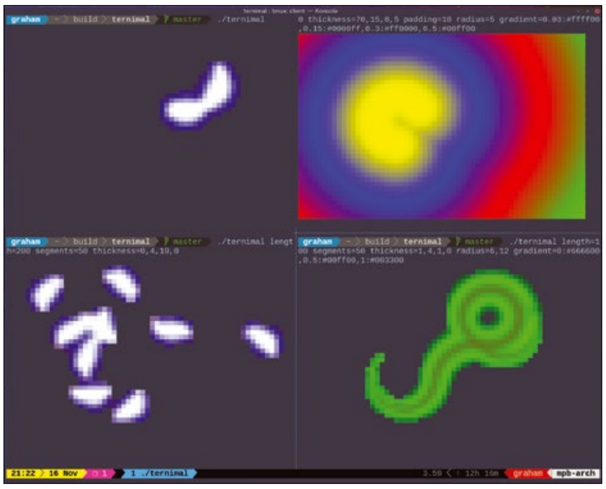
Console candy

Ternimal

This find is pure eye candy. It creates an animated life form in your terminal, which you can watch evolve and change within the confines of your safe character space. Playing with the many arguments can change the appearance of the life form from a worm-like snake to a swarm of angry will-o'-the-wisp after drinking too much caffeine. Colors, thickness, number of segments, size of arcs, and gradients can all be changed. While it serves no particular purpose, there are a few good reasons for trying it out.

One reason is that it puts your console under stress. This makes it an ideal burn-in tool for testing your environment, or perhaps comparing different consoles on your desktop. Because it works within Screen or tmux, it can also help you see how your console scales or works over a network. It's also implemented with Unicode characters providing the blocks that draw the animation. That means you can also test for Unicode support and speed; it helps you see whether the blocks assigned to those Unicode characters work as they should within the confines of your terminal.

The code is written in Rust, making this an excellent example to open up and take a look at if you want to learn a modern high-performance language that can do cool things on the command line. Installation is as simple as downloading the code and building it with the Rust compiler. While the 1,000-plus lines of code make this more than a simple hack, everything is well commented and easy enough to understand, making the addition of a few extra life forms or color modes the perfect place to get started with a little Rust programming.



Test your terminal's Unicode support with Ternimal.

Project Website
https://github.com/p-e-w/ternimal

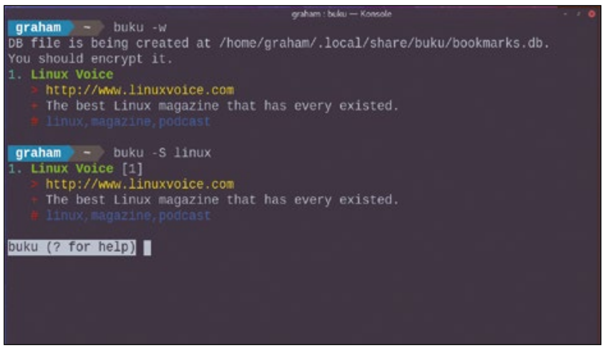
Bookmark manager

Buku

Bookmark managers, where you store the title and location of your favorite web destinations, are a little like password managers. We all used to trust online services and applications to hold our secret links, but with the changing times, many of us have changed our processes. Some of the best password managers are now operated from the command line, with new ones appearing every month, and it's a similar story with bookmark managers. Keeping your bookmarks local and in a single place, where any files can then be versioned with `git` or encrypted via GnuPG, gives you the kind of control you don't get with Firefox, Chrome, or any one of the bookmark add-ons designed for specific browsers.

Storing your bookmarks locally is what Buku does; it's a simple bookmark manager that hopes to put everything you need at your fingertips.

With Buku installed, the easiest way to invoke its functionality is to enter `buku -w`, which will open a simple template with your favorite text editor (e.g., Vim), with empty lines left for adding your URL, a title for the URL, tags, and any comments. This sounds more cumbersome than it really is, because it only takes a simple *Paste* and *Save* if you want to record a bookmark. It's a quicker process than doing the same thing from a browser, especially if you're already on the command line. You can perform the same function with arguments too, obviating the need for an editor completely. Tags are used with the internal search, which is a great way of retrieving your Linux-related bookmarks, for instance. You can then sync bookmarks with a web service, export them to HTML, or create scripts for your clipboard or for talking directly to your favorite browser.


```
graham  ~  buku -w
DB file is being created at /home/graham/.local/share/buku/bookmarks.db.
You should encrypt it.
1. Linux Voice
   > http://www.linuxvoice.com
   + The best Linux magazine that has every existed.
   # linux,magazine,podcast

graham  ~  buku -S linux
1. Linux Voice [1]
   > http://www.linuxvoice.com
   + The best Linux magazine that has every existed.
   # linux,magazine,podcast

buku (? for help) 
```
Buku can import HTML and Markdown bookmarks and be used as your central command-line bookmark repository.

Project Website
https://github.com/jarun/Buku

Google drive synchronization

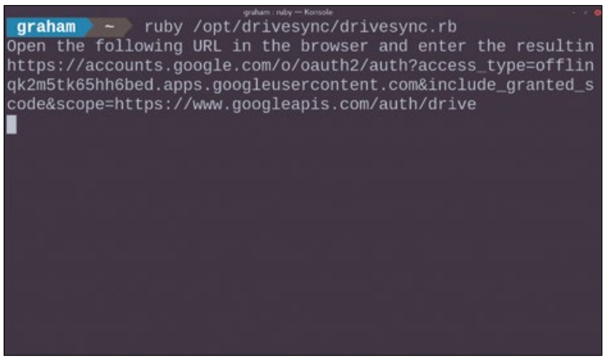
DriveSync

In many ways, it's remarkable that Google has yet to provide an official client for accessing its cloud filesystem; although, it may well be a blessing in disguise. A Google client may well collect more metrics, silently sync them to its servers, and be unlikely to be open source, making it less convenient, less privacy-conscious, and less easy to distribute. Perhaps this is why one hasn't been created. Credit goes to Google for tolerating the many third-party utilities that have come and gone over the years, most notably the desktop integration that pushes a local folder to your cloud storage. Having Google Drive integration is still worth the effort, though, even with the privacy and proprietary issues, because many of us now collaborate with colleagues in Google Docs, and it's the way many remote workers need to work.

DriveSync is one of the best. It's small, unobtrusive, and efficient, and it slots perfectly into the Linux way of doing things. Installation is via a `git clone` and execution of the Ruby `bundle` command. With that done, you simply need to edit a configuration file. Arch users will have the location of this file pointed out to them during installation (`/opt/drivesync/config.yml`). This allows you to change the local drive folder and change settings, such as whether locally deleted files will also be deleted remotely. Authentication is handled by running the command first, giving you a URL to which to point a browser. Finally, with that working, you need to add

```
*/1 * * * * ruby /opt/drivesync/drivesync.rb
```

to your crontab, which will take care of the synchronization process by running `drivesync.rb` every minute. With the process running, files will automatically be configured according to your configuration, and you won't have to worry about Google releasing an official client.


```
graham  ~  ruby /opt/drivesync/drivesync.rb
Open the following URL in the browser and enter the resultin
https://accounts.google.com/o/oauth2/auth?access_type=offlin
qk2m5tk65hh6bed.apps.googleusercontent.com&include_granted_s
code&scope=https://www.googleapis.com/auth/drive
```
Until Nextcloud can compete with Google Docs (hopefully soon!), a decent Google Drive sync utility is still a great addition.

Project Website
https://github.com/MStadlmeier/drivesync

Game compendium

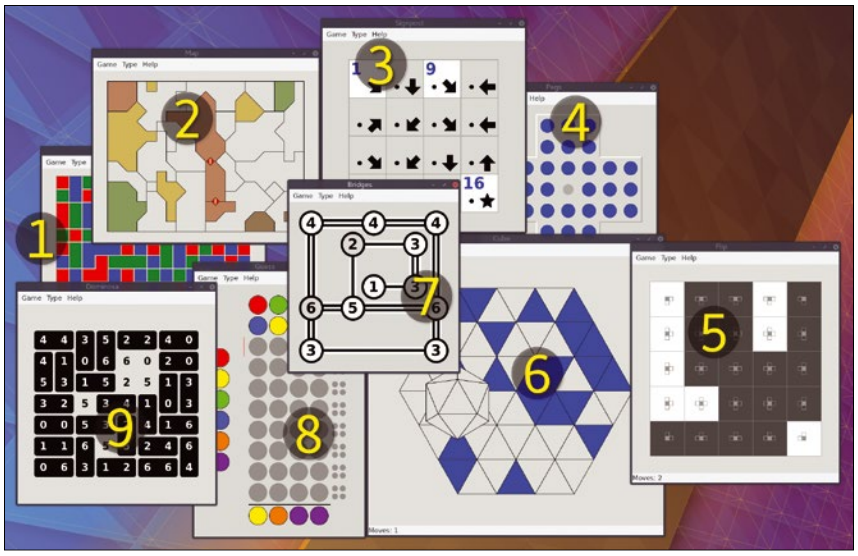
Simon Tatham's Portable Puzzle Collection

In the early days of computing, before the emergence of even single developer AAA games, you could buy collections of well-known, traditional pre-computer games. These were usually put together from source code or ideas that had been knocking around universities in the 1970s, pooled into a single collection with a menu, and sold in the back pages of a newspaper's Sunday supplement. These collections would often include games like checkers, crosswords, word searches, and even board or card games with simple AI opponents, with just enough variation to fill a rainy afternoon.

Simon Tatham's Portable Puzzle Collection feels very similar to those old compendiums. It's composed of 39 separate, simple, portable logic games that can all be played on your own whenever you have a spare five minutes to kill or need to switch context to help with concentration. Simon Tatham is also the author of the essential PuTTY SSH client for Microsoft Windows, helping thousands of Windows users access the joys of Linux servers running SSH. If you're familiar with PuTTY's functional and slightly austere UI, you'll know what to expect from these games. Usually, the only UI is a menu and the game itself, with which you can interact using the mouse, and quite a number of the games you'll have experienced before. Sudoku, Minefield, Mastermind, Solitaire, and Pipes all have open source (and renamed) equivalents, and it's a real nostalgia trip playing many of the games.

Names are purely descriptive, rather than hinting at the inspiration behind some of the games. Netslide, for example, is like an image-sliding game, where the image itself is a network that needs to be connected. Same Game requires you to remove touching groups of the same color, and Map has a similar challenge with a Risk-style playing field. The graphics may be perfunctory, but they're perfectly suited to the genre and style of the games. When there's animation, it's executed so smoothly you can hardly see the frames. Examples include the walking 3D cube in Cube, the compass bearing of the arrow as you drag and drop to make a link in Signpost, and the line dragging in Untangle as you attempt to position the points so the lines don't cross.

Every game requires some considered thinking, and each is different enough to present a contrasting challenge, initially as you work out how to win, and then by playing and increasing the skill level or the size of the game area from the menu. The menu can also be used to save the current state of the game, undo or redo moves, set specific seeds, and even solve the current screen if it all gets too much for you. Like those original compendiums, many of the games are simple and quick enough to play, but they're also like jazz standards or simple chord progressions, being the foundation for many other games or ideas that could be more ambitious. Because it's completely open source, cross-platform (including iOS and Android), and even ported to JavaScript, you can study the code, make your own changes, and easily build your own to add to the collection.



