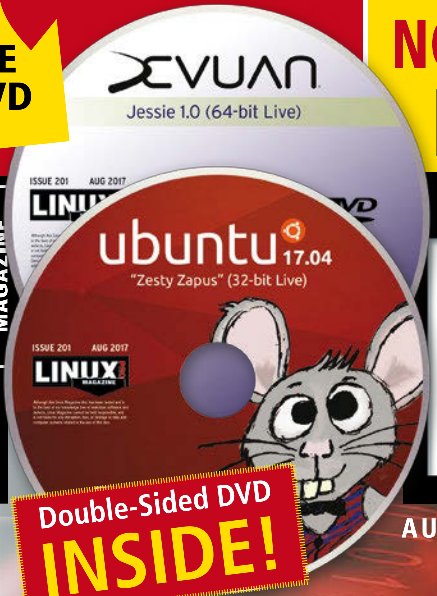
FREE DVD



Double-Sided DVD INSIDE!

NOW INCLUDES LINUXVOICE

LOCK DOWN
NETWORK SECURITY TOOLKIT

LINUX PRO MAGAZINE

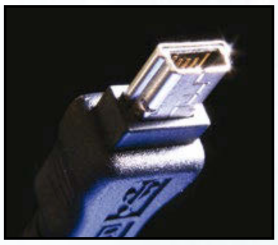
AUGUST 2017

LOCKED DOWN
with the Network Security Toolkit

Journalctl
Getting info from systemd

Flatpak and Snap
Package systems for the container generation

Rewriting PDFs in Linux with Master PDF Editor 4



Testing Flash
Does your 4GB USB stick really hold 4GB?

Advanced Layout
Expert tricks for LibreOffice

Calamares
Universal Linux installer

LINUXVOICE

- MP3 is Dead, Long Live MP3
- Interactive Scripts
- The Saga of Ubuntu
- FAQ: Apache Spark

FOSSPicks
- Google Drive for Plasma
- Audio Shop

Tutorials
- ImageMagick
- Markdown

Issue 201
Aug 2017
US$ 15.99
CAN$ 17.99

WWW.LINUXPROMAGAZINE.COM

HETZNER ONLINE

RYZE 'N' SHINE

More parallelization. More power.

Dedicated Root Server AX60-SSD

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

monthly $ 65

The ideal solution for highly-parallelizable processes.

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

www.hetzner.de/us

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

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.

"VOTING RECORDS"

Dear Reader,

Politics is everywhere this year. It is hard to get away from the news – and it seems that most of the talking is done by people who aren't too interested in finding any kind of common ground with other viewpoints, which makes it all the more excruciating for the ears. I'm well aware that people read Linux magazines to *get away* from politics.

Sometimes, however, politics really is a high-tech topic. A recent news story reports that 198 million Americans were affected by the "largest ever" voting records "leak." The analytics data was stored on an unsecured Amazon server. According to reports, the server was apparently owned by Deep Root Analytics, a consulting firm specializing in targeted, analytics-based outreach on behalf of pro-trade Republican candidates, although the Republicans were apparently following the lead of previous efforts by Democrats to bring Big Data techniques to voter data analysis.

The insecure server was discovered by cyber-security expert Chris Vickery, who reported the problem to the owner and allowed Deep Root to fix the issue before making the information public. As far as anyone knows, the data was not discovered by foreign powers or nefarious actors before it was locked back down again. The story is getting a lot of traction in the security news, and many are talking about it as a "wake up call" for better security, which it certainly is. I'm not questioning the need for better security on cloud-based servers, but I'm wondering if *security* is truly the best category for this report.

My colleagues in the press use the term "voter records" to describe this information on the Deep Root server, but what is it really? Who owns it? And who are the rightful parties who are supposed to be able to access it? It appears the information consists of easily accessible voter data available through state, county, and local government offices, mapped to proprietary information compiled through market research firms. The government voter data could be a list of who is registered or who voted in the last election. The proprietary part of the data might be a set of attributes associated with each voter predicting how the voter will behave. For instance, if you shop at a certain store or read a particular blog that might be useful in predicting how likely you are to change your vote, the goal of this kind of database is to capture that information and use it to direct very granular, personalized advertising at you to sway your opinions. This data is compiled very much like consumer data is compiled throughout the Internet, and as we all know by now, the whole Internet is powered by consumer data acquisition.

So before you say, "Whoa, I'm sure glad no foreign hackers got hold of those 'unsecured voter records'," take a moment to consider that a foreign hacker who wanted all this data could get it any time just by hiring a market research company to go out and get it for them. If you are scared about this kind of information "falling into the wrong hands," it would probably be a good idea to ask what would be the *right* hands for this kind of information, and, once it exists, how you could reasonably hope to control who will get to see it.

So no matter what they say on the forums and social media, this story isn't really about security. It is about the very structure of the Internet.

Joe

Joe Casad,
Editor in Chief

LINUX MAGAZINE

WHAT'S INSIDE

This month we show you the Network Security Toolkit, a smart collection of powerful security tools bundled into a single interface. Other highlights include:

- **I-Nex** – a graphical tool for monitoring system information (page 56).
- **Calamares** – universal Linux installer (page 48).

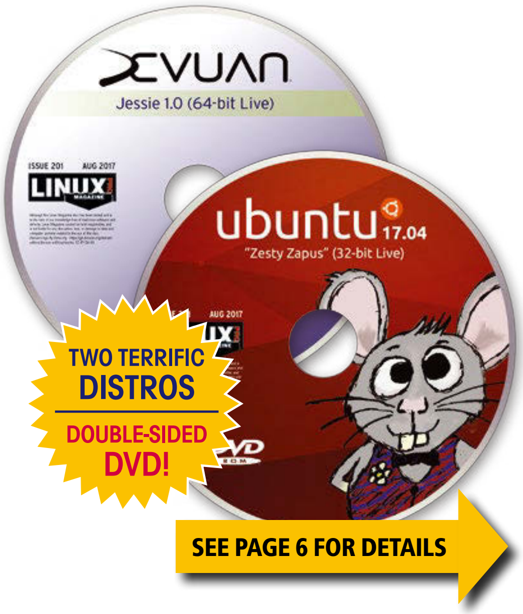
Check out Linux Voice for a look at interactive scripts, a brief history of Ubuntu, and a tutorial on the ImageMagick photo processing utility (page 69).

SERVICE

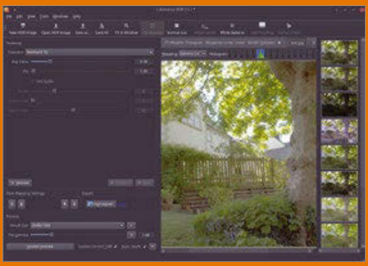
NEWS

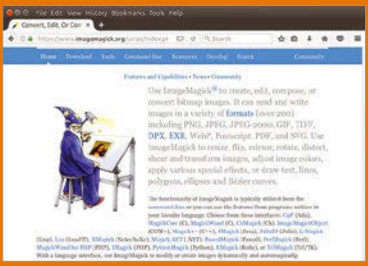
COVER STORY

IN-DEPTH

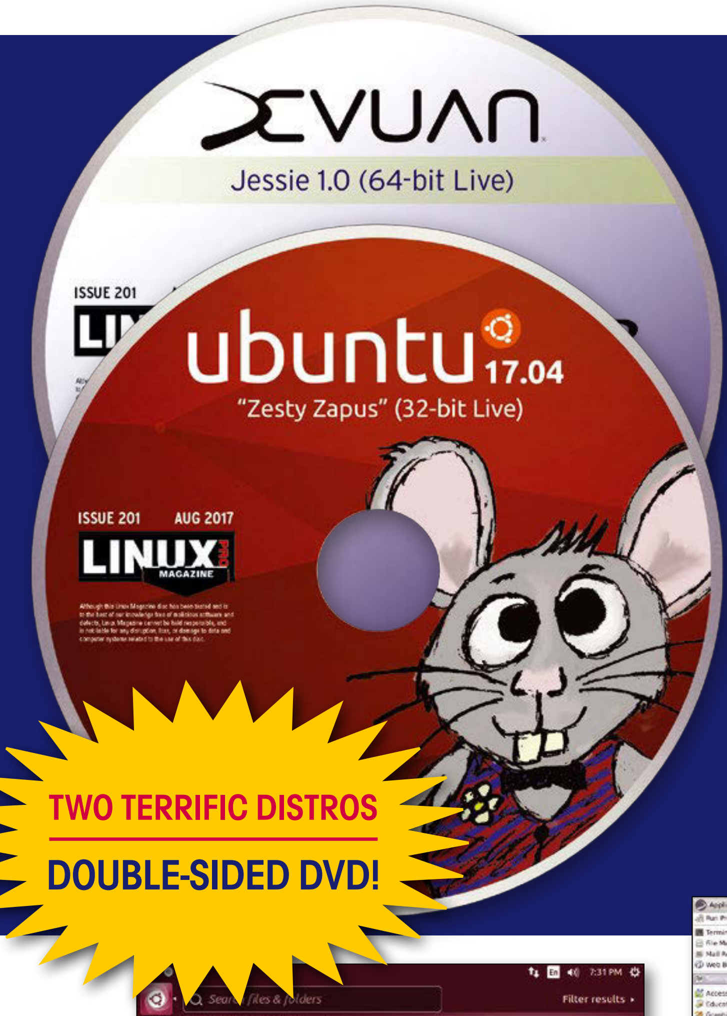


LINUXVOICE





On the DVD



Ubuntu 17.04 (32-bit Live)