1 Same Game: Clear the grid by linking colors. **2 Map:** Clear the grid by never linking colors. **3 Signpost:** Make a path out of arrow directions. **4 Pegs:** Jump pegs and remove the jumped peg to clear the board. **5 Flip:** Light groups of squares. **6 Cube:** Roll a cube (or other 3D shape) over a path to clear up the squares. **7 Bridges:** Connect the island and make more large bridges. **8 Guess:** Try to work out the colors hidden at the bottom. **9 Dominosa:** Dominoes.

Project Website
https://www.chiark.greenend.org.uk/~sgtatham/puzzles/

Jump! Quick directory change at the command line

Bd, autojump, and Fasd improve the workflow for command-line aficionados thanks to quick navigation in the filesystem. BY FERDINAND THOMMES

With a little practice, working at the command line lets you feel the true power of Linux. However, the standard tools are not always as convenient as they are powerful, or as the user would hope.

If you frequently work at the command line, you most likely use the `cd` "change directory" command on a daily basis. In this article, I introduce a few helpers that promise greater convenience and speed, especially when changing directories in deeply nested paths. Bd, autojump, and Fasd are available as tools for different shells.

Inconvenient

The various Linux shells already provide some help with improved navigation in the command-line directory jungle, starting with finding out where you are. The `pwd` command helps by outputting your working directory's path. If you now want to jump up one or more levels, you can usually do this with:

```
cd ..
```

The two periods mean one step up in the direction of the root. You can jump up two steps by using:

```
cd ../..
```

However, frequent directory changes involve much typing, which can lead to errors. If you have to switch often between two directories in different places in the directory tree, the `cd -` command helps by taking you to the last directory you visited.

The directory stack provides further assistance: Multiple directories can be piled onto a stack. Bash provides the `pushd`, `popd`, and `dirs` commands to navigate in this stack [1]. If you replace `cd` with `pushd` for the directory change, the shell always places the directory to enter at the top of a stack. It remembers the stack – so you don't need to – and displays it after the command. Using `popd`, you can work your way back up the stack.

For orientation, the `dirs -l -v` command displays the entire stack. To create the stack, you first have to access the directories with `pushd` to add them to the stack, which proves useful if you have to switch between a few directories in a working session: The tool consecutively numbers the stack, and you can jump to the folders with the use of these numbers.

However, bd, autojump, and Fasd offer a more convenient approach; all the major distributions have them in their repositories. The `whohas` command, installed via your package manager, tells you where to find them. Typing `whohas autojump` gives you a list of distributions with the package, including the version and the respective branch.

You also can set up the bd [2] script for many distributions via a package manager. After installing, enter the two commands from Listing 1 to facilitate running bd. If bd is not available in the Linux derivative that you use, install it manually following notes from the project site on GitHub.

Shorter Return Path

Imagine that you have used `cd` to enter the `/home/fritz/foo/bar/bat/test/bd/` directory and now want to go back to `/home/fritz/foo/bar/`. You would usually enter:

```
cd ../../../
```

With the bd script you just installed, `bd bar` would do the trick: bd supports autocompletion, so you do not have to type out the directory names; usually you just need two or three letters.

Bd only operates backward, not forward. But it can also be used with other commands, for example: `ls`, `du`, `zip`, and `tar`. The command

Listing 1: Running bd

```
$ echo alias bd='. bd -si' >> ~/.bashrc
$ source ~/.bashrc
```

```
ls -l `bd ba`
```

lists the content of the /bar directory, even though you are still in /bd. The call

```
du -cs `bd fr`
```

displays the size of /fritz (Figure 1).

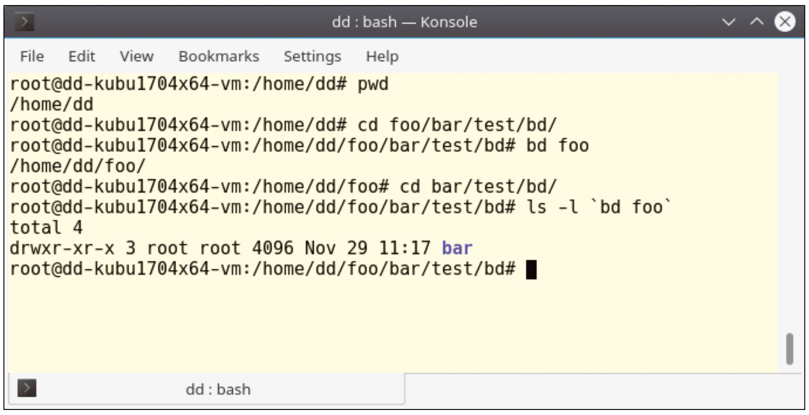
autojump

Autojump [3] makes use of the basics of the directory stack described above and further expands on them. It also facilitates navigation in the directory tree in both directions, unlike bd. Autojump can be used under Linux, Mac OS, and Windows with the Bash (from version 4.0), Zsh, and fish shells and experimentally with the tcsh and Clink shells.

Most distributions offer autojump packages. Under Debian and its subsidiaries; you can add the tool to the system using the command from Listing 2. Autojump works with a database that you initially have to fill by working with cd for a while. Alternatively, you can target folders that you use often.

You will find the database under ~/.local/share/autojump/autojump.txt. You can query the contents of the database at any time using j --stat. The j --purge call removes directories that no longer exist from the database.

If you also want to test Fasd [4], the final helper discussed in this article, it makes sense to install it now, because it also benefits from the actions for filling the database. To do this, type the entries from Listing 3 after the install.



```
root@dd-kubu1704x64-vm:/home/dd# pwd
/home/dd
root@dd-kubu1704x64-vm:/home/dd# cd foo/bar/test/bd/
root@dd-kubu1704x64-vm:/home/dd/foo/bar/test/bd# bd foo
/home/dd/foo/
root@dd-kubu1704x64-vm:/home/dd/foo# cd bar/test/bd/
root@dd-kubu1704x64-vm:/home/dd/foo/bar/test/bd# ls -l `bd foo`
total 4
drwxr-xr-x 3 root root 4096 Nov 29 11:17 bar
root@dd-kubu1704x64-vm:/home/dd/foo/bar/test/bd#
```

Figure 1: The bd script quickly takes you back to the directory tree and allows additional commands to be integrated.

Database Based

You can easily jump into directories that autojump keeps in its database using the j wrapper. For example, j Dow would take you to the ~/Downloads directory. In the same way, j loa also takes you there. The depth of the directory

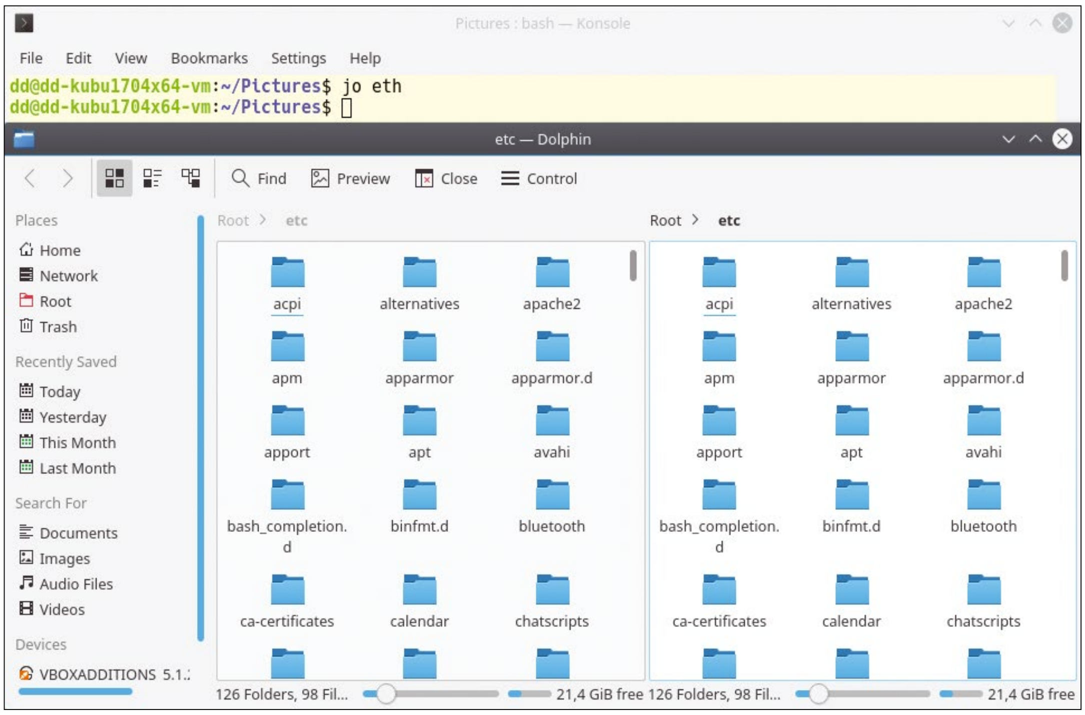
Listing 2: Installing autojump

```
$ echo '. /usr/share/autojump/autojump.sh'  >> ~/.bashrc
$ source ~/.bashrc
```

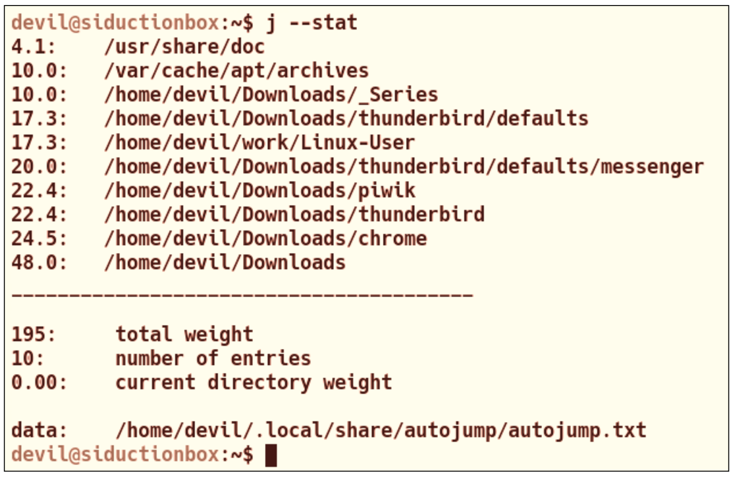
Listing 3: Installing Fasd

```
$ echo eval '$(fasd --init auto)'  >> ~/.bashrc
$ source ~/.bashrc
```

Figure 2: If you have lost track at the command line, autojump will also open the desired directory in the file manager.



```
dd@dd-kubu1704x64-vm:~/Pictures$ jo eth
dd@dd-kubu1704x64-vm:~/Pictures$
```



```
devil@siductionbox:~$ j --stat
4.1:     /usr/share/doc
10.0:    /var/cache/apt/archives
10.0:    /home/devil/Downloads/_Series
17.3:    /home/devil/Downloads/thunderbird/defaults
17.3:    /home/devil/work/Linux-User
20.0:    /home/devil/Downloads/thunderbird/defaults/messenger
22.4:    /home/devil/Downloads/piwik
22.4:    /home/devil/Downloads/thunderbird
24.5:    /home/devil/Downloads/chrome
48.0:    /home/devil/Downloads
_______________________________________

195:      total weight
10:       number of entries
0.00:     current directory weight

data:     /home/devil/.local/share/autojump/autojump.txt
devil@siductionbox:~$
```

Figure 3: The numbers in the database are used for internal weighting and state which directories were visited and how often.

hierarchy is not important; the main thing is that you visited the desired directory at least once using `cd`.