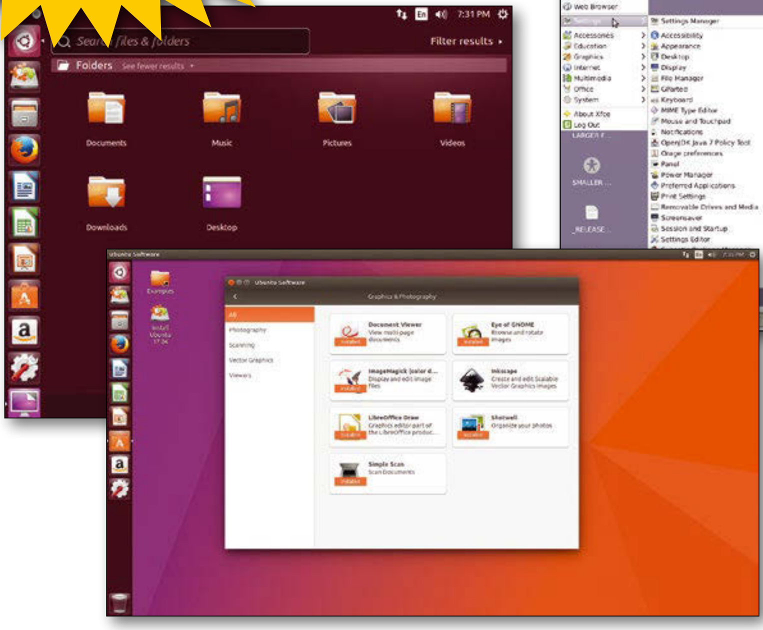
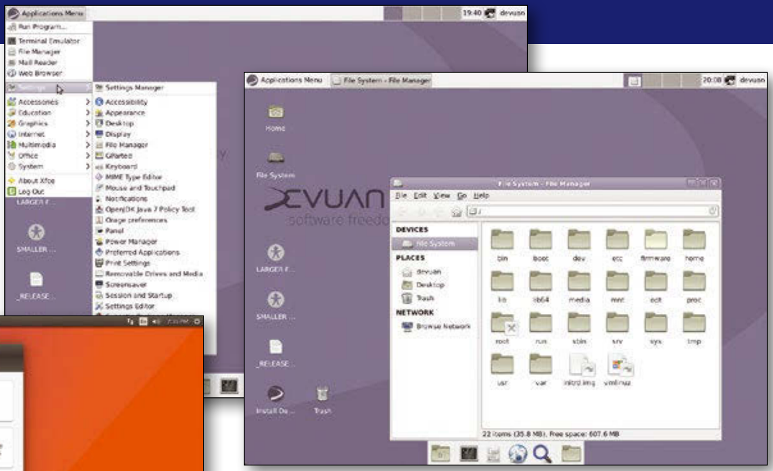
"Zesty Zapus" is the latest release of this popular Debian-based distro [1]. Changes and improvements include:

- a new default DNS resolver tailored for systemd
- the use of swap files instead of swap partitions
- driverless print support
- a week view in the Calendar app
- LibreOffice updated to 5.3

Zesty is the last version of Ubuntu that will include the Unity desktop in the default configuration. Ubuntu 17.04 is supported until January 2018. The latest long-term support version is 16.04 LTS [2].

Devuan 1.0 (64-bit Live)

Devuan GNU+Linux [3] is a fork of Debian that diverged from the main project in a dispute over Debian's adoption of the systemd init daemon. Devuan continues to use the classic SysVinit. The developers state that "Devuan Jessie provides continuity as a safe upgrade path from Debian 7 (Wheezy) and a flawless switch from Debian 8 (Jessie)." Devuan is intended as a base distribution that will form the foundation for derivatives that target specific usage scenarios. The Devuan project maintains package repositories distinct from Debian. Both Debian and Devuan documentation serve as sources of help for Devuan users [4].

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

ADDITIONAL RESOURCES

[1] Ubuntu desktop: *https://www.ubuntu.com/desktop*

[2] Ubuntu wiki: *https://wiki.ubuntu.com/ZestyZapus/ReleaseNotes*

[3] Devuan: *https://devuan.org*

[4] Doc Devuan: *https://git.devuan.org/groups/devuan-doc*

open build service

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

openbuildservice.org

openSUSE®

NEWS

Updates on technologies, trends, and tools

Trojan Turns Raspberry Pi into a Cryptocurrency Mining Device

The Russian security firm Doctor Web has discovered two Trojan programs that target Linux machines. One turns Raspberry Pi machines into a cryptocurrency mining device, and the other runs a proxy server on Linux systems.

The Trojan named Linux.MulDrop.14 targets Raspberry Pi devices, changing the password on the devices it infects, then unpacking and launching a miner, which, in an infinite loop, starts searching for network nodes with an open port 22 to replicate itself.

According to Doctor Web, "The Trojan is a script that contains a compressed and encrypted application designed to mine cryptocurrency."

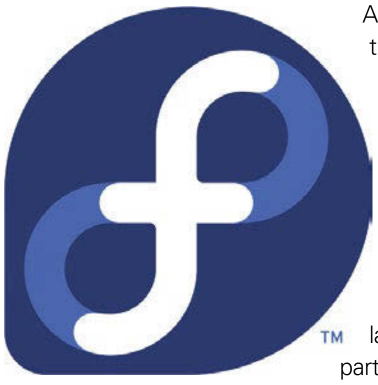
The second Trojan, dubbed Linux.ProxyM, uses a special range of methods to detect honeypots – special decoy servers used by digital security specialists to examine malicious software.

"Once launched, it connects to its command and control server and, after getting confirmation from it, runs a SOCKS proxy server on the infected device. Cybercriminals can use this Trojan to ensure that they remain anonymous online," noted Doctor Web.

Fedora 26 Beta Comes with New Features

The Fedora project has announced the beta of Fedora 26, the latest version of Fedora OS. Three editions of Fedora target three different markets: Fedora 26 Workstation Beta, Fedora 26 Server, and Fedora 26 Atomic.

As the name implies, Workstation targets desktop users, the Server edition is aimed at sysadmins running servers, and Atomic targets DevOps, for managing cloud- and container-centric infrastructures.



Although each edition targets a different market segment, they all share the same fundamental Fedora technologies, and the only differences are in what comes packaged with each edition.

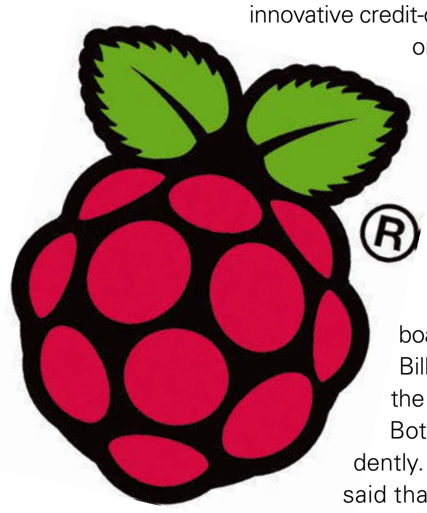
All three versions share these new components: updated compilers and languages, including GNU Compiler Collection (GCC) 7; Go 1.8; Python 3.6 and Ruby 2.4; DNF 2.0 with backward compatibility with Yum; a new storage configuration screen for the Anaconda installation program, enabling "bottom-up" configuration from devices and partitions; and updates to Fedora Media Writer that enable users to create bootable SD cards with Fedora for ARM-based devices, like Raspberry Pi.

New in the desktop edition is Gnome 3.24, which offers many new features, including batch rename of files and night mode. Some highlights of this release include many improvements to Builder – to support a number of application build systems, including Flatpak, CMake, Meson, and Rust – in addition to integration with Valgrind for project profiling.

As containers become more and more important, Fedora Atomic offers a great platform for running container-based workloads in the cloud or on bare metal. One of the most notable features of Fedora Atomic Host is containerized Kubernetes to run different versions of the container orchestration engine.

All three editions are available for download and testing.

Raspberry Pi Foundation Merges with CoderDojo Foundation



Two open source organizations, the Raspberry Pi Foundation and CoderDojo Foundation, are joining forces. Whereas the Raspberry Pi Foundation is known for their innovative credit-card-sized single-board computers, CoderDojo focuses on exposing young people to computer programming.

According to Philip Colligan, CEO of Raspberry Pi Foundation, "Bringing together Raspberry Pi, Code Club, and CoderDojo will create the largest global effort to get young people involved in computing and digital making."

It's not a simple merger. The Raspberry Pi Foundation will become a corporate member of the CoderDojo Foundation, and Colligan will join the CoderDojo board as a director. In return, co-founders of CoderDojo, Bill Liao and James Whelton, will become members of the Raspberry Pi Foundation.

Both organizations plan to continue to operate independently. Giustina Mizzoni, Executive Director of CoderDoJo, said that it would remain an independent charity based in Ireland. "In practical terms, this merger will see our two organizations working closely together to advance our shared goals," said Mizzoni. "It will enable us to leverage assets and capabilities ultimately driving further value for the CoderDojo Community."

The Raspberry Pi Foundation will provide practical, financial, and back office support to the CoderDojo Foundation.

"With this extra support we will be able to reach and benefit even more young people globally by investing more time in resource development, community support and growth strategies to make it easier for our volunteers to start and keep running a Dojo in their community," said Mizzoni.

The merger doesn't imply that Raspberry Pi will become the exclusive platform for CoderDojo. As always, CoderDojo will remain software and hardware neutral.

Samba Vulnerability Patched But Risk Is Bigger

The world barely recovered from the havoc caused by WannaCry ransomware before a new vulnerability was found in the open source Samba networking utility.

According to Samba.org, "All versions of Samba from 3.5.0 onwards are vulnerable to a remote code execution vulnerability, allowing a malicious client to upload a shared library to a writable share, and then cause the server to load and execute it."

In pure open source tradition, the patch was released immediately, and most Linux distributions have pushed it into their repository.

The real-world situation is more grim than it appears. First, it's not a new bug. The bug has been lurking around for the past seven years, since version 3.5.0 was released in 2010. It exposes a serious problem in the Linux world: It doesn't have enough eyeballs to make all bugs shallow.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

SquashFS • Jeff Layton
In my life experience, I have found that people like to keep pretty much every piece of data that's ever crossed their hard drive.

Parallel I/O for HPC • Jeff Layton
Amdahl's law says that your application will only go as fast as its serial portion. As the application is run over more processors, the decrease in run time gets smaller.

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

Highly Available Hyper-V in Windows Server 2016 • Marc Grote
Most of the new features in Windows Server 2016 relate to Hyper-V. Microsoft has introduced numerous changes to make the product even more interesting to companies that have not used virtualization or are running an older version of Hyper-V.

Digital Signatures in Package Management Tim Schürmann
Many distributions develop, test, build, and distribute their software via a heterogeneous zoo of servers, mirrors, and workstations that make central management and protection of the end product almost impossible.

Installing .NET on Linux • Thorsten Scherf
To understand .NET fully, it is a good idea to look at past events. Development on the NGWS included work on a framework that was officially released in 2002 as .NET 1.0.

The second problem that makes this bug more problematic is that the open source reimplementation of Microsoft's SMB protocol, which was the culprit in the WannaCry ransomware, is used in every single product that offers any kind of file-sharing capability.

If you have a NAS device, media streaming box, or any device that offers file storage and sharing capability, then it's more than likely running Samba server on it. Despite running a Linux-based distribution, these devices are not designed for automatic updates and don't offer users an easy interface to update the packages.

At the same time, in most cases, vendors have no incentive to keep the devices patched, which leaves them vulnerable. If you are aware of this bug and you are running one of these devices, there is literally nothing you can do to fix it, other than unplugging it from the server. The best course of action is to keep an eye on the support site of the product and look for any updates. If updates are available, install them immediately.

Red Hat Announces OpenShift.io

Red Hat has created a cloud native developer tool called OpenShift.io, announced at Red Hat Summit, Boston.

The platform is based on Kubernetes, a Linux Foundation-hosted open source project. Built from Eclipse Che, fabric8, and Jenkins technologies, OpenShift.io provides developers with application development tools and the environments they need.

According to Red Hat, "OpenShift.io, combined with OpenShift Online, provides an integrated approach to DevOps, including all the tools a team needs to analyze, plan, create and deploy services."

The platform was created for team collaboration and offers real-time stack analysis, which helps development teams better detect critical vulnerabilities and uncommon usage patterns.

OpenShift.io enables developers to use the entire platform without a requirement to install anything locally, and their applications are built into Linux containers by default.

OpenShift.io also includes a free subscription to the Red Hat Developer Program, a no-cost Red Hat Enterprise Linux developer subscription, Red Hat JBoss Enterprise Middleware, and other Red Hat technologies. OpenShift.io is available in a limited developer preview.

Microsoft Bakes Linux into Windows Server

Microsoft is graduating to become a Linux vendor. It started with Microsoft introducing WSL (Windows Subsystem for Linux) for Windows 10, which was the company's attempt to help developers using Windows 10 manage their Linux machines on Azure cloud.

The company then worked with Docker not only to create Docker for Windows, but also to bring Docker containers to Linux servers, allowing customers to run more than 900,000 Linux containers on Windows Servers.

Now Microsoft is baking WSL into Windows Server. According to a Microsoft blog, "This unique combination allows developers and application administrators to use the same scripts, tools, procedures and container images they have been using for Linux containers on their Windows Server container host."

With Bash on Ubuntu for Windows Servers, IT professionals can now use *nix utilities on their Windows servers to manage Linux containers.

With this move, Microsoft is moving closer toward becoming a Linux provider. It must be noted that Microsoft already uses Linux as a core piece in its Azure cloud. The operating system for Azure Networking Switch runs on a Linux kernel.

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

NORTH AMERICA

SEPTEMBER 11 - 14, 2017 | LOS ANGELES, CA

LinuxCon, ContainerCon, CloudOpen and the new Open Community Conference combine under one umbrella name in 2017 - the Open Source Summit.

At Open Source Summit, you will collaborate, share information and learn across a wide variety of topics, with 2,000 technologists and community members.

Co-located events include Linux Plumbers Conference, Linux Security Summit, MesosCon and the all new Diversity Empowerment Summit.

LINUX PRO READERS SAVE 15% ON ATTENDEE REGISTRATION WITH CODE LNM15.

go.linuxfoundation.org/ossnalnm

Zack's Kernel News

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

By Zack Brown

Tracking Inode Versioning

Just as the double slit experiment behaves one way when it's being watched, and another when it thinks it can get away with something, Jeff Layton suggested that Linux could get away with incrementing the inode version number only when some other part of the kernel was actually paying attention.

The inode version number (`i_version` in the inode structure) is generally incremented whenever the inode's metadata changes. Or, as Jeff explained, filesystems increment the version number whenever something happens that they care about. In the case of Btrfs, ext4, and XFS, they increment the version for file data changes as well. The problem with too much version change is that the inode has to be written back to disk each time, which can be slow on write-heavy operations.

Jeff believed he'd discovered that the only time the version number really mattered was when some other bit of code queried the inode. If there was no observer, there would be no need to keep the inode version precisely up to date. And, when an observer came along, that would be a good time for the inode to show it had been paying attention, by updating its version number.

So, Jeff posted a patch that would update inode version numbers in those filesystems if both a change and a query had occurred. He asked if folks thought this was the right way to go – and he predicted that eventually other kernel changes would make something like this necessary in any case.

Christoph Hellwig asked for some performance numbers to show that Jeff's patch made a real improvement. Jeff offered up some numbers showing a twofold speed improvement, though he also pointed out that the numbers would vary depending on workload.

Bruce Fields also had some comments but was enthusiastic about the whole idea. His main concern was making sure versioning worked properly for all filesystems. At one point, he wrote up notes for an `i_version` man page, describing the semantics as he understood them:

- It would be a 64-bit number, which was big enough that there should be no question of rolling over when the version got too high, although those tend to be famous last words.
- It works at least for files and directories and possibly for symlinks and other special file types.
- It increments between separate queries if and only if the data or metadata changes between the two queries.
- It works even if a reboot occurs between two version queries.

Jeff liked Bruce's write-up and said it covered the semantic issues very nicely. But then he and Bruce went back and forth for a bit on the feasibility of `i_version` working across reboots. The problem was, how would the system behave if the system crashed in between the time the version number increased and the relevant data made it onto the drive? If they couldn't guarantee consistency across reboots, then what guarantees could they reasonably make?

Jeff and Bruce agreed it was a real problem and needed to be addressed to avoid data inconsistency. At one point, Jeff said, "I think we'll have a hard time making this fully atomic. We may end up having to settle for something less (and doing our best to warn users of that possibility)."

But he also said that his `i_version` patch didn't make any atomicity issues worse than they already were in the kernel. In particular, as Dave Chinner had pointed out, NFS had always had a consistency issue between client and server. And Jan Kara also pointed out, "similarly as with XFS, ext4 doesn't guarantee atomicity unless `fsync()` has completed on the file. Until that you can see arbitrary combination of data & `i_version` after the crash. We do take care to keep data and metadata in sync only when there are security implications to that (like exposing uninitialized disk blocks) and if not, we are as lazy as we can to improve performance."

ZACK BROWN

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

Finally, Jeff remarked, “I think what we’ll have to do here is ensure that those filesystems do an `fsync` prior to reporting the `i_version` getattr codepath. It’s not pretty, but I don’t see a real alternative.” But Dave replied, “I think that’s even more problematic. -> getattr currently runs completely unlocked for performance reasons – it’s racy w.r.t. to ongoing modifications to begin with, so /nothing/ that is returned to user space via stat/statx can be guaranteed to be ‘coherent’.”

Jan suggested just updating `i_version` for all inodes after any system crash. Since the version was a 64-bit number, such updates wouldn’t really risk hitting the limit. One problem with this would be that it would invalidate any existing cached data, which would then have to be reloaded from disk. As Bruce put it, “Even if it’s rare, it may be really painful when all your clients are forced to throw out and repopulate their caches after a crash. But, yes, maybe we can live with it.”

At one point, Dave said, “The big question is how do we know there was a crash? The only thing a journaling filesystem knows at mount time is whether it is clean or requires recovery. Filesystems can require recovery for many reasons that don’t involve a crash (e.g. root fs is never unmounted cleanly, so always requires recovery). Further, some filesystems may not even know there was a crash at mount time because their architecture always leaves a consistent filesystem on disk.”

Jan pointed out that each filesystem had its own level of awareness of the state of the system or the occurrence of crashes. He suggested that each filesystem could implement a separate “crash flag,” although he acknowledged that this would require each filesystem to change their on-disk format, which would be undesirable.

The discussion continued, with a variety of suggestions proposed and rejected. Ultimately the problem seems to boil down to having a vast array of filesystems, all of which would ideally be supported. At the same time, the goal of Jeff’s code is to speed things up, so there’s a limit to how bad a performance hit any supporting infrastructure will be allowed to introduce, and, of course, filesystem maintainers are always reluctant to change on-disk data formats, so there are limits to how big a change Jeff can expect from those codebases.

Controlling Console Output

Calvin Owens from Facebook pointed out that some hardware consoles were slower than others, making it difficult to know how much debugging output a piece of code should produce. Too much output could clog a slow console, while a fast one would have no problem. But because Linux offered no per-console way to control console verbosity, users would have to configure the kernel to match the slowest console attached to the system. Calvin wanted to change that.

He posted a patch to let the user control the verbosity of each individual console. Joe Perches gave an immediate thumbs up, and Sergey Senozhatsky also really liked the concept, although he had some technical suggestions, which Steven Rostedt agreed with. They both thought that Calvin’s code should respect the “ignore loglevel” parameter, which Calvin’s code ignored. He said he’d update the patch to take care of it.

Petr Mladek also suggested, “people want to see all messages even on the slow console during `panic()`, `oops()`, with `ignore_loglevel`. It means that the new per-console setting must not limit it. Also any console must not go below `minimum_console_level`.”

The discussion didn’t go very far. Everyone seemed to be on board with the basic idea, and there didn’t appear to be any big stumbling blocks. I’d probably put this in the category of features everyone will appreciate but that no one knew they wanted until companies like Facebook, Google, and the rest started building massive data centers.

Cleaning Up Data Transfer Code

Al Viro knew that the mechanisms for copying large amounts of data between kernel and userspace had become more and more unmanageable over time, with code duplication, inconsistent semantics, and a big pile of bugs. The problem was in the `uaccess.h` code, which was all over the map.

His first step toward fixing it was to force everyone to use the same `uaccess.h` file. He’d already done this in a recent kernel release, and it allowed the rest of the issues to at least be addressed in a central location.

He posted a fresh patch that began to address the remaining problems, although he acknowledged that more would be needed. He said, "About 2.8K [lines of code] removed, a lot of cruft is gone and semantics is hopefully in sync now. All but two architectures (ia64 and metag) had been switched to new mechanism; for these two I'm afraid that I'll need serious help from maintainers."

The patch defined several standard function calls for copying data to and from userspace and kernel space. Each function also had clear behaviors for failure conditions.

Ultimately, the patch reduced the total number of functions to just eight, with standardized calling conventions. He added, "Some of per-architecture branches might be even done right; however, most of them got no testing whatsoever, so any help with testing (as well as 'Al, for f**k sake, dump that garbage of yours, here's the correct patch' from maintainers) would be very welcome."

He mentioned that the reason he hadn't implemented his patches for the ia64 and metag architectures was that "both architectures zero-pad in `__copy_from_user_inatomic()` and that really needs fixing. In case of metag there's `__copy_to_user()` breakage as well, AFAICS, and I've been unable to find any documentation describing the architecture wrt exceptions, and that part is apparently fairly weird. In case of ia64 … I can test mckinley side of things, but not the generic `__copy_user()` and ia64 is about as weird as it gets. With no reliable emulator, at that …. So these two are up to respective maintainers."

Vineet Gupta was happy that Al had taken on this work, and he posted an additional short patch of his own, to inline some of the code (i.e., to automatically replace function calls with duplicated code at compile time). Sometimes it's faster.

In this particular case, Al thought that any speed improvement would probably depend on the architecture and shouldn't be done for the whole kernel by default. He wanted to see some performance numbers showing that inlining really offered a speedup. And regarding inlining, Linus Torvalds said:

I would suggest against it.

The only part I think is worth inlining is the compile time size checks for kasan – and that only because of the obvious 'sizes are constant only when inlining' issue.

*We used to inline a *lot* of user accesses historically, pretty much all of them were bogus.*

The only ones that really want inlining are the non-checking ones that people should never use directly, but that are just helper things used by other routines (ie, the `unsafe_copy_from_user()` kind of things that are designed for `strncpy_from_user()`).

Once you start checking access ranges, and have `might_fault` debugging etc, it shouldn't be inlined.