If you want to switch to your file manager for a better overview, `jo Dow` opens the directory for Plasma in Dolphin or for Gnome in Nautilus (Figure 2). If you just enter `j`, the command jumps to the most frequently visited directory, for which Autojump uses internal weighting.

If you call the content of the database with `j --stat`, you will see different numbers in front of the respective paths, for example:

```
22.4: /home/devil/Downloads/piwik
```

This sequence of digits is the internal weighting (Figure 3). As an example, the following directories are in the database:

```
30 /home/user/mail/inbox
10 /home/user/work/inbox
```

If you enter `j in`, high weighting causes autojump to switch to `/home/user/mail/inbox/`, whereas `j w in` would be the correct command to access the lower weighted directory. You can use the `--increase` (`-i`) and `--decrease` (`-d`) parameters to change the weighting, and `j --help` for additional information about syntax.

Alternatively, if you have several similar directories, you can display the numbered entries by means of tab-based autocompletion and then select the corresponding entry by number.

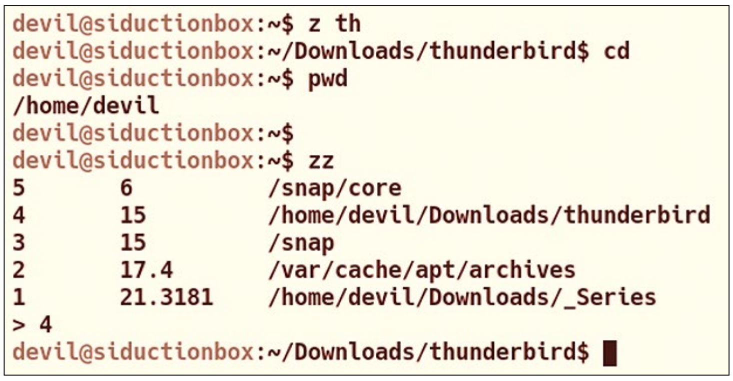
A helpful tip when using autojump: Sometimes the database suddenly empties, which can be extremely annoying, because you then have to start indexing from scratch. Sometimes it is better to back up the database manually.

Turbo

Fasd [4], the final tool covered in this article, was inspired by autojump. However, the software exceeds its role model: Fasd no longer restricts the navigation options to the directory tree, and it also indexes the data. Fasd works with the Ash, Bash, Zsh, mksh, pdksh, dash, and BusyBox shells under Linux, as well as `/bin/sh` in FreeBSD 9 and OpenBSD.

Fasd lets you open files by entering just a few characters in Vim or your default editor. You can view PDFs or play movies and music with MPlayer or your favorite player. The letters in the Fasd acronym stand for predefined aliases that process different types of data.

When you enter `f`, the tool shows you the visited files, including their weighting. Entering `d` provides the same result for the directories, and `a` lists both. `zz` takes you to a view in which you select an entry by the prefixed number (Figure 4).



```
devil@siductionbox:~$ z th
devil@siductionbox:~/Downloads/thunderbird$ cd
devil@siductionbox:~$ pwd
/home/devil
devil@siductionbox:~$
devil@siductionbox:~$ zz
5       6         /snap/core
4       15        /home/devil/Downloads/thunderbird
3       15        /snap
2       17.4      /var/cache/apt/archives
1       21.3181   /home/devil/Downloads/_Series
> 4
devil@siductionbox:~/Downloads/thunderbird$ 
```

Figure 4: In Fasd, you can select the desired directory using `z` and a few characters. You can select from the database by entering `zz`.

Defining additional aliases lets you open various files: for example, PDFs (Figure 5). In turn, you can create the following proxy in `.bashrc`:

```
alias o='a -e xdg-open'
```

Now source the file, as shown in Listings 2 and 3. The following also applies: Like autojump, Fasd only detects objects that have already been indexed.

Movies and music also can be started with just a few keystrokes. For this to work, enter

```
alias m='f -e mplayer'
```

in `.bashrc`. Alternatively, use a different player or change the `m` call switch to another. Then, simply start the movies or music using `m<part of title>`.

Like autojump, Fasd also lets you use letters from the middle of the name. The aliases' structure is simple: `-e` stands for execute. For files, select `f`; for directories, `d`. If the alias includes both, as with xdg-open [5], select `a`. On GitHub, the project provides an overview of the tool's many other possibilities.

Conclusions

All three tools presented in this article have one thing in common: They do not run in drop-down terminals, such as Yakuake or Guake, just in consoles or terminals. However, you will not read this anywhere, which led to me occasionally doubting the tools' capabilities in the lab. All three helpers are meaningful, and they build on each other.

If you do not constantly work at the command line, bd makes it easy to navigate without having to learn something new. Autojump offers much more and jumps into directories in random directions. This suits the workflow of users who frequently work on servers and like to handle administrative work at the command line. After the introduction phase, autojump saves some time here.

Fasd expands the feature set again, because it also indexes files and thus takes control of all the files on the computer. It also offers professional options to further customize the approach to suit your own needs. ■■■

Info

[1] Shell built-ins: *https://www.howtoforge.com/tutorial/linux-command-line-tips-tricks-part-2/*

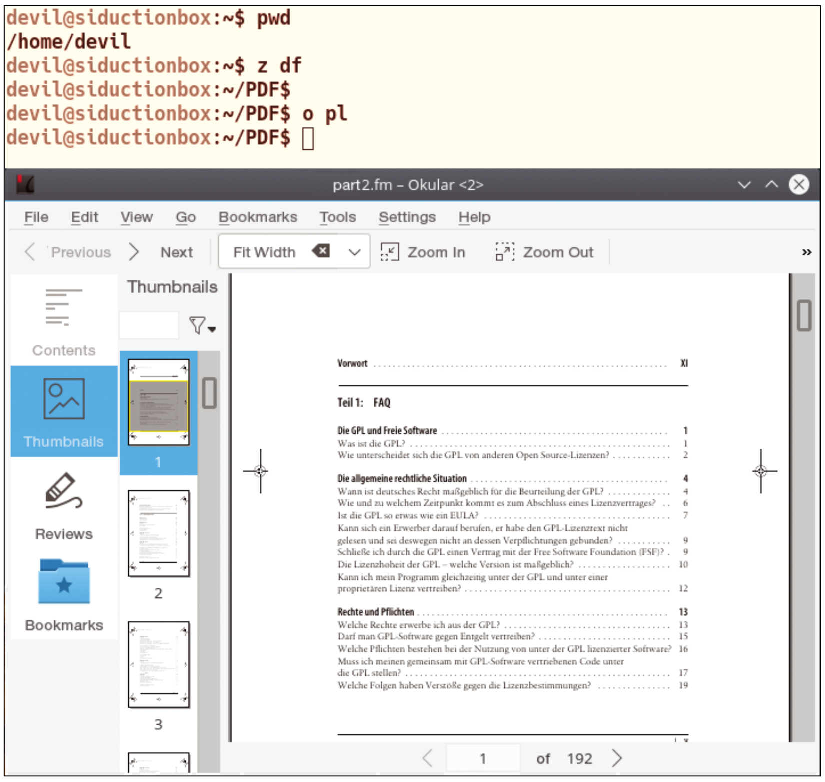
[2] bd: *https://github.com/vigneshwaranr/bd*

[3] autojump: *https://github.com/wting/autojump*

[4] Fasd: *https://github.com/clvv/fasd*

[5] xdg-open: *http://www.ubuntugeek.com/tag/xdg-open-example*

Figure 5: Defining additional aliases extends Fasd to display PDFs and other formats.



```
devil@siductionbox:~$ pwd
/home/devil
devil@siductionbox:~$ z df
devil@siductionbox:~/PDF$
devil@siductionbox:~/PDF$ o pl
devil@siductionbox:~/PDF$ 
```

Vorwort ... XI

Teil 1: FAQ

Die GPL und Freie Software ... 1
Was ist die GPL? ... 1
Wie unterscheidet sich die GPL von anderen Open Source-Lizenzen? ... 2

Die allgemeine rechtliche Situation ... 4
Wann ist deutsches Recht maßgeblich für die Beurteilung der GPL? ... 4
Wie und zu welchem Zeitpunkt kommt es zum Abschluss eines Lizenzvertrages? ... 6
Ist die GPL so etwas wie ein EULA? ... 7
Kann sich ein Erwerber darauf berufen, er habe den GPL-Lizenztext nicht gelesen und sei deswegen nicht an dessen Verpflichtungen gebunden? ... 9
Schließe ich durch die GPL einen Vertrag mit der Free Software Foundation (FSF)? ... 9
Die Lizenzhoheit der GPL – welche Version ist maßgeblich? ... 10
Kann ich mein Programm gleichzeitig unter der GPL und unter einer proprietären Lizenz vertreiben? ... 12

Rechte und Pflichten ... 13
Welche Rechte erwerbe ich aus der GPL? ... 13
Darf man GPL-Software gegen Entgelt vertreiben? ... 15
Welche Pflichten bestehen bei der Nutzung von unter der GPL lizenzierter Software? ... 16
Muss ich meinen gemeinsam mit GPL-Software vertriebenen Code unter die GPL stellen? ... 17
Welche Folgen haben Verstöße gegen die Lizenzbestimmungen? ... 19

First Aide

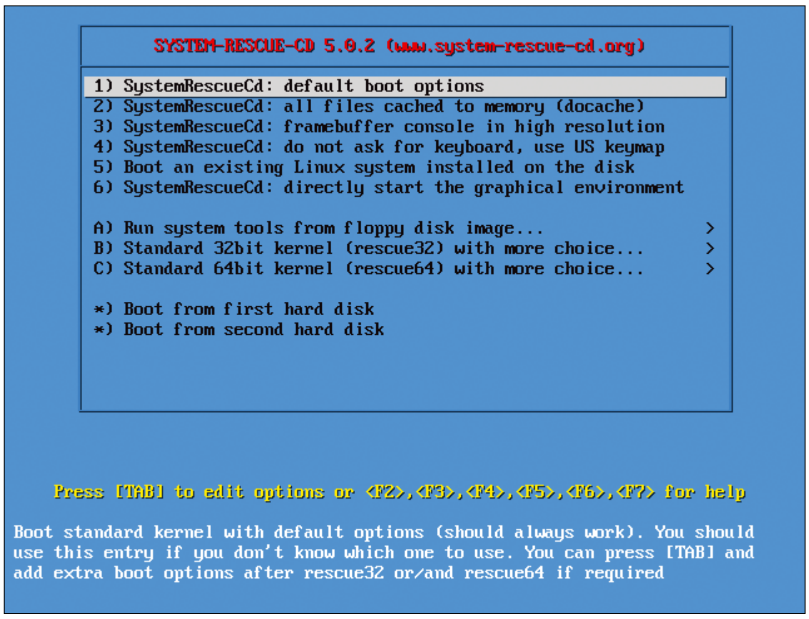
If you accidentally delete data or format a disk, good advice can be expensive. Or maybe not: You can undo many data losses with SystemRescueCd.

BY ERIK BÄRWALDT

The price for mass storage devices of all types has been falling steadily in recent years, with a simultaneous increase in capacity. As a result, users are storing more and more data on local storage media – often without worrying about backing it up. Once the milk has been spilled, the anxious search begins for important photos, videos, correspondence, and spreadsheets. SystemRescueCd can help in these cases by providing a comprehensive toolbox for every computer, with the possibility of restoring lost items.