Al agreed and reiterated that inlining could speed things up on certain architectures, but probably not for the general case. However, he also felt that keeping the option to inline available for any given architecture was not a problem and wouldn't make things more complicated in the general case. But Linus replied:

I disagree.

I think one of the biggest problems with our current `uaccess.h` mess is just how illegible the header files are, and the `INLINE_COPY_{TO,FROM}_USER` thing is not helping.

I think it would be much better if the header file just had:

```
extern unsigned long _copy_from_user⮧
  (void *, const void __user *, ⮧
   unsigned long);
```

and nothing else. No unnecessary noise.

*The same goes for things like `[__]copy_in_user()` – why is that thing still inlined? If it was a *macro*, it might be useful due to the `might_fault()` thing giving the caller information, but that's not even the case here, so we'd actually be much better off without any of that inlining stuff. Do it all in `lib/usercopy.c`, and move the `might_fault()` in there too.*

Al was neither for nor against. But he wanted to keep certain issues separate from the current patch, given the scope and complexity of the problems addressed. He first wanted to get all the architecture-dependent fixes to assembly language routines out of the way; in fact, he wanted to get everything out of the `arch/` directory altogether and to simplify all the many `uaccess.h` files.

He pointed out a large steaming pile of remaining disgustingness that would be higher on his list than the inlining issue.

But on that issue, in a nearby post, Al said, "Just to make it clear – I'm less certain than Linus that uninlined is uniformly better, but I have a strong suspicion that on most architectures it *is*. And not just in terms of kernel size – I would expect better speed as well. The only reason why these knobs are there is that I want to separate the "who should switch to uninlined" from this series and allow for the possibility that for some architectures inlined will really turn out to be better. I do _not_ expect that there'll be many of those; if it turns out that there's none, I'll be only glad to make the guts of `copy_{to,from}_user()` always out of line."

Elsewhere, Linus agreed that there were bigger fish to fry than the inlining issue, and there was no need to address everything at once. But he added, "I just think that we really *should* aim for a simpler `uaccess.h` in the long term, so I would prefer we not encourage architectures to do things that simply won't matter."

Meanwhile, Martin Schwidefsky tested Al's patch on the S/390 architecture and found that it worked fine, except for one bug, which he patched and Al accepted.

Alexey Dobriyan also did some testing, and offered some technical suggestions to Al.

Russell King did some testing on the ARM architecture and confirmed that the system would boot successfully, although he added, "there's definitely a reproducible drop in performance for some reason when switching between current and Al's uaccess patches, which is partly recovered by switching to the out of line versions." He said, "I'd suggest that we immediately switch to the uninlined versions on ARM so that the impact of that change is reduced. We end up with a 1.9% performance reduction rather than a 6% reduction with the inlined versions."

Al posted a test patch that was not intended to go into the kernel but only to help uncover the behavior Russell had seen. Linus also speculated on what might cause the slowdown.

Kees Cook also tested Al's code on the ARM and x86 architectures and reported no problems. He also said to Al, "Thanks for working on this! I've wanted to see this done for a long time; I'm glad you had the time for it!"

Al posted an updated patch based on the discussion so far, and the thread ended.

This is the kind of stuff Al works on – things the end user never sees or hears about but that make life easier for massive numbers of kernel developers, make it harder to introduce security bugs, and in general speed the kernel up, or at least won't really slow it down. ■■■

OpenStack Summit Boston

Alive and Kicking

Contrary to rumors, OpenStack is far from dead. *By Swapnil Bhartiya*

Jonathan Bryce, the executive director of the OpenStack Foundation, kick-started OpenStack Summit Boston by quoting a tweet: "OpenStack is as good as dead." He was referring to the continuous flow of stories that OpenStack is losing momentum. He presented a fact-based counterargument demonstrating the continuous growth of OpenStack.

The Foundation conducts regular surveys to keep a tab on usage. According to the latest survey, OpenStack deployments have grown 44 percent year after year. Bryce said that more than 50 percent of Fortune 100 companies are running OpenStack. It's a global phenomenon. OpenStack is powering over five million cores of compute power, spread across 80 countries. And two-thirds of these deployments are in production.

Bryce cited some of the reasons behind this growth. One reason is the evolution of the ecosystem and usage beyond hyperscale public clouds. He called it the second generation of cloud where a new consumption model has popped up – the remotely managed private cloud. It allows companies of all sizes to consume OpenStack.

The bottom line is that OpenStack is far from "as good as dead." These are signs of OpenStack's very healthy and organic growth.

However, with great growth comes great bloat. OpenStack paid a heavy cost with initial growth and interest in the project. As new and established companies started getting involved in OpenStack, they began creating sub-projects around OpenStack, which led to what Mark Shuttleworth called "BS-as-a-Service." This all led to unnecessary bloat and confusion around OpenStack.

The OpenStack Foundation organized a leader's summit to discuss these issues. OpenStack Foundation's Lauren Sell (VP, marketing and community services) and Thierry Carrez (VP, engineering) discussed the steps they have taken to cut down the bloat and improve communication to eliminate confusion around what's part of OpenStack and what's not. They also improved the process to create new leaders as OpenStack is growing globally.

Sell also stressed the importance of working with adjacent communities, like Kubernetes, that play a critical role in the OpenStack ecosystem.

Bryce invited Patrick Weeks, senior director of cloud, automation, and operations at GE Healthcare, to deliver a keynote speech about the work they have done to consume private cloud. Weeks was followed by Beth Cohen, cloud product technologist at Verizon, who talked about edge computing and how Verizon is using OpenStack to power a cloud-in-a-box system that's as big as a home router.

One of the highlights of the day was a keynote speech by Major Julianna Rodriguez (Director) and Chris Apsey (Deputy

Director) of the Cyber Technical College at the US Army Cyber School. The school is moving away from legacy technologies and heavily adopting open source and OpenStack to allow cyber warriors to protect US assets from cyber attacks.

Bryce also sat down with Jim Whitehurst, the CEO of Red Hat, to talk about OpenStack's governance model, which enabled OpenStack to have one upstream (i.e., one upstream project in which everyone contributes – no forks) that allowed the ecosystem to grow.

The general sessions concluded with the announcement of the OpenStack Superuser Award. This year the award was shared by two companies – Paddy Power Betfair and UKCloud.

Mark Collier, OpenStack Foundation's chief operating officer, led the second day of the Summit. Collier talked about the critical role computing power is playing in our world and went as far as saying that we have reached a point where all science is computer science.

That also means that the community has a massive responsibility to build tools that can be reliably used by all disciplines and more importantly for those tools to be accessible. That's where Collier mentioned that the open source first methodology is accepted as a best practice when you are solving a complicated problem. "Open Source is the best way to solve hard problems," Collier said.

Collier's sessions are known for live demonstrations, and in this case featured technology demos including CockroachDB, Ironic/Neutron, and deployment of Cinder as a standalone service through containers.

Living up to OpenStack Summit tradition, Brad Topol, IBM distinguished engineer, organized the interop challenge where more than a dozen engineers representing different companies performed a mega demo to deploy the same solution across different clouds. This time there were 16 participants, which means the same script was deployed across 16 different clouds within a matter of minutes.

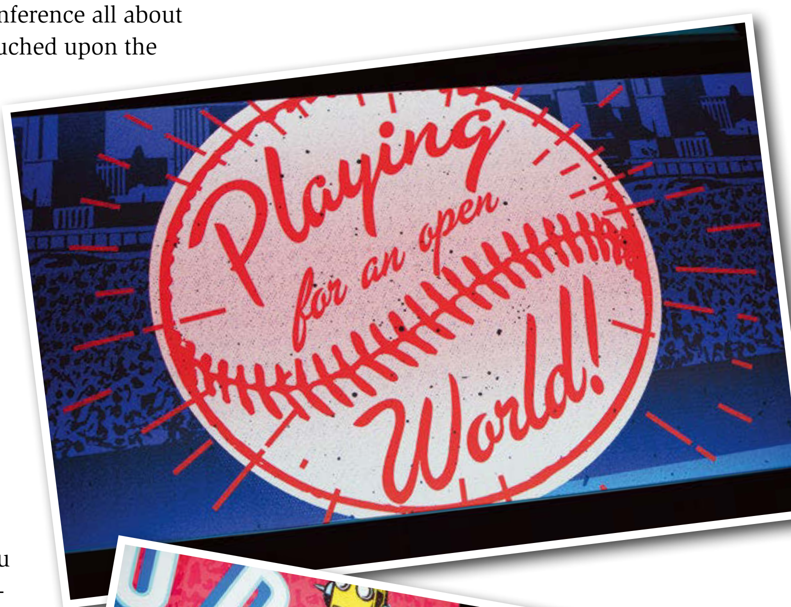
The highlight of the second day was the live interview with the former NSA contractor, Edward Snowden. Snowden was interviewed by Collier through live video conferencing from Russia. One might wonder why a whistleblower was at a commercial conference all about cloud. Snowden touched upon the core points that build a case for private cloud.

Comparing OpenStack private cloud with public cloud, Snowden said, whether you are a small business, a large business, or a community of technologists, you can own it, control it, and shape it. "You can build, you can lay the foundation upon which everybody builds, and I think that's probably one of the most powerful ideas that shapes the history of the Internet and hopefully will allow us to direct the future of the Internet in a more free, rather than more closed way," said Snowden.

Questioning proprietary technologies when there are security flaws, Snowden went on to say, "We can't evaluate if their response was positive or negative, if it was good enough, or not good enough, and ultimately even if we don't like it, we have no influence over it."

Snowden said that the point of open source is that we don't have to compromise. "We want a better world, and so we're going to build it," he said.

OpenStack is also known for throwing great parties, and this year they took attendees to Fenway Park, the Boston Red Sox stadium. In addition to a visit to the historical stadium, they also offered food and drinks – in a "free as in beer" manner! ■■■



Track security vulnerabilities with Network Security Toolkit

Network Audit

Securing networks against attackers is not a trivial task. The Network Security Toolkit is a convenient solution for users who want to keep a collection of security tools in easy reach. *By Erik Bärwaldt*

One of the administrator's most important tasks is keeping the managed network free from malicious software and intruders. You can't wait until the milk is spilled; you need to anticipate possible vulnerabilities and close them based on thorough analysis.

The Linux environment is home to many security tools; you first need to separate the chaff from the wheat – or maybe choose a distribution that specializes in security. For years, the Network Security Toolkit (NST) [1] has been carefully maintained and developed; it offers a veritable plethora of test and inspection tools that help to root out even the most exotic vulnerabilities.

Getting Started

The Fedora-based NST comes as a 2.8GB 64-bit ISO [2]. The hybrid image can be deployed from an optical disk, a USB stick, or a virtual machine (VM).

After booting, NST shows you GRUB with several options: In addition to a console-based Live mode without a desktop environment, you can also choose to launch a graphical environment. In this case, the system starts the Mate desktop (version 1.14.1) and presents itself with a look that is anything but spectacular (Figure 1).