The Gentoo derivative is a hybrid image for 32- and 64-bit computers that comes in just under 470MB [1]. The entire distro fits on a CD, so it is also suitable for use on older systems. To boot the operating system from a USB stick, use the commands:

```
$ isohybrid ⮐
  systemrescuecd-x86-<Version>.iso
$ sudo dd ⮐
  if=systemrescuecd-x86-<Version>.iso ⮐
  of=/dev/sd<X> bs=1M
```



```
SYSTEM-RESCUE-CD 5.0.2 (www.system-rescue-cd.org)

1) SystemRescueCd: default boot options
2) SystemRescueCd: all files cached to memory (docache)
3) SystemRescueCd: framebuffer console in high resolution
4) SystemRescueCd: do not ask for keyboard, use US keymap
5) Boot an existing Linux system installed on the disk
6) SystemRescueCd: directly start the graphical environment

A) Run system tools from floppy disk image...            >
B) Standard 32bit kernel (rescue32) with more choice...  >
C) Standard 64bit kernel (rescue64) with more choice...  >

*) Boot from first hard disk
*) Boot from second hard disk

Press [TAB] to edit options or <F2>,<F3>,<F4>,<F5>,<F6>,<F7> for help

Boot standard kernel with default options (should always work). You should
use this entry if you don't know which one to use. You can press [TAB] and
add extra boot options after rescue32 or/and rescue64 if required
```

Figure 1: SystemRescueCd offers a variety of startup options, including the ability to launch various floppy disk images.

Replace the drive information `sd<X>` in the copy command with the correct device name for the USB storage medium used.

Diversity

When starting the Live system, the GRUB boot manager (Figure 1) welcomes you with numerous options that include different kernels for different Intel hardware architectures.

The system can also be copied completely into main memory and operated from there, which has advantages for very old and correspondingly slow computers. To avoid problems with the graphics card, the distribution also integrates a standard VESA graphics driver. In the case of very old computer systems with 4:3 aspect monitors, the graphical user interface can also be started in SVGA or XGA resolution.

Various diagnostic tools are available for retrieval as floppy disk images, which is particularly useful if you can narrow down the cause of data loss and do not need a complete operating system with all the tools. Memtest and HDT images are available for hardware and monitoring tests without long start times.

If you suspect damage to the hard disk, start the MHDD tool from the boot manager for low-level checks of the volumes. If the target system is used in a heterogeneous environment, you can also change or reset passwords from other operating systems using the SystemRescueCd boot manager.

The *default boot options* (option 1 on the start screen) and *directly start the graphical environment* (option 6) start the system and provide the entire collection of software. Whereas *default boot options* opens a root console after manually setting the keyboard layout, *directly start the graphical environment* calls the X server after modifying the keyboard layout and then the Xfce desktop.

A number of editors are available from the console, including text-based web browser links, after manually setting up network access. To start the graphical user interface from the console, enter the `startx` command. After a short time, a deliberately simple and somewhat rustic-looking Xfce desktop with an open terminal window appears (Figure 2).

Under Xfce, you can easily configure access to the Internet for various technologies, either from *Settings | Network Connections* or by clicking the network icon at bottom right in the panel. It is currently not possible to set up wireless access to the Internet with state-of-the-art encryption technology at the command line. You need to use a wired connection. Set up access with the `net-setup <interface>` call by using `eth0` for the first LAN interface, `eth1` for the second, and so on.

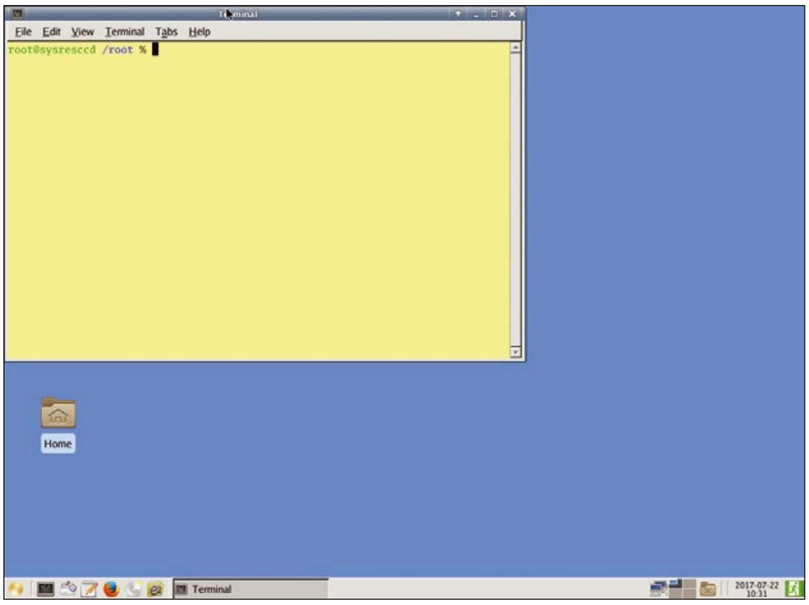
Software

SystemRescueCd software inventory is aligned strictly with the requirements of a rescue system and therefore dispenses with the usual standard programs, such as LibreOffice and Gimp. True to the purpose of the application, it is a purely Live system that cannot be installed permanently from the boot manager or the graphical user interface. If you still want to set up SystemRescueCd manually on a mass storage device, it is possible to install the system on a partition with a filesystem supported by Linux. However, you need to make a number of manual corrections. Alternatively, it works well on a Windows filesystem [2].

Although the Xfce desktop offers the usual *Multimedia* and *Office* groups in its start menu, they do not contain any software packages for everyday use. Only the ePDFViewer is stored under *Office*, and ISO Master and Xfburn programs are found under *Multimedia*. The applications in the Internet submenu are also strictly oriented toward system rescue tasks: The Firefox web browser version 52.1.2 ESR (extended support release) offers a private mode but does without preinstalled extensions. Additionally, you will find only the TigerVNC Viewer for controlling remote computers and the GTKTerm terminal.

The features in the *Accessories* submenu are more extensive. You can choose between the Thunar and emelFM2 file managers, and you can start up gVim as a graphical front end for the Vim editor. The Xfburn program burns optical discs, and the Bulk Rename graphical tool allows you to rename a large number of files automatically in batch mode. In addition to a process list, a simple task manager graphically displays RAM and CPU usage (Figure 3).

In the bar at the bottom of the screen, SystemRescueCd provides quick-start icons for some commonly used applications. In addition to the file manager emelFM2 (Figure 4), you'll find the graphical partitioning tool GParted, as well as Xfburn, Firefox, the Geany text editor, and a terminal.



```
root@sysresccd /root %
```

Figure 2: The Xfce desktop, designed for use on old computers or computers with limited resources, dispenses with frills.

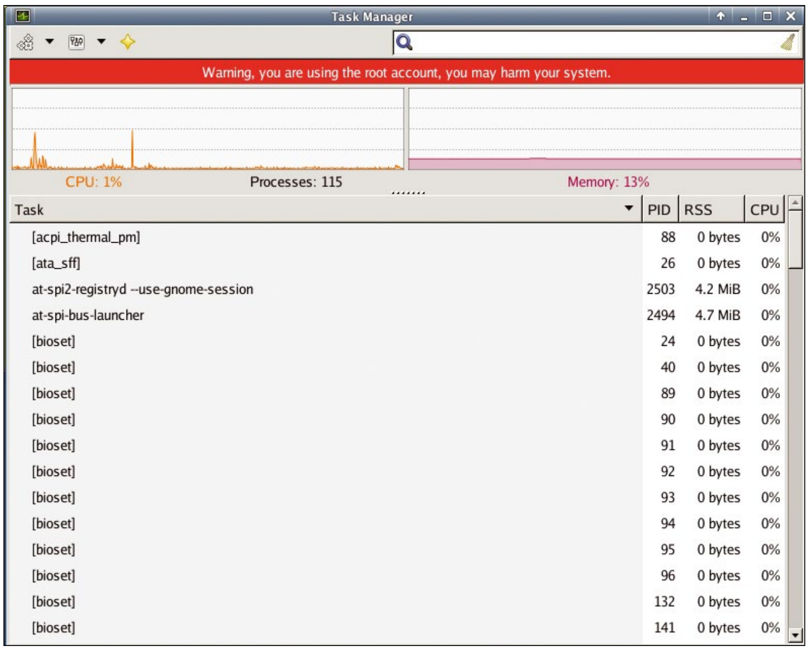
The System menu offers up the well-established Midnight Commander, which is an ncurses program that takes over the management of files and directories visually, simply, and efficiently.

Test Programs

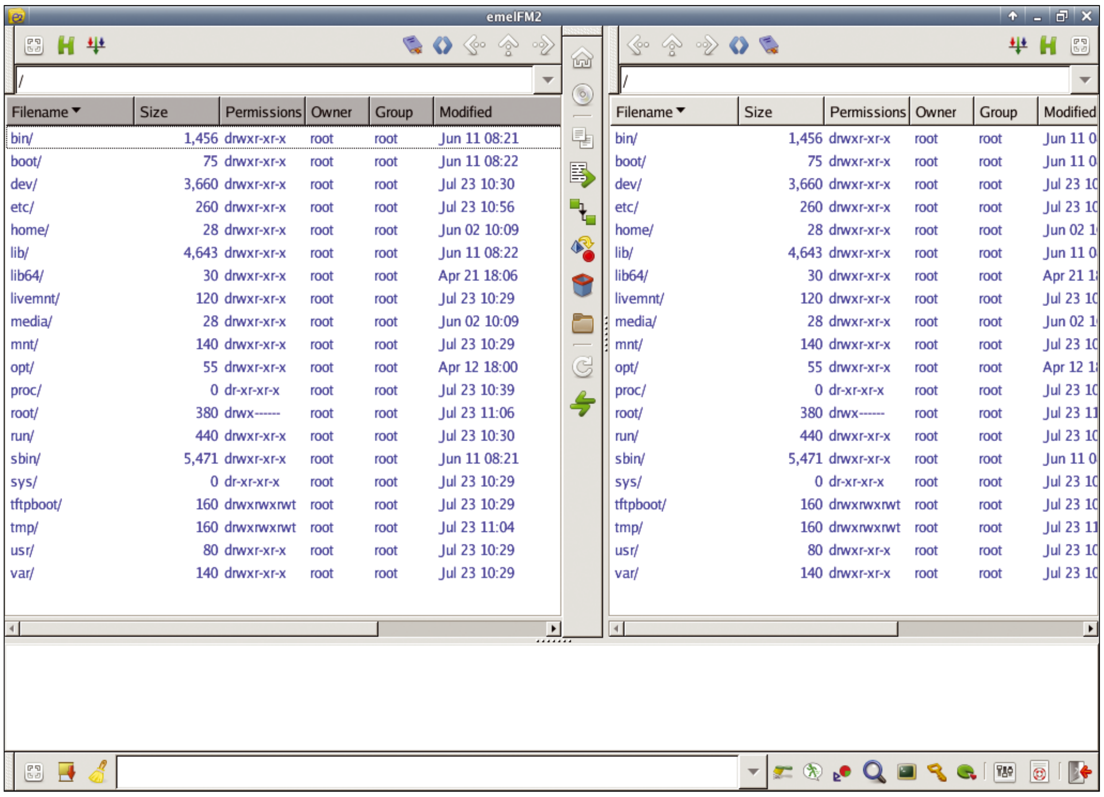
Under Linux, you can usually troubleshoot and locate hardware problems quickly thanks to numerous test and monitoring programs. SystemRescueCd offers a useful mix of tools for the graphical user interface and the terminal. The System submenu holds not only Htop, which lists running processes and their resource consumption, but also Hardware Lister, which graphically displays the hardware components of the target system.