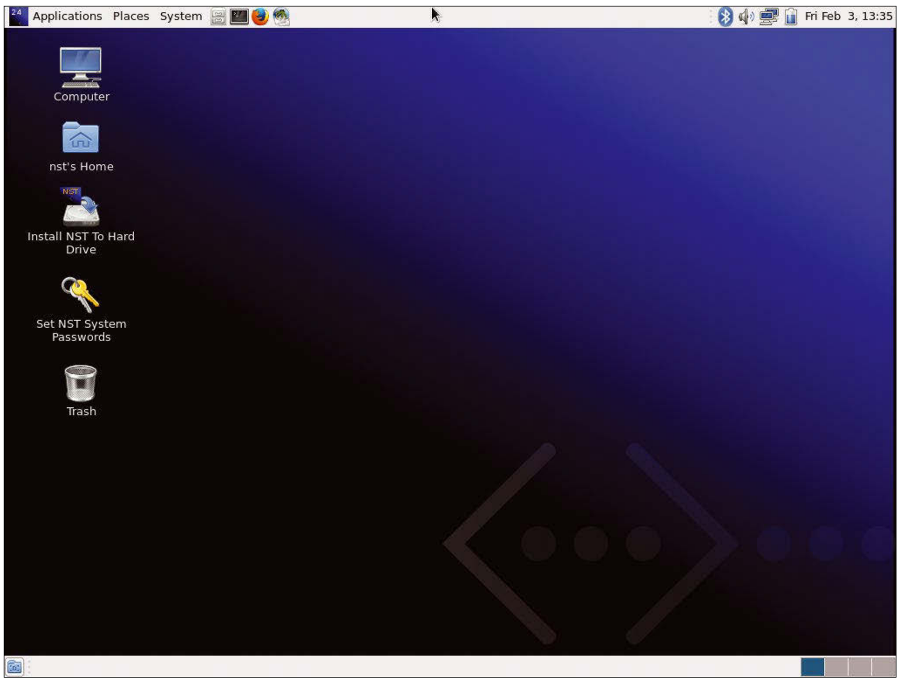


Figure 1: At first glance, there is little difference to be seen between NST and a conventional distribution.

On the desktop are a couple of standard buttons and two custom icons, *Install NST to Hard Drive* and *Set NST System Passwords*. If you want to use the tool collection in Live mode, you can change the keyboard layout if you like. To do this, use the ncurses dialog via the *System* | *Administration* | *Keyboard* menu.

Deployment on a dedicated machine, or a virtual system, is recommended for stationary operation. This means that you can also easily save data and updates, which are inevitably lost at the end of the session on an optical disk or on a USB flash drive without a persistent storage area due to the inability to write.

Virtual Machine

NST can be deployed on a VM with Oracle's VirtualBox or VMware. If you want to install the Fedora derivatives under VirtualBox, it is a good idea to assign the VM at least 1.5GB RAM and two CPU

cores. Otherwise, the distribution runs very slowly and sluggishly. The toolkit's usability can be thus limited on older Core2 Duo processors for desktops and notebooks, because they do not support hyperthreading and have only two physical cores.

As soon as you have started the virtualization environment on your system and created the VM with the ISO image as a bootable disk, the system boots to the Mate screen, from where you call the Fedora Installer Anaconda by selecting *Install NST to hard drive*.

With only three steps – localization, partitioning, and password assignment – the wizard bundles the system onto your virtual mass storage device. Then you can turn off the VM and remove the ISO image from disk management in the configuration dialog of the virtualization environment. The Network Security Toolkit boot should be much faster from the virtual mass storage than previously.

If it did not create a user account during the install, the Anaconda installer automatically shows you the dialog for creating a user right after the first boot. Clicking on *Complete Configuration* bottom right in the Anaconda dialog terminates the wizard and takes you to the login screen.

You can then access the work environment. Please note that due to the technical nature of VMs, there is an inherent limit to the use of NST in such an environment: For example, WiFi tools do not work because only virtual Ethernet interfaces are available.

Menus

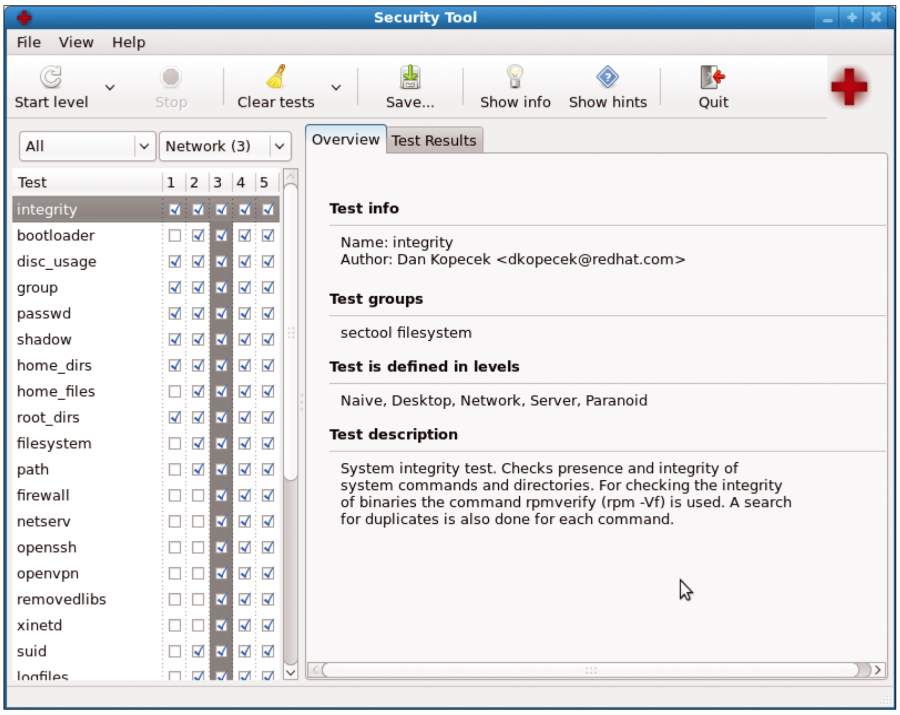
NST integrates most of the security applications listed on SecTools.org [3]. Although you will find the standardized hierarchical menu structure on the system, the program scope shows significant differences from popular all-around desktops.

For example, the *Applications* | *Internet* submenu contains a range of well-known security tools, from AirSnort and the Angry IP scanner, through EtherApe and Ettercap, to Tcptrack and WiFi Radar. The *System Tools* submenu contains more analysis tools, such as Cockpit, Traceroute, or Wireshark. Here you will also find SecTools, an auditing tool that belongs to the Fedora collection (Figure 2).

Web-Based: WUI

Because the tools listed in the menus by no means reflect the complete inventory of tools – including all of these applications would clutter the menus – the NST developers have outsourced most of the packages to what they call the Web User Interface (WUI), which can be opened in any browser.

To use this interface, you first need to modify the authentication data on the NST system. This is done in the terminal with the `nstpasswd` command or alternatively in the splash screen by clicking on the *Set NST System Passwords* button. Customizing the authentication data in the terminal at the same time starts the SSH and HTTPS servers so that you can log in from remote machines over an encrypted connection. When you install on a hard disk, you only need to change the authentication data



Test	1	2	3	4	5
integrity	✓	✓	✓	✓	✓
bootloader	☐	✓	✓	✓	✓
disc_usage	✓	✓	✓	✓	✓
group	✓	✓	✓	✓	✓
passwd	✓	✓	✓	✓	✓
shadow	✓	✓	✓	✓	✓
home_dirs	✓	✓	✓	✓	✓
home_files	☐	✓	✓	✓	✓
root_dirs	✓	✓	✓	✓	✓
filesystem	☐	✓	✓	✓	✓
path	☐	✓	✓	✓	✓
firewall	☐	☐	✓	✓	✓
netserv	☐	☐	✓	✓	✓
openssh	☐	☐	✓	✓	✓
openvpn	☐	☐	✓	✓	✓
removedlibs	☐	☐	✓	✓	✓
xinetd	☐	☐	✓	✓	✓
suid	☐	✓	✓	✓	✓
logfiles	☐	✓	✓	✓	✓

Test info

Name: integrity
Author: Dan Kopecek <dkopecek@redhat.com>

Test groups

sectool filesystem

Test is defined in levels

Naive, Desktop, Network, Server, Paranoid

Test description

System integrity test. Checks presence and integrity of system commands and directories. For checking the integrity of binaries the command rpmverify (rpm -Vf) is used. A search for duplicates is also done for each command.

Figure 2: Fedora's SecTools performs an initial check of the system for vulnerabilities.

once; if you rely on an optical disk as the boot medium, you need to create passwords at each restart.

Next, open the web browser, which already has the NST interface as its home page, and enter the authentication data that you defined previously. If you want secure access instead of the open HTTP connection you are offered, go to `https://127.0.0.1:9943`. The insecure connection uses port 9980.

In the rustic-looking overview window of the toolkit (Figure 3), you will find data and instructions at the bottom and a menubar at the top. It contains the individual tools in hierarchically organized submenus. For command-line tools, you get a terminal display with the appropriate input and output below the menubar, in many cases supplemented by context-sensitive information.

From the *Docs* menu, you can reach the comprehensive program documentation. Some is available in wiki format; other parts deal with specific scripts and explain the use of the Toolkit. You will also find contact links for the developers, as well as lists of existing packages.

The *Tools* menu adds system tools to the desktop. These include file converters, terminals, and editors, as well as developer tools. The *System* menu holds applications for system configuration and customizing services such as CUPS or Apache. You will also find expansions and updates for NST and can launch the installers from here if necessary. You do not typically need to configure the services. This menu also includes tools for virtual environments, viewers for logfiles, and various file managers.

The *Network* menu deals with network-specific problems. Here you will find monitoring software and analysis programs, various applications for NFS and CIFS services, and various tools for WiFi. The *Security* menu contains numerous security applications, broken down into intrusion detection systems (submenu *Intrusion Detection*), active and passive network scanners, and virus scanners.

The toolbox includes EtherApe, Hping, InetVis, OpenVAS, Nikto, Nmap, and Xprobe 2, as well as virus scanners for use in heterogeneous environments, ClamAV, ClamWin, McAfee, and Norton. This is also where you will find references to useful information.

The *Geolocation Tools* section is where NST organizes programs that let you locate and display individual computers or entire networks around the globe – and in a visually appealing way. Various tools for generating hashes and passwords complete the offering.