This inventory tool is particularly useful when it comes to checking the revision status of the

Figure 3: The task manager provides information about system resource utilization and running processes.



```
Warning, you are using the root account, you may harm your system.

CPU: 1%                 Processes: 115                    Memory: 13%

Task                                          PID   RSS       CPU
  [acpi_thermal_pm]                             88  0 bytes    0%
  [ata_sff]                                     26  0 bytes    0%
  at-spi2-registryd --use-gnome-session       2503  4.2 MiB    0%
  at-spi-bus-launcher                         2494  4.7 MiB    0%
  [bioset]                                      24  0 bytes    0%
  [bioset]                                      40  0 bytes    0%
  [bioset]                                      89  0 bytes    0%
  [bioset]                                      90  0 bytes    0%
  [bioset]                                      91  0 bytes    0%
  [bioset]                                      92  0 bytes    0%
  [bioset]                                      93  0 bytes    0%
  [bioset]                                      94  0 bytes    0%
  [bioset]                                      95  0 bytes    0%
  [bioset]                                      96  0 bytes    0%
  [bioset]                                     132  0 bytes    0%
  [bioset]                                     141  0 bytes    0%
```



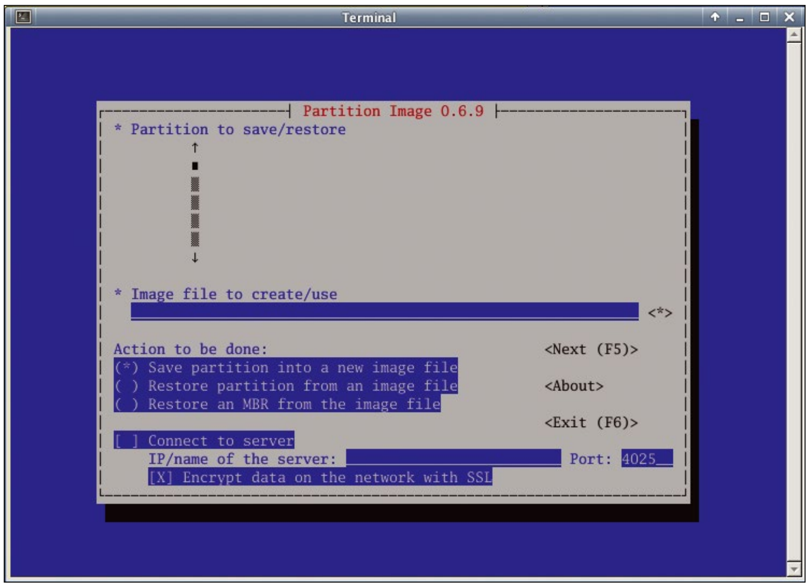
emelFM2

Filename	Size	Permissions	Owner	Group	Modified
bin/	1,456	drwxr-xr-x	root	root	Jun 11 08:21
boot/	75	drwxr-xr-x	root	root	Jun 11 08:22
dev/	3,660	drwxr-xr-x	root	root	Jul 23 10:30
etc/	260	drwxr-xr-x	root	root	Jul 23 10:56
home/	28	drwxr-xr-x	root	root	Jun 02 10:09
lib/	4,643	drwxr-xr-x	root	root	Jun 11 08:22
lib64/	30	drwxr-xr-x	root	root	Apr 21 18:06
livemnt/	120	drwxr-xr-x	root	root	Jul 23 10:29
media/	28	drwxr-xr-x	root	root	Jun 02 10:09
mnt/	140	drwxr-xr-x	root	root	Jul 23 10:29
opt/	55	drwxr-xr-x	root	root	Apr 12 18:00
proc/	0	dr-xr-xr-x	root	root	Jul 23 10:39
root/	380	drwx------	root	root	Jul 23 11:06
run/	440	drwxr-xr-x	root	root	Jul 23 10:30
sbin/	5,471	drwxr-xr-x	root	root	Jun 11 08:21
sys/	0	dr-xr-xr-x	root	root	Jul 23 10:29
tftpboot/	160	drwxrwxrwt	root	root	Jul 23 10:29
tmp/	160	drwxrwxrwt	root	root	Jul 23 11:04
usr/	80	drwxr-xr-x	root	root	Jul 23 10:29
var/	140	drwxr-xr-x	root	root	Jul 23 10:29

Filename	Size	Permissions	Owner	Group	Modified
bin/	1,456	drwxr-xr-x	root	root	Jun 11 0
boot/	75	drwxr-xr-x	root	root	Jun 11 0
dev/	3,660	drwxr-xr-x	root	root	Jul 23 1
etc/	260	drwxr-xr-x	root	root	Jul 23 1
home/	28	drwxr-xr-x	root	root	Jun 02 1
lib/	4,643	drwxr-xr-x	root	root	Jun 11 0
lib64/	30	drwxr-xr-x	root	root	Apr 21 1
livemnt/	120	drwxr-xr-x	root	root	Jul 23 1
media/	28	drwxr-xr-x	root	root	Jun 02 1
mnt/	140	drwxr-xr-x	root	root	Jul 23 1
opt/	55	drwxr-xr-x	root	root	Apr 12 1
proc/	0	dr-xr-xr-x	root	root	Jul 23 1
root/	380	drwx------	root	root	Jul 23 1
run/	440	drwxr-xr-x	root	root	Jul 23 1
sbin/	5,471	drwxr-xr-x	root	root	Jun 11 0
sys/	0	dr-xr-xr-x	root	root	Jul 23 1
tftpboot/	160	drwxrwxrwt	root	root	Jul 23 1
tmp/	160	drwxrwxrwt	root	root	Jul 23 1
usr/	80	drwxr-xr-x	root	root	Jul 23 1
var/	140	drwxr-xr-x	root	root	Jul 23 1

Figure 4: The emelFM2 file manager comes with a GTK-based interface and follows in the footsteps of Midnight Commander.

components in the computer (e.g., to indicate whether you should update the firmware). The graphical user interface uses information from the Lshw command-line program, which often provides much more detailed information. The `iotop` tool is not directly accessible from a menu but can be started by entering the `iotop` command in the terminal. In the event of sporadically occurring high system load and associated high latencies, `iotop` can alert you to problems with individual hardware components that bury the entire system under a flood of IRQ requirements. Server administrators, especially, appreciate this small program.



```
Terminal

┌──────────────────┤ Partition Image 0.6.9 ├──────────────────┐
│ * Partition to save/restore                                  │
│                                                              │
│ * Image file to create/use                                   │
│                                                        <*>   │
│                                                              │
│ Action to be done:                          <Next (F5)>      │
│ (*) Save partition into a new image file                     │
│ ( ) Restore partition from an image file    <About>          │
│ ( ) Restore an MBR from the image file                       │
│                                             <Exit (F6)>      │
│ [ ] Connect to server                                        │
│     IP/name of the server: ____________ Port: 4025_          │
│     [X] Encrypt data on the network with SSL                 │
└──────────────────────────────────────────────────────────────┘
```

Figure 5: Old-fashioned, but useful: Partition Image clones any partitions and restores them if necessary.

Safe and Secure

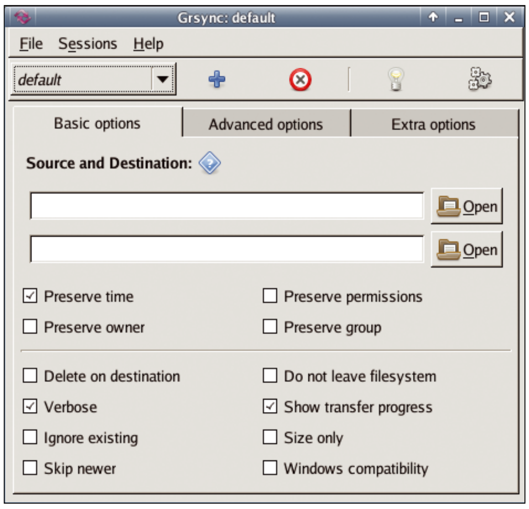
The distribution places particular emphasis on data recovery. Therefore, it comes with many tools for the maintenance and care of mass storage. For this purpose, the System submenu harbors the Show Filesystems, GParted, Partimage, and Testdisk entries.

Show Filesystems opens a terminal and calls `fsarchiver`; the Partition Image ncurses program hides behind the Partimage entry, which allows you to create images of hard disk partitions in a few steps. Because this is also possible with system partitions, you could have a snapshot available in minutes to restore the original system (Figure 5).

The GParted graphical program allows you to edit partition tables of mass storage devices; the software can handle a number of different filesystems and includes external storage media, if required. The current versions of the SystemRescueCd contain GParted in the new 5.x version, which also gets along with the modern Btrfs filesystem.

The powerful terminal program Testdisk is suitable not only for reconstructing partitions, but also for restoring the boot sector of mass storage devices in case of accidental or malicious boot sector destruction. The graphical program Grsync helps with file and directory synchronization. The application is based on the Rsync command-line tool and uses its most important parameters (Figure 6).

The Rsnapshot command-line program, which is also based on Rsync, creates snapshots of entire partitions – much like Partition Image. It is also suitable for the use of an external USB hard disk as a backup medium. With multiple snapshots on a single target medium, Rsnapshot only saves copies of unchanged files once; it then uses hardlinks to these files in subsequent snapshots, saving storage space. However, newer backups inevitably depend on the older ones – if they are missing, the reconstruction fails. Search for Rsnapshot in the SystemRescueCd menus in vain – the software is called directly from the command prompt.



☑ Preserve time
☐ Preserve owner
☐ Preserve permissions
☐ Preserve group
☐ Delete on destination
☑ Verbose
☐ Ignore existing
☐ Skip newer
☐ Do not leave filesystem
☑ Show transfer progress
☐ Size only
☐ Windows compatibility

Figure 6: The small Grsync tool synchronizes your data with just a few mouse clicks. It is based on Rsync, the well-known sync software.

Despite its name, the `tob` (tape-oriented backup) shell script not only saves data on tape, but also on conventional filesystems. Its numerous options are revealed by the `tob --help` command.

Photorec, which is also called in the terminal, teams up with Testdisk for data reconstruction. The duo restores accidentally deleted data or data that is no longer accessible because of a hardware defect. Although its name suggests otherwise, Photorec is not limited to restoring digital images: It also knows many other file formats and reconstructs them.

Extra Equipment

For performance comparisons between individual hardware components or complete computer systems, SystemRescueCd comes with a number of prominent benchmark programs. The most popular, Bonnie++ and Stress, are launched in a terminal window and thus don't appear in the Xfce menus.

By measuring the read and write throughput of mass storage devices, Bonnie++ can provide information on possible hardware defects in the event of poor system performance. On the other hand, the Stress benchmark tool creates high load on various hardware components (e.g., CPU, memory, bus). Like Bonnie++, you control the stress level through a variety of parameters (see the list with `stress --help`); `htop` then lets you see how much stress the system is under.

SystemRescueCd also comes with some forensic programs, including chkrootkit, which scans the computer for hidden malware that opens a back door for attackers, and CmosPwd, which reveals BIOS passwords. Because these passwords are stored in EEPROM modules on many computers, especially notebooks, unrestricted access is not possible. For experienced users, CmosPwd offers a way to read or modify passwords. However, it primarily considers older BIOS variants and cannot cope with newer versions, especially (U)EFI systems [3].

Magic Rescue is a useful tool for reconstructing file content. However, the software does not use the filesystem allocation tables but relies on "magic numbers," which are located in the header of the respective files and denote the file type. As a result, Magic Rescue works even with corrupted or destroyed file allocation tables. The command-line program Foremost also recovers damaged or deleted files by using Magic Rescue information from the standardized file headers and footers.

SystemRescueCd also allows a more comprehensive analysis of network access with the standard console tools Nmap, Traceroute, Netcat, and Netselect; graphical packages such as Zenmap and Wireshark are missing. Therefore, if you have network-specific problems, it is better to use specialized distributions like Wifislax [4] or Kali Linux [5].

Conclusions

SystemRescueCd v5.0.4 is fast, stable, and contains hardly any unnecessary ballast. The developers have removed software that is not critical to its mission, as well as several programs with overlapping functions. The resource-saving Xfce desktop and a concentration of proven command-line tools allow SystemRescueCd to be used on computers with old or incompatible graphics hardware.

The developers have taken great care in putting the system together. For example, the latest updates have improved many central programs and replaced less powerful applications with better ones. The integration of tools from other operating system worlds – including DOS applications that can be started separately – also makes SystemRescueCd ready for data recovery in a heterogeneous environment. ■■■

Info

[1] Download SystemRescueCd: *https://www.system-rescue-cd.org/Download*

[2] Installation manual: *http://www.system-rescue-cd.org/manual/Installing_SystemRescueCd_on_the_disk/*

[3] CmosPwd documentation: *http://www.cgsecurity.org/wiki/CmosPwd*

[4] Wifislax: *http://www.wifislax.com* (in Spanish)

[5] Kali Linux: *https://www.kali.org*

Docker Talk – Docker swarms the Kubernetes community

At this year's DockerCon Europe, Docker announced that it is officially supporting the Google-sponsored Kubernetes ochestration engine – an unexpected development that surprised many observers. BY SWAPNIL BHARTIYA

At DockerCon EU in Copenhagen (Figure 1), Solomon Hykes, the founder of Docker, announced the availability of Kubernetes as an additional orchestrator along with Swarm, the company's own orchestrator.

The news took many by surprise. Why would Docker support an orchestration platform that competes with its own in-house tool? A closer look at this question reveals some insights into the direction of Docker as a company – and it also offers a fascinating window into the innerworkings of open source communities.

A Little History

When Docker hit version 1.0 in 2014, it became an instant hit with the developer community. Developers were able to develop, test, and deploy the same code in production that they ran on their laptops. CIOs and IT professionals were able to migrate applications from virtual machines to containers. New features and functionalities could be packaged inside easy-to-use containers and deployed immediately. They were able to innovate faster. They were able to stay ahead of their competitors. What was there to not to like in Docker containers? Businesses started to flock to containers.

But with big adoption comes big responsibilities. This remarkable growth meant an ever-increasing user base and partner ecosystem. It meant more competitors. Docker faced growing pressure – from partners, customers, and competitors – to loosen its grip. Companies like CoreOS were not happy with things like run time and image format. CoreOS, in fact, came out with its own competing run time, called rkt, to address those concerns. There were talks of forking the Docker Engine, so companies could fix these problems.

Docker has been a champion of open source since the beginning, and it has been open sourcing pieces of Docker Engine since its early days. But the explosive adoption of Docker containers demanded more from the company. Container stakeholders also wanted standardization around two core components of Docker containers: the container run time and container image format. Under the umbrella of the Linux Foundation, a new Collaborative Project, named Open Container Initiative (OCI), was formed. The primary goal of OCI was to finalize the container run-time specification and container image format specification.

Over time, Docker unbundled more components from the Docker engine and released them as independent open source projects. These components included runc, containerd, SwarmKit, libnetwork, Notary, HyperKit, VPNKit, DataKit, and InfraKit.

Breaking the monolith of the Docker engine into smaller pieces made Docker modular and scalable, but it also posed new challenges for the company.

Figure 1: The Docker faithful gathered at Copenhagen's Bella Center for this year's DockerCon EU.

Now Docker engineers had to gather these components and assemble them to build Docker.

Players were coming out with new use cases, expecting Docker to support their platforms. Docker ended up creating Docker for Windows, Docker for Mac OS, Docker for Cloud, Docker for different Linux distributions, Docker for Azure, Docker for AWS, and more. Every time Docker set out to build a specialized edition of Docker using open sourced components, its own production model started to fail. It experienced redundancies and wasted efforts.

To solve this problem, Docker created an assembly line within the company that allowed different teams to collaborate with each other, and with the ecosystem, to use these open sourced components. This approach cut down redundancy and made the entire process more efficient.

Docker decided to release this assembly line – the processes, tooling, components, and framework – as open source so the entire ecosystem could benefit from it. Docker called it the Moby Project. Moby is a framework that allows anyone to assemble their own container system using different components that were open sourced by Docker.

There was one more missing piece.

A Linux subsystem is a prerequisite for Docker, and it's not available in every environment that Docker supports. Docker had already done a lot of work with companies like Microsoft to add a subsystem so that Docker could run on different platforms. Docker released that work as open source and called it LinuxKit. It's a container-native toolkit that enables developers to create their own containerized operating systems that are secure, modular, and portable.

With LinuxKit and the Moby Project, Docker had achieved a Red Hat-like nirvana. The Moby Project was to Docker what Fedora was to Red Hat Enterprise Linux (RHEL). Moby was hosted on GitHub and became the upstream for Docker Editions, the same way Fedora is upstream for RHEL.

Docker Swarms Around Kubernetes

Docker was aware of Kubernetes' growing popularity as a great orchestration tool. Many Docker customers were toying with the idea of using Kubernetes to manage containers.

Why not give them what they want?

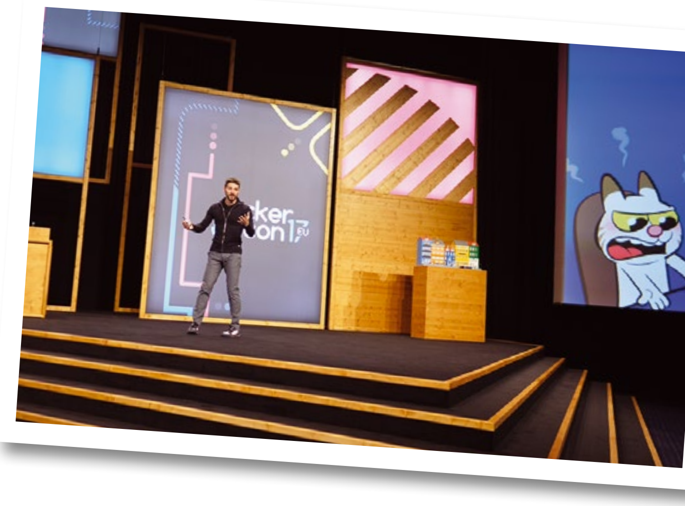
I saw many surprised looks at DockerCon from Docker users who criticized the company for being secretive about this Kubernetes support. "That's not how open source works; you don't keep things secret. Open source means you start talking about it from day zero," critics whispered.

But according to Patrick Chanezon, a member of the technical staff at Docker, "Moby is where all the action happens. All of the Kubernetes work was done in Moby; it's been going on for months. Some people may be surprised because they don't spend all of their time in our GitHub repos. What we announced here is the productization of that work."

Hykes told me that open source is not one-way traffic. If you want to do it right, you need to get engaged at the upstream level. That's exactly what Docker did when they decided to embrace Kubernetes. And Hykes stressed that you can't takes bits and pieces from an upstream project, use it in your product, and then try to keep up with the upstream. "When we realized that we had to support more than one orchestrator, we decided to do it the Docker way. It's a full-on upstream collaboration. We have been working on it for a while, and there is a very high level of engagement. It's a two-way upstream engagement with the Kubernetes community. I guess the Kubernetes community will get as much out of this engagement, in terms of code and operational hardening, as the rest of the Docker community."

Commoditization of Orchestrators

Despite all this excitement around orchestrators like Kubernetes and Swarm, Hykes said that eventually orchestrators will become commodities. When Docker was announced four years ago, everyone was excited about the container run time. Today we have a set of specifications from OCI for container run time and image format. It's commoditized.



"I think the same thing will happen with orchestration. Right now we have a lot of innovation and differentiation, but over a long period, there is going to be commoditization and people are going to converge to one of the two systems," said Hykes.

Two competing technologies being used by competitors is a norm in any industry. Eventually, one of the two technologies will fade out. What's unique about this situation is that Docker is embracing two competing technologies in its own product. Which will be the last orchestrator running?

During DockerCon, Docker engineers gave a demo displaying both Swarm and Kubernetes in the Universal Control Plane of Docker, showing how you can use either of the two to orchestrate your containers. A potential Docker customer may wonder which one to use – Swarm or Kubernetes? The short answer is whichever suits your needs. Swarm is an in-house project of Docker, whereas Kubernetes is a Google-led, community driven project hosted by CNCF. The projects have different goals and a different release cadence.

The broad scope and developer base of the Kubernetes project raises the question of how Swarm could ever keep up. But according to Docker CEO Steve King, Swarm is not going away, assuring users that Docker will support the project for many years to come.

Docker needs to continue to invest in Swarm because it is the integrated orchestrator for the Docker Enterprise Edition. According to Chanezon, "The people who work on the open source project behind Swarm, called SwarmKit, have a very narrow focus: how to provide orchestration in the Docker platform. Because of that narrow focus, they can move faster than Kubernetes, where you have to work with all the companies who have invested in it to solve their own use cases. There is only one use case for us."

Swarm is actually a bit ahead of Kubernetes in terms of more advanced features needed by Docker customers. It's the Kubernetes community that then tries to catch up with those features. Chanezon gave an example of cryptographic node identity that was first introduced in Swarm and later implemented by Kubernetes. "Implementing both Swarm and Kubernetes on the Docker platform will allow us to innovate at the pace we want for our enterprise customers. When those features trickle down to Kubernetes, our customers will be able to leverage both."

On the other hand, Kubernetes has its own benefits for enterprise customers. In an interview, Docker Senior VP Scott Johnson highlighted the additional benefits customers get when they use Kubernetes with Docker Enterprise Edition. "Customers now have the ability to create their secure supply chain while using Kubernetes as an orchestrator for their run-time environment. They get compatibility across their deployment models – whether they use Linux, Windows, or a mainframe. They get benefits of Kubernetes and all the additional capabilities, especially around security, that Docker Enterprise Edition offers."

Orchestrators are the New Linux

Hykes feels that, over time, Swarm and Kubernetes will converge a lot. "It's the same engineers. They are watching each other's work and taking each other's ideas. That's how open source is supposed to work. Orchestration, over time, will become a commodity, which means both Swarm and Kubernetes will become commodities. That's OK. Linux is a great commodity." ■■■

redhat
CONTAINERS FOR THE ENTERPRI
FROM THE LEADER IN OPEN SOU
COMICS & COFFEE
Create your own comic. And caffeinate.

Building Frames

FFmpeg is good not only for converting and fusing videos together, it can also generate streams on the fly, which you can then use for compositing and effects.

BY PAUL BROWN

The casual user might only use the FFmpeg multimedia framework for converting from one audio or video format to another, but FFmpeg can do much more than that. I'll take a look at one of FFmpeg's most powerful secret weapons: the `lavfi` virtual device.

Using lavfi

FFmpeg's `lavfi` (short for *libavfilter*) virtual device sounds more complicated than it really is. Instead of using prerecorded video or audio files as streams, `lavfi` lets you create streams out of thin air. You can use these streams on the fly and combine them with clips and other dynamically generated streams (e.g., from a webcam or microphone) to create your output file.