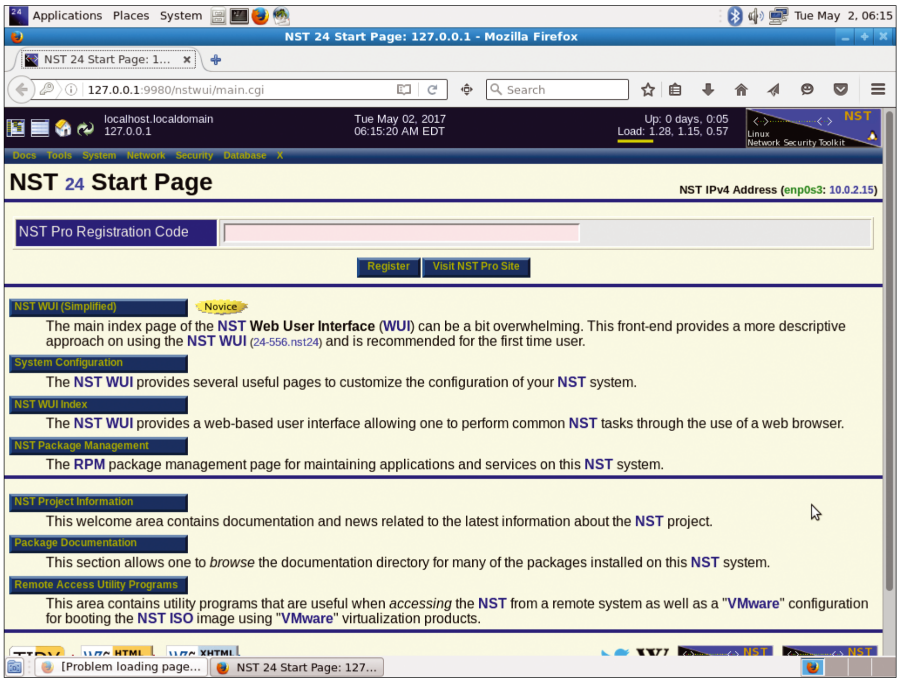
The *Database* submenu is far less extensive, focusing on tools for MySQL and MariaDB databases. The last menu item on the right, an *X*, summarizes the above-mentioned groups and integrates individual applications for network and system management from the standard Mate menu (Figure 4).

Directly above the menubar, you'll see several buttons on the left; of these, the screen button at the edge of the screen opens a view with the documentation and – grouped by the menu labels – also shows the applications as executable links in a table view. Here, you will find some basic statistical data, such as the number of users logged on to the system and the number of running processes. You can also call the applications grouped here directly via the links. This eliminates the long-winded approach via the very full menus.

Unique Selling Points

IT security does not need to be a boring affair at the command line, reserved only for geeks – as NST proves with its geolocation option. It lets you determine the locations of various hosts between which connections exist and shows a world map where changes appear almost in real time. This is all based on Google services, proprietary scripts, and databases, as well as Traceroute and Ntop.

Discovering locations is a convenient process in the web-based interface – no need to type console commands. The wiki


Applications Places System
Tue May 2, 06:15
NST 24 Start Page: 127.0.0.1 - Mozilla Firefox
NST 24 Start Page: 1...
127.0.0.1:9980/nstwui/main.cgi
Search
Docs Tools System Network Security Database X
Linux Network Security Toolkit
NST
Register
Visit NST Pro Site
Novice
[Problem loading page...
NST 24 Start Page: 127...

localhost.localdomain
127.0.0.1

Tue May 02, 2017
06:15:20 AM EDT

Up: 0 days, 0:05
Load: 1.28, 1.15, 0.57

NST 24 Start Page

NST IPv4 Address (enp0s3: 10.2.2.15)

NST Pro Registration Code

NST WUI (Simplified)
The main index page of the NST **Web User Interface** (WUI) can be a bit overwhelming. This front-end provides a more descriptive approach on using the NST WUI (24-556.nst24) and is recommended for the first time user.

System Configuration
The NST WUI provides several useful pages to customize the configuration of your NST system.

NST WUI Index
The NST WUI provides a web-based user interface allowing one to perform common NST tasks through the use of a web browser.

NST Package Management
The RPM package management page for maintaining applications and services on this NST system.

NST Project Information
This welcome area contains documentation and news related to the latest information about the NST project.

Package Documentation
This section allows one to *browse* the documentation directory for many of the packages installed on this NST system.

Remote Access Utility Programs
This area contains utility programs that are useful when *accessing* the NST from a remote system as well as a "VMware" configuration for booting the NST ISO image using "VMware" virtualization products.

Figure 3: It may look frumpy, but it is functional: the main window of the NST web-based interface.

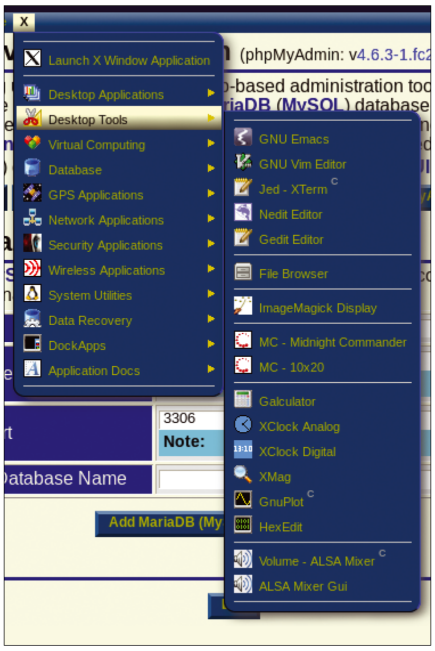


Figure 4: The menus in NST are generally well filled and include many important tools out of the box.

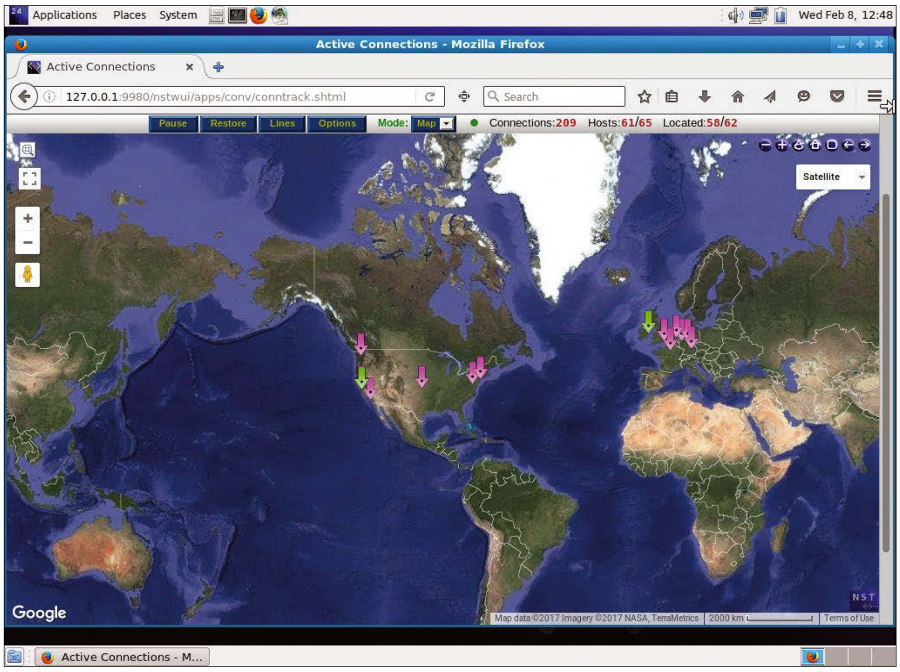

Applications Places System
Wed Feb 8, 12:48
Active Connections - Mozilla Firefox
Active Connections
127.0.0.1:9980/nstwui/apps/conv/conntrack.shtml
Search
Pause
Restore
Lines
Options
Mode: Map
Connections:209 Hosts:61/65 Located:58/62
Satellite
Google
Map data ©2017 Imagery ©2017 NASA, TerraMetrics 2000 km
Terms of Use
Active Connections - M...

Figure 5: Visualizing connections with NST's geolocation features.

explains in detail what geolocation features NST offers and how to use them [4]. Screencasts get newcomers off to a good start (Figure 5).

Performance

Beginners and less experienced admins might feel overwhelmed on first contact with NST because of the vast range of applications. The menu structure in the WUI, which is quite complicated in part, can present a considerable learning curve, especially for less experienced users searching for a particular application. To mitigate this shortcoming, there is an optional simplified interface, which you can access from the splash screen by pressing the *NST WUI (Simplified)* button.

The routine branches to a list view with only four groups of tools: *Network Tools*, *System Information*, *System Administration*, and *Serial Port Tools* (Figure 6). Because the developers have also cleaned up the subgroups here, this view is much better suited for getting started.

More intuitive names for the subgroups help you find specific tools faster: For example, the *Network Tools* group includes the subgroups *Network Sniffer*, *Network Scanning*, and *Network Monitoring*. *Network Penetration Testing* takes you to applications for performing security checks. Thanks to this view, both new and experienced users can start production operations with the applications within a very short time.

Documentation

NST's extensive documentation thoroughly describes the functions of the individual applications, the structure of the system, and the way the web-based interface works.

You can reach the various guides via the *Docs* menu and its subgroups in the conventional web-based interface. You can also do this via the *Package Documentation* and *NST Project Information* submenus in the simplified work environment.

Additionally, most browser views offer a NST Wiki button, either at the top or bottom right, which opens the comprehensive wiki in a new tab.

Conclusions

NST is a toolkit that no serious administrator would want to do without. Although this collection of software can be a bit intimidating, the simplified interface – in combination with excellent documentation – enormously facilitates the user's first steps.