Maybe it is better explained with an example:

```
ffmpeg -f lavfi ⮒
  -i color=c=red:size=640x480:rate=30 ⮒
  -t 10 red.webm
```

Most basic `ffmpeg` instructions you have seen probably have at least one input file, but the instruction above has none because "input" is coming from the `lavfi` virtual device, not from a file. The content the virtual device generates is described by what goes after the `-i` parameter, that is, `color=c=red:size=640x480:rate=30`. Broken up, `color` tells FFmpeg what sort of stream it should expect – in this case, a simple color video stream. After the first equals sign comes the `color` parameters:

- `c` sets the color of the frame; you can browse a complete list of colors that FFmpeg understands [1] and use one of those (e.g., `red` here), or you can use the *#RRGGBB[AA]* notation.
- `size` establishes the frame size.
- `rate` sets the frames per second.

The `-t 10` parameter makes the resulting clip 10 seconds long.

Making a 10-second video of a flat color is probably the simplest thing you can do with `lavfi`, but making a video of a test pattern is also easy:

```
ffmpeg -f lavfi -i testsrc=s=640x480 ⮒
       -t 10 testscr.webm
```

Combining `lavfi`-generated streams with other streams is what makes it really powerful. As you can see in Listing 1, `lavfi` streams can be used the same way you use regular streams, as described in last month's issue [2]. In this case, you are blending a generated test card stream with a clip taken from the *Big Buck Bunny* [3] movie (Figure 1). If you would like to see what it will look like before rendering a new video file, check out the "Going Live" box and the `ffplay` utility.

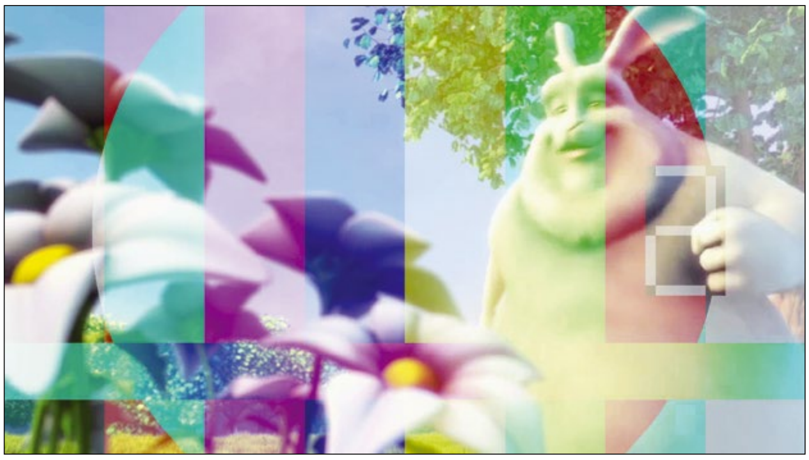


Figure 1: You can combine lavfi-generated streams to achieve all sorts of interesting effects.

Meme Factory

All of the above is well and good, but `lavfi`-generated streams do have some real-world applications, such as in animated GIF building.

GIFs are a really old and not very efficient graphical format, but they have become very popular in the last years. Twitter, Facebook, WhatsApp, and Telegram all implement ways for users to convey their deepest thoughts using animated clips of celebrity facepalms and cats falling off of things.

However, not all GIFs are equal; in fact, it is easy to distinguish a "professional" GIF artisan from a

Listing 1: Superimposing a Generated Stream

```
ffmpeg -i BBB_flowers.mp4 -f lavfi -i testsrc=s=1280x720[out] -t 5 -filter_complex "[1:v]setsar=sar=1[red];[0:v]setsar=sar=1,
       format=rgba[bg];[bg][red]blend=all_mode=addition:all_opacity=0.5,format=yuva422p10le" -codec:a copy testscr.mp4
```

Going Live

FFmpeg comes with a simple media player that allows you to preview the results of your experiments live. Substitute `ffplay` in place of `ffmpeg`, and the result will be output directly to a window on your desktop instead of a file – a great way to check that you are achieving the effect you want to achieve without waiting for the whole clip to be rendered.

For example, to play the clip of red frames as it is generated, you can run:

```
ffplay -f lavfi ⮐
  -i "color=c=red:size=640x480:rate=30"
```

To stop the script, press Ctrl+C in the terminal window in which you are running FFmpeg.

To show the test card, run:

```
ffplay -f lavfi -i "testsrc=s=640x480"
```

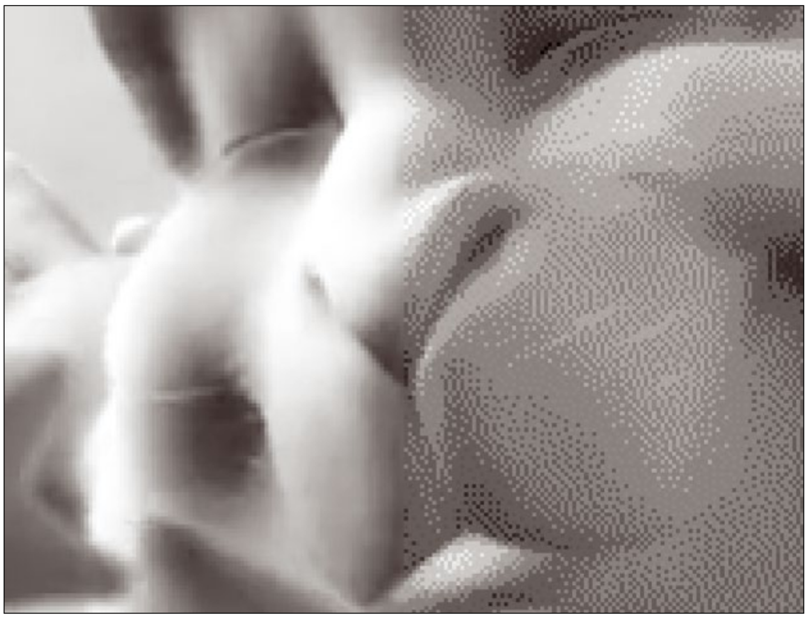


Figure 2: Smooth pixels make a high-quality GIF animation.

wannabe artist by the caliber of their pixels. Let me explain with Figure 2.

You are looking at a compound image of two different GIFs, each generated using a different technique. On the left, the image looks smooth, with a gradual gradient between each shade of gray. On the right, you can clearly see the pixels, and the transition between areas with different shades of gray is abrupt and jagged. Believe it or not, both are the exact same resolution, and the smooth, good-looking version on the left weighs a whole 2MB less than the one on the right.

To understand why this happens, you must look at how GIF uses palettes. GIF images use a palette of only 256 colors, which is not enough to cover the whole spectrum of a full-color video and make it look good; at least, it is not good enough by today's standards. The easiest and laziest way of generating a GIF from a clip is to enter:

```
ffmpeg -i clip.mp4 image.gif
```

However, this is not optimum by a long shot. All of the colors in all of the frames in your clip must be

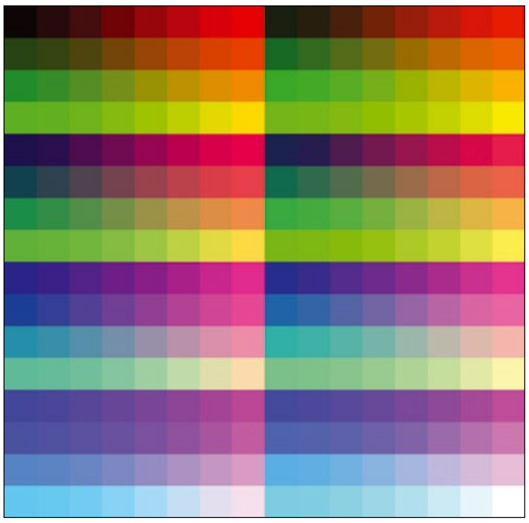


Figure 3: The default FFmpeg palette used when generating GIFs.

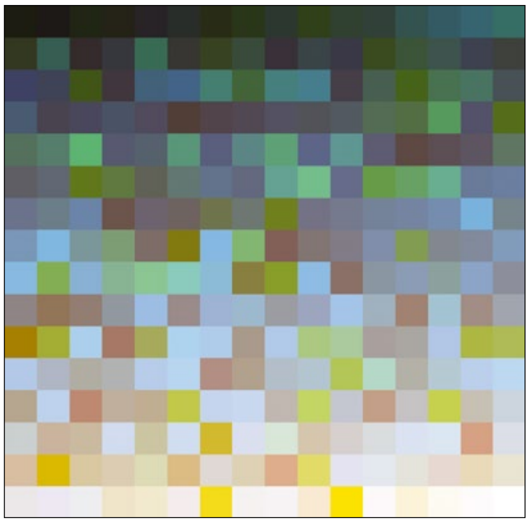


Figure 4: A palette taken from a five-second clip lifted out of *Big Buck Bunny*.

matched to one of the colors in the default palette used by FFmpeg (Figure 3). Not surprisingly, the final result can look grainy.

However, you can create a palette that draws its colors from the clip you are converting itself:

```
ffmpeg -i clip.mp4 ↩
      -filter_complex "fps=15,scale=320:-1:↩
       flags=lanczos,palettegen" ↩
       clip.palette.png
```

Here, FFmpeg creates a palette called `clip.palette.png` (Figure 4) with the colors from `clip.mp4`. Notice how the colors are softer and more pastel than those of the default palette and how they match the general palette of the clip much more closely.

Once you have your custom palette, you can then use it to generate your GIF:

```
ffmpeg -i clip.mp4 -i clip.palette.png ↩
  -filter_complex "paletteuse,fps=10" clip.gif
```

Apart from using the palette, you are also reducing the frame rate (`fps=10`) to make a smaller, lighter clip.

If your clip still doesn't look very good, remember you still only have 256 colors. The palette in Figure 4 does not look like a uniform gradient, which is a sure sign your GIF will probably look splotchy. If your original clip has a large variety of colors (e.g., with frames shot at night and bright, colorful frames shot during the day), your custom palette will be more uneven and your GIF will look grainier.

Filters and algorithms let you compensate for this effect, but none are simple, nor do they tend to improve the quality that much. Instead, "professional" GIF creators have come up with something else.

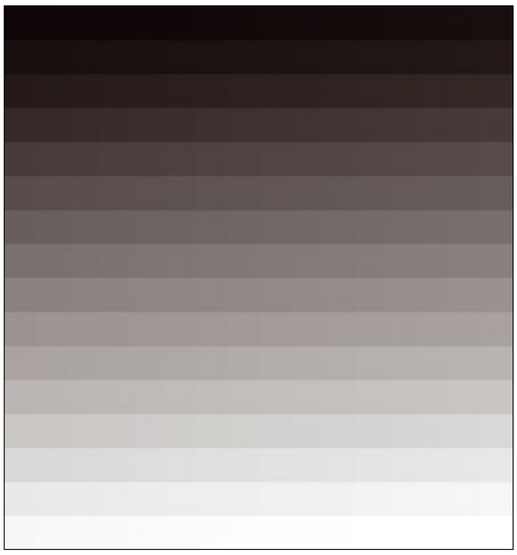


Figure 5: A grayscale palette generated from a black-and-white clip taken from *Big Buck Bunny*.

256 Shades of Gray

You might have noticed that many GIFs are rendered in shades of gray. This is because it is easier to get smooth transitions from one shade of gray to the next with a palette of 256 grays than with a palette trying to cover a wider array of colors.