The web-based environment and preconfigured services reduce the overhead for leveraging the benefits of NST and ensure smooth deployment on the network. NST thus establishes itself as an ideal port of call, whenever you need to protect heterogeneous networks against attacks or hunt down security vulnerabilities. ■■■

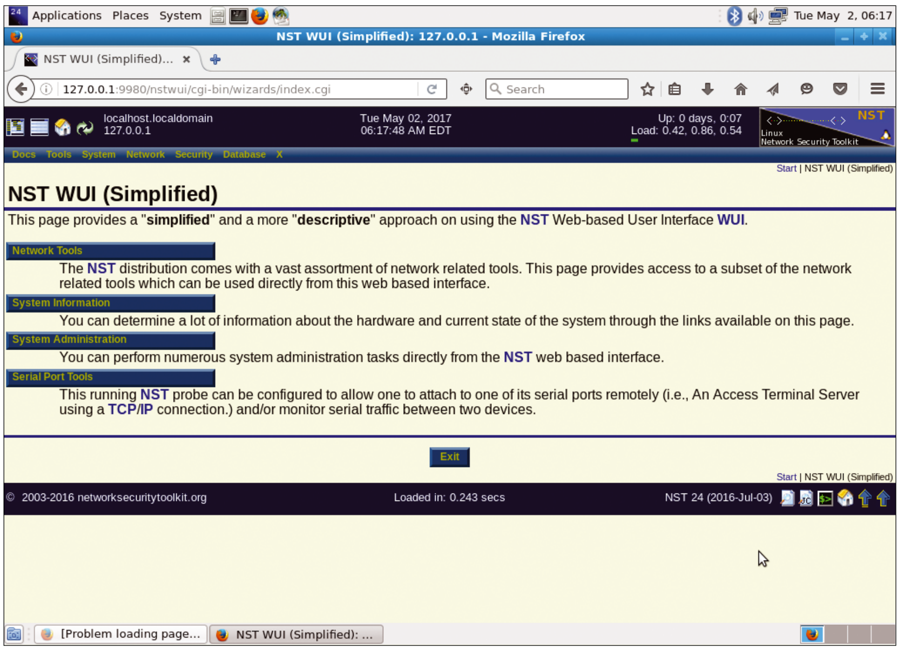
INFO

[1] NST: *http://www.networksecuritytoolkit.org/nst/*

[2] NST download: *https://sourceforge.net/projects/nst/files/NST/NST%2024-7977/*

[3] SecTools: *http://sectools.org/*

[4] Geolocation: *http://wiki.networksecuritytoolkit.org/index.php/HowTo_Geolocate_Data_Using_The_NST_WUI*


Applications Places System
Tue May 2, 06:17
NST WUI (Simplified): 127.0.0.1 - Mozilla Firefox
NST WUI (Simplified)...
127.0.0.1:9980/nstwui/cgi-bin/wizards/index.cgi
Search
Docs Tools System Network Security Database X
Exit
[Problem loading page...
NST WUI (Simplified): ...

localhost.localdomain
127.0.0.1

Tue May 02, 2017
06:17:48 AM EDT

Up: 0 days, 0:07
Load: 0.42, 0.86, 0.54

Start | NST WUI (Simplified)

NST WUI (Simplified)

This page provides a "**simplified**" and a more "**descriptive**" approach on using the **NST** Web-based User Interface **WUI**.

Network Tools
The **NST** distribution comes with a vast assortment of network related tools. This page provides access to a subset of the network related tools which can be used directly from this web based interface.

System Information
You can determine a lot of information about the hardware and current state of the system through the links available on this page.

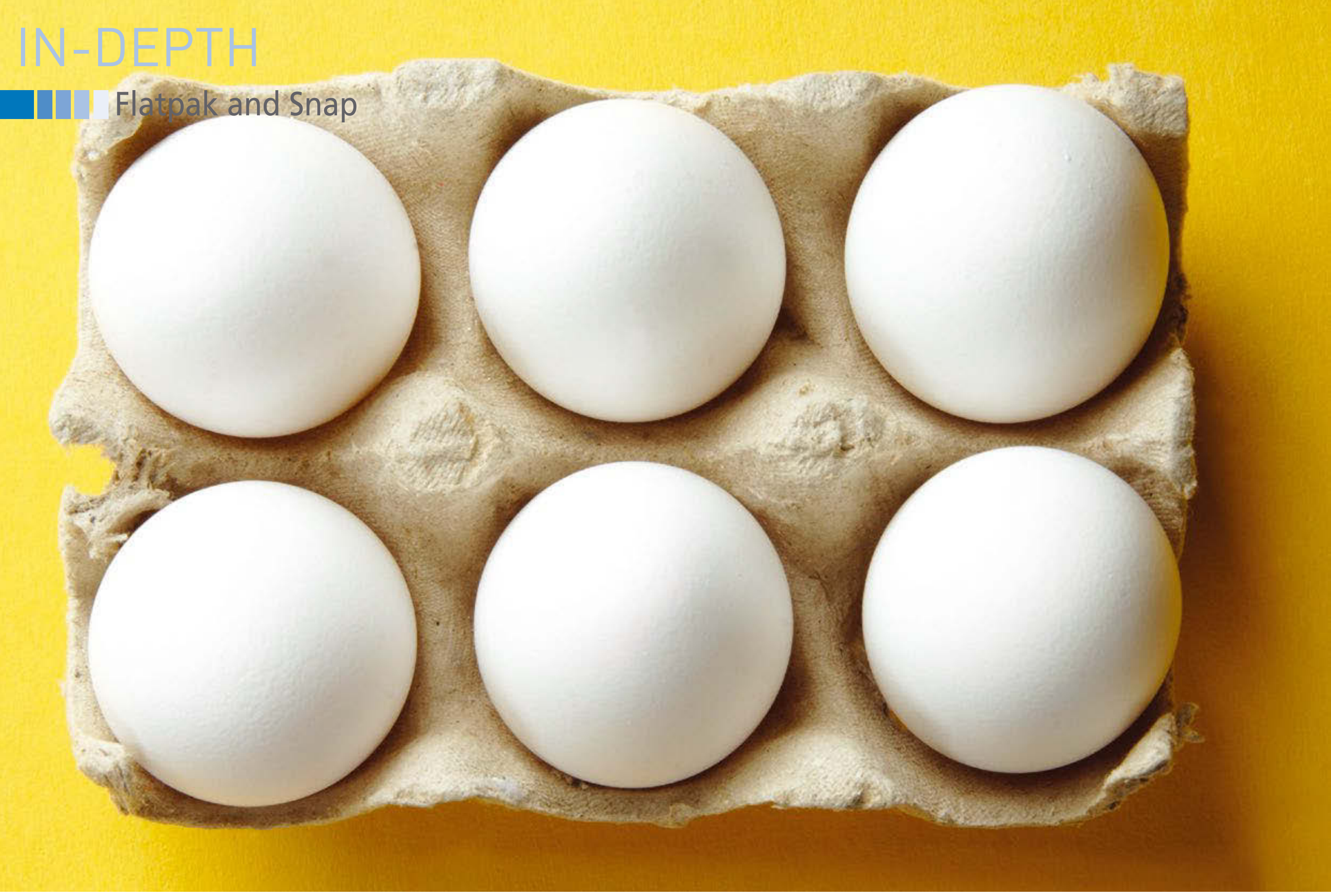
System Administration
You can perform numerous system administration tasks directly from the **NST** web based interface.

Serial Port Tools
This running **NST** probe can be configured to allow one to attach to one of its serial ports remotely (i.e., An Access Terminal Server using a **TCP/IP** connection.) and/or monitor serial traffic between two devices.

Start | NST WUI (Simplified)

© 2003-2016 networksecuritytoolkit.org   Loaded in: 0.243 secs   NST 24 (2016-Jul-03)

Figure 6: The simplified NST WUI view.



Exploring the new Flatpak and Snap package formats

Smartly Packed

The new container-inspired package formats Flatpak and Snap have landed in the territory occupied by conventional Linux package systems such as RPM and Dpkg. *By Martin Loschwitz*

Linux distributions are subject to constant change. One of the most recent victims of this constant quest for self-renewal is the classic init system, which most distros have now replaced with systemd.

Another central feature of most Linux distributions is the package management system. The most common package systems are RPM [1], which is used with Red Hat, and Dpkg [2], which is favored by Debian and Ubuntu.

The RPM and Dpkg systems have both benefited from changes and improvements over the past few years, and the idea of completely replacing one of these package management systems with something else has never been seriously considered – until now.

Flatpak [3] and Snap [4] swim in the wake of the container fleets. Both tools envision the package as something like a small Linux system. The package contains all programs and libraries that the application needs, which simplifies the host system, letting it function as a pure container hypervisor. The underlying system still provides a home for containers, but it does not run any programs.

Conventional Packages

No matter how practical RPM and Dpkg may be in everyday life, they regularly cause headaches for software developers and admins.

One problem with conventional package systems is that they are slow – even if the database resides on a fast SSD. In addition, practically every admin will be familiar with the problem of *dependency hell*. If you use an LTS distribution, but you need a new version of a single program, you cannot simply install the necessary package from the successor distribution. Instead you have to create backports of different libraries and packages and install them collectively.

And, because of the diversity of the Linux platform, if you want to provide a tool for SUSE, Red Hat, Debian, and Ubuntu at the same time, you have to create four individual packages and individually keep them up to date and in line with the distribution release cycle. The complexity of package maintenance is one of the reasons why many developers only release the source code of their pro-

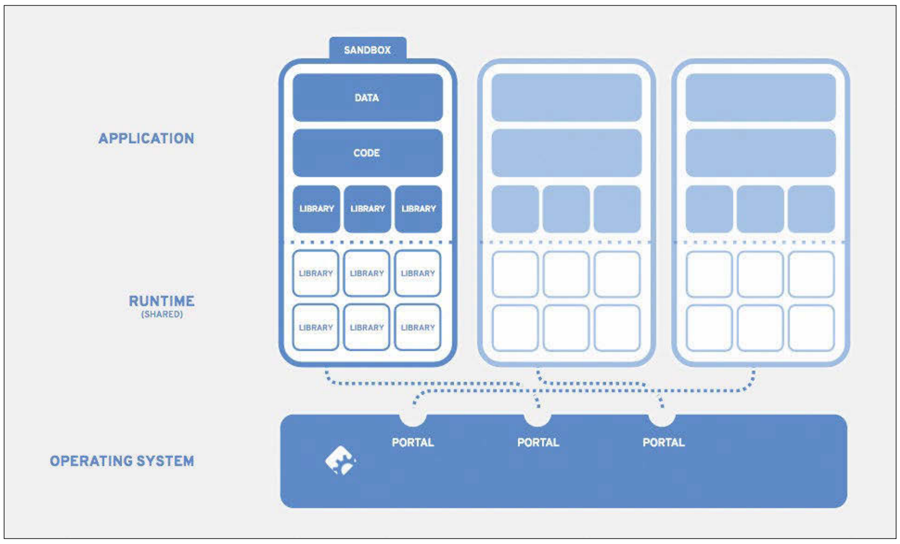


Figure 1: **All generic data come from the Runtime – the Flatpak contains only the app and the necessary libraries.**

grams and hope that the various Linux vendors will independently create a suitable package for it.

Light at the End of the Tunnel?

Flatpak and Snap offer a new approach to the issues that affect RPM and Dpkg. Although these container-inspired solutions officially have nothing to do with Docker, LXC, and other container projects, both solutions rely on the Linux kernel functions that support containers, achieving strict separation of the guest and host using Cgroups [5] and Namespaces [6].

The package formats for Flatpak and Snap come with their own userland. It is not only the programs that reside in this space, but also all the libraries and tools that the program needs to run smoothly.

Flatpak and Snap each deliver their own control program that runs on the physical host and performs a kind of bridge function. Although the programs packaged in Snaps or Flatpaks basically act separately from the rest of the system, they still need access to various host files. For example, they need to parse their own configuration files, which are usually stored in `/etc` or in the home directory of the executing user. Both Snap and Flatpak therefore define internal interfaces that allow access to external resources.

Exploring Flatpak

Before you get started with Flatpak, it is best to start with a few important terms.

First is the *runtime*, which is not part of a Flatpak but is an external component that Flatpak integrates. The runtime consists of a basic system with the most important files and libraries (Figure 1).