To prepare a clip for the "GIFification" process, you need to reduce it to shades of gray:

```
ffmpeg -i clip.mp4 ↩
  -filter_complex "format=gray,scale=640:-1" ↩
  clip_gray.mp4
```

It's important to notice how you should resize your clip to the size you want your GIF to be (`scale=640:-1`). If you mess with the size when converting to GIF, it will mess with your palette.

By converting your clip to gray, you would seemingly have made matters worse. The default palette in Figure 3, has very little in the way of gray. Instead of being able to pick from 256 colors, if FFmpeg were to use the default palette, it would have to choose from 10 or 15 colors, making the splotchiness even worse than if you left the clip in full color.

However, if you generate a palette with the following,

```
ffmpeg -i clip_gray.mp4 ↩
      -filter_complex "fps=15,scale=320:-1:↩
       flags=lanczos,palettegen" ↩
       clip_gray.palette.png
```

you'll see something like Figure 5.

If you use this palette to generate your GIF,

```
ffmpeg -i clip_gray.mp4 ↩
      -i clip_gray.palette.png ↩
      -filter_complex "paletteuse,fps=10" ↩
      clip_gray.gif
```

open `clip_gray.gif` in an image visualizer and zoom, you will notice how the transitions between colors are much smoother.

Go with lavfi

Generating a GIF in two steps – generating the palette and then applying it – is not elegant. Besides, it leaves behind a PNG file that is useless for anything else.

Thanks to the `lavfi` virtual device, you can do everything in one go (Listing 2), which has the advantage of making you look like a veritable FFmpeg wizard, without generating any PNG cruft to boot.

At this stage, it should be clear what is going on, but for the sake of clarity:

Listing 2: One-Step GIF Generation

```
ffmpeg -i clip_gray.mp4 -f lavfi -i "movie=clip_gray.mp4,fps=10,scale=320:-1:flags=lanczos,palettegen[out]"
       -filter_complex "paletteuse,fps=10" clip_gray.gif
```

- The `clip_gray.mp4` file is the grayscale and scaled-down video you want to convert.
- The `lavfi` filter options in quotation marks generate the palette image and pipe it through `[out]` to the …
- … `-filter_complex` chain of filters.

Apart from using the palette you created with the `lavfi` virtual device, you reduce the frame rate to 10 frames per second and push the resulting frames out to `clip_gray.gif`.

The resulting GIF image is smooth, acceptably lightweight, and created in seconds.

Conclusion

Not surprisingly, FFmpeg is the back end for so many video editing applications: The number of things you can do with it is astounding. Although it is true that some of the command lines you have to generate can be eye-wateringly complex, it is often worth working directly with FFmpeg, because it gives you that extra control on the final result. Apart from lavfi, you can learn more about FFmpeg devices by taking a look at the "More Devices" box. ■■■

Info

[1] FFmpeg colors: *https://ffmpeg.org/ffmpeg-all.html#Color*

[2] "Tutorials – FFmpeg" by Paul Brown, *Linux Magazine*, issue 206, January 2018, pg. 92, *http://www.linux-magazine.com/Issues/2018/206/Tutorials-FFmpeg*

[3] *Big Buck Bunny*: *https://peach.blender.org/*

[4] video4linux2 device: *https://www.ffmpeg.org/ffmpeg-devices.html#video4linux2_002c-v4l2*

[5] alsa device: *https://www.ffmpeg.org/ffmpeg-devices.html#alsa*

[6] pulse device: *https://www.ffmpeg.org/ffmpeg-devices.html#pulse*

[7] x11grab device: *https://www.ffmpeg.org/ffmpeg-devices.html#x11grab*

More Devices

Apart from `lavfi`, a few other FFmpeg devices are worth investigating, especially if you want a raw, but very reliable way of grabbing streams from your camera or desktop:

(1) `video4linux2`, or `v4l2` [4] for short (you can use either), captures streams from your webcam. To work out what devices you have on which to capture streams, run `ls /dev/video*`. My laptop has, apart from its built-in webcam, an external webcam with its own microphone, so I see:

```
> ls /dev/video*
/dev/video0 /dev/video1
```

If you run `ffplay -f v4l2 -i /dev/video1`, a window pops up showing the feed from the external webcam.

(2) On a related note, `alsa` [5] and `pulse` [6] capture audio. To know which device to capture audio from, if you want to use `alsa`, enter `arecord -l`. Again, on my laptop, apart from the built-in microphone, I have the microphone attached to the external webcam and a microphone input on an external USB sound card, so I see:

```
> arecord -l
**** List of CAPTURE Hardware Devices ****
card 0: PCH [HDA Intel PCH], ↩
 device 0: ALC3241 Analog [ALC3241 Analog]
 Subdevices: 1/1
 Subdevice #0: subdevice #0
card 1: Device [USB Sound Device], ↩
 device 0: USB Audio [USB Audio]
 Subdevices: 1/1
 Subdevice #0: subdevice #0
card 2: H2300 [HP Webcam HD 2300], ↩
 device 0: USB Audio [USB Audio]
 Subdevices: 1/1
 Subdevice #0: subdevice #0
```

If I want to record from the microphone on the webcam, I can see it is *card 2* (*HP Webcam HD 2300*) connected to *device 0*. This means `ffmpeg -f alsa -i hw:2,0 voice.mp3` records to a file called `voice.mp3`.

With `pulse`, things are a bit different. First, you need to find out the *name* of your devices:

```
> pactl list sources
Source #2
  State: SUSPENDED
  Name: alsa_input.usb-Hewlett_Packard_HP_Webcam_HD_↩
   2300-02.analog-stereo
  Description: HP Webcam HD 2300 Analog Stereo
  Driver: module-alsa-card.c
  Sample Specification: s16le 2ch 44100Hz
  Channel Map: front-left,front-right
  Owner Module: 7
  Mute: no
  Volume: front-left: 48497 /  74% / -7.85 dB,   ↩
    front-right: 48497 /  74% / -7.85 dB
    balance 0.00
  Base Volume: 41350 /  63% / -12.00 dB
  Monitor of Sink: n/a
  Latency: 0 usec, configured 0 usec
  Flags: HARDWARE HW_MUTE_CTRL HW_VOLUME_CTRL DECIBEL_↩
   VOLUME LATENCY
[...]
```

Once you have located the device from which you want to record, you can then use its name in FFmpeg:

```
ffmpeg -f pulse -i alsa_input.usb-Hewlett_Packard_HP_↩
   Webcam_HD_2300-02.analog-stereo voice.mp3
```

In this case, it captures from an external webcam microphone and records to `voice.mp3`.

(3) `x11grab` [7] is interesting if you want to make screencasts. You can grab the whole screen, just a rectangle, or a rectangle centered around the cursor. For example, the line

```
ffmpeg -f x11grab -follow_mouse centered -framerate 25 ↩
   -video_size 640x480 -i :0.0 screncast.mp4
```

captures a `640x480` region centered around the mouse and records it to `screencast.mp4`.

You can mix and match all the above devices and apply all sorts of effects to the streams they produce. Altogether, they provide a very complete and complex screencasting solution.

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*.

Embedded Linux Conference

Date: March 12–14, 2018

Location: Portland, Oregon

Website: *http://events.linuxfoundation.org/events/embedded-linux-conference*

The Embedded Linux Conference is a vendor-neutral technical conference for companies and developers using Linux in embedded products. ELC has a large collection of sessions dedicated exclusively to embedded Linux and embedded Linux developers.

Open Networking Summit

Date: March 26–29, 2018

Location: Los Angeles, California

Website: *http://events.linuxfoundation.org/events/open-networking-summit-north-america*

Join business and technical leaders across enterprise, cloud, and service providers to share information, highlight innovation, and discuss the future of open networking and orchestration.

Linux Storage Filesystem and MM Summit

Date: April 23–25, 2018

Location: Park City, Utah

Website: *https://events.linuxfoundation.org/events/lsfmm-2018/*

Experts map out and implement improvements to the Linux filesystem, storage, and memory management subsystems that will find their way into the mainline kernel and Linux distributions in the next 24 to 48 months.

Events

FAST '18	February 12–15, 2018	Oakland, California	https://www.usenix.org/conference/fast18
Open Source Leadership Summit	March 6–8, 2018	Sonoma Valley, California	http://events.linuxfoundation.org/events/open-source-leadership-summit
SCALE 16x	March 8-11, 2018	Pasadena, California	http://www.socallinuxexpo.org/scale/16x
Chemnitz Linux Days 2018	March 10-11, 2018	Chemnitz, Germany	https://chemnitzer.linux-tage.de/2018/en/
Maker Faire Ruhr	March 10-11, 2018	Ruhr, Germany	https://www.makerfaire-ruhr.com/
CloudFest 2018	March10-16, 2018	Rust, Germany	https://www.cloudfest.com/
Embedded Linux Conference	March 12–14, 2018	Portland, Oregon	http://events.linuxfoundation.org/events/embedded-linux-conference
OpenIoT Summit	March 12–14, 2018	Portland, Oregon	http://events.linuxfoundation.org/events/openiot-summit
NSDI '18	April 9–11, 2018	Renton, Washington	https://www.usenix.org/conference/nsdi18
heise Security Tour 2018	April 10, 12, 18, 24, and 26, 2018	Numerous European cities	https://www.heise-events.de/securitytour
Open Networking Summit	March 26–29, 2018	Los Angeles, California	http://events.linuxfoundation.org/events/open-networking-summit-north-america
2018 HPC for Wall Street	April 16, 2018	New York, New York	http://www.flaggmgmt.com/linux/
Cloud Foundry Summit North America	April 18–20, 2018	Boston, Massachusetts	https://www.cloudfoundry.org/event/nasummit2018/
Linux Presentation Day 2018.1	April 21, 2018	Europe-wide in many cities	http://www.linux-presentation-day.org/
Linux Storage Filesystem and Memory Management Summit	April 23–25, 2018	Park City, Utah	http://events.linuxfoundation.org/events/lsfmm-2018/

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to *edit@linux-magazine.com*.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at: *http://www.linux-magazine.com/contact/write_for_us.*

NOW PRINTED ON recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Authors

Bernhard Bablok	60
Erik Bärwaldt	34, 84
Swapnil Bhartiya	8, 30, 88
Paul Brown	92
Zack Brown	12
Bruce Byfield	52, 56
Joe Casad	3
Mark Crutch	69
Rainer Grimm	22
Karsten Günther	46
Jon “maddog” Hall	72
Lothar Hiller	60
Heike Jurzik	28
Charly Kühnast	50
Vincent Mealing	69
Pete Metcalfe	66
Graham Morrison	74
Simon Phipps	70
Mike Schilli	42
Tim Schürmann	16
Ferdinand Thommes	80

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
Lori White

Cover Image
© maksym yemelyanov, 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.

Approximate

UK / Europe	Feb 05
USA / Canada	Mar 02
Australia	Apr 02

On Sale Date

Issue 208 / March 2018

Free Voice Assistants

Move over Siri and Alexa – these free personal voice assistants will answer your questions without the spying.

Preview Newsletter

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: ***www.linux-magazine.com/newsletter***

Lead Image © David Sandonato, 123RF.com

HETZNER ONLINE

RYZE 'N' SHINE

More parallelisation.
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 £ 100

monthly £ 50

The ideal solution for highly-parallelisable 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 virtualisation, encryption, and data compression, which require several processor cores.

www.hetzner.de/gb

* 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 per additional TB used.

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. All rights reserved by the respective manufacturers.