The Sandbox is the container in which the Flatpak app runs. (The developers avoid using the term container because it suggests comparisons with Docker [7] or LXC [8].) The Sandbox is the virtual environment that runs on the hypervisor and where the application is executed.

Bundled libraries are part of the sandbox. The Flatpak often needs libraries that are not part of the Runtime and therefore must be provided separately.

The *SDK* (Software Development Kit) is also an important element of the Flatpak ecosystem. The SDK is similar to the runtime but also contains the development files. In other words, the developer builds the Flatpak against a certain SDK version, and this Flatpak then runs reliably with the runtime associated with the SDK.

To get started with Flatpak, check whether the package is available through your current package manager. Arch Linux and Fedora include a Flatpak package out the box. Debian Testing comes with a `flatpak` package, and a backport for Debian Jessie is also available [9]. Packages from the manufacturer are also available for OpenSUSE Tumbleweed [10].

If you use Ubuntu, you'll find the packages in a PPA by the main Flatpak author on Ubuntu's Launchpad code platform, and you can install them with the following three commands:

```
sudo add-apt-repository ⮒
  ppa:alexlarsson/flatpak
sudo apt update
sudo apt install flatpak
```

Rolling your Own Flatpak

Once Flatpak is installed and the `flatpak` program is available, the developer creates an empty directory for a new flatpak:

```
flatpak build-init ⮒
  helloworld org.example.⮒
  HelloWorld org.gnome.Sdk ⮒
  org.gnome.Platform 3.22
```

The individual parameters behind `build-init` each have a special significance: `helloworld` specifies the directory name within which the flatpak is created; `org.example.HelloWorld` defines the D-bus name that the flatpak uses later to communicate with D-Bus. The SDK and the runtime, which the flatpak uses, are defined by `org.gnome.Sdk` and `org.gnome.Platform`, and `3.22` is the version number.

This example uses Gnome's SDK and runtime, which have been extensively tested and are fine for most everyday applications, but other options are also available.

The next step adds the program you wish to include in the flatpak. The `flatpak build` command is very useful; the package builder uses it to execute arbitrary commands in the flatpak build environment. For example,

```
flatpak build helloworld touch /app/test
```

creates a `test` file in the flatpak `app` directory. You can also compile a program within Flatpak. First, download the sources [11] and unzip them. Next, type `cd` to change to the folder with the unpacked sources; enter the following commands when you get there:

```
flatpak build ⮒
  flatpak_folder_name gcc ⮒
  $(pkg-config --cflags ⮒
  dbus-1 glib-2.0) -o ⮒
  send-hello dbus-send-hello.c ⮒
```

LISTING 1: Running the Flatpak

```
flatpak build-export repo helloworld
flatpak --user remote-add --no-gpg-verify --if-not-exists example-repo repo
flatpak --user install example-repo org.example.HelloWorld
flatpak run org.example.HelloWorld
```

```
  $(pkg-config --libs ⮐
  dbus-1 glib-2.0)
```

`flatpak_folder_name` is followed by the relative path to the directory just created by `flatpak build-init`.

In the next step, you create the actual flatpak:

```
flatpak build-finish ⮐
  helloworld --socket=x11 ⮐
  --share=network ⮐
  --command=send-hello ⮐
  --socket=system-bus ⮐
  --socket=session-bus
```

The parameters `--socket=x11` and `--share=network` give the flatpak app access to X11 and the network, which is not necessary for this example but is for many applications.

In the next step, you can execute the flatpak with the help of the commands from Listing 1.

The four commands create a Flatpak repository based on the local flatpak, add it to the Flatpak repository configuration, install the Flatpak from the repo, and finally execute it. The reward for all this effort is a *Hello World* message in the D-Bus logfile on the host system.

Powerful Tool

This example demonstrates only a tiny part of Flatpak's capability. The Flatpak website offers examples of how the package system can accommodate far more complex applications in a flatpak [12]. The granularity with which the developers have organized rights assignments is worthy of note. For each flatpak, the package maker determines whether the software needs access to X11, Wayland, D-Bus, or individual folders within the host filesystem. You can even control access to audio devices.

If you experiment with more complex Flatpak examples, you will also stumble across `flatpak-builder`, which sounds a lot like `flatpak-build` but is a separate tool. `Flatpak-builder` accesses information from a JSON file that you created up front and builds a flatpak automatically.

The JSON file essentially contains the same information that you previously passed into `flatpak-build` at the command line. The tool thus extends the principle to include a kind of batch mode, which proves to be very useful in practice (Figure 2).

Exploring Snap

Unlike Flatpak, Snap is not targeted at desktop users. Canonical instead envisions Snap as a tool for revolutionizing the way enterprise applications reach servers. Like Flatpak, Snap relies on the kernel's various isolation techniques, combining them so applications run in their own isolated environments.

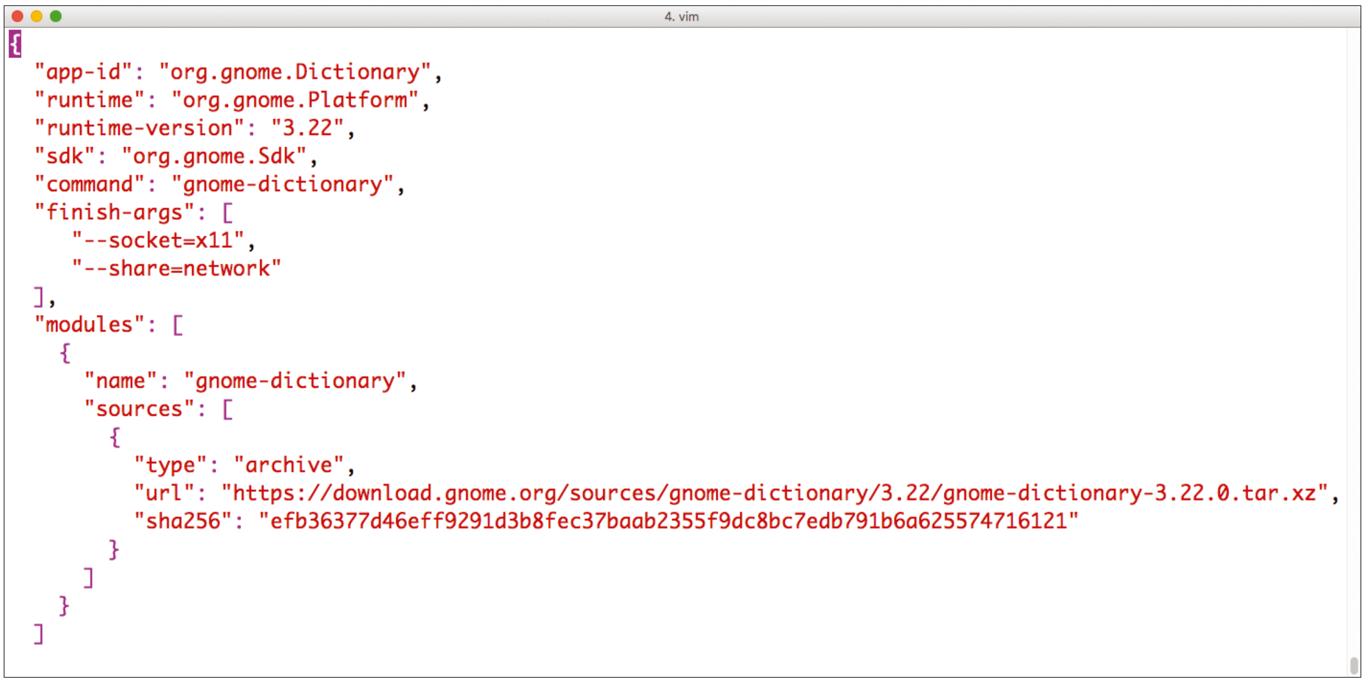
The structure of a snap is not fundamentally different from that of a flatpak, but the terms have different names. Instead of a runtime, Snap refers to a framework, which is a basic set of libraries on which Snap can build.

Unlike Flatpack, Snap is not aimed at general-purpose systems. Canonical is developing a basic system known as the Snappy Ubuntu Core [13] that offers different frameworks designed to work with the Snap package system (Figure 3).

YAML Files

Like Flatpak, Snap offers the ability to create a package based on a JSON file, but YAML format is standard. You can define keywords and paragraphs in the YAML file and then feed the file to Snapcraft [14], which builds a complete Snap from the information.

Snap comes with the GNU Autotools plugin, which you can use for building a program from the source code. You can



```
{
  "app-id": "org.gnome.Dictionary",
  "runtime": "org.gnome.Platform",
  "runtime-version": "3.22",
  "sdk": "org.gnome.Sdk",
  "command": "gnome-dictionary",
  "finish-args": [
     "--socket=x11",
     "--share=network"
  ],
  "modules": [
    {
      "name": "gnome-dictionary",
      "sources": [
        {
          "type": "archive",
          "url": "https://download.gnome.org/sources/gnome-dictionary/3.22/gnome-dictionary-3.22.0.tar.xz",
          "sha256": "efb36377d46eff9291d3b8fec37baab2355f9dc8bc7edb791b6a625574716121"
        }
      ]
    }
  ]
```

Figure 2: You can generate a Flatpak using a JSON file.

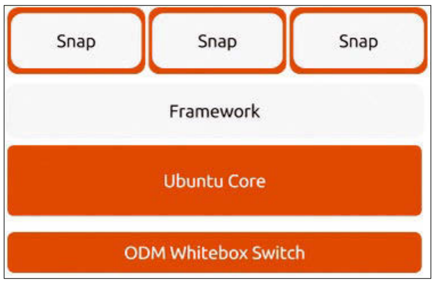


Figure 3: **Snap system is designed to work with Snappy Ubuntu Core.**

use the built-in plugin interface to include functions that are missing in the Snap core. The Autotools plugin ensures that the familiar three-step process, `./configure`, `make`, and `make install` works when you convert source code to binary code.

The keywords, `name`, `version`, and `summary` let the snap creator define the name of the snap, the version number, and the summary it will report. The `confinement` keyword is particularly important: options include `strict` (the default option) or `devmode`. `strict` strictly isolates the Snap from the rest of the system. For production snaps, `confinement: strict` is more or less mandatory. A ready-to-use YAML file for a `Hello World` snap would look like Listing 2.

If you are building snaps of more complex programs, you need to expand the `parts` section. If the program requires dependencies in the form of libraries, enter them in the appropriate order in the YAML file.

To build the snap, store the YAML file as `snapcraft.yaml` and run the `snapcraft` command within the directory. (Be sure the `snapcraft` package is installed on the system.) On Ubuntu systems, you will find `snapcraft` in the official archive; for other distributions, see the instructions online [15]. Snapcraft is available for both Fedora and OpenSUSE.

Following the Snapcraft call, find a file with a `.snap` file suffix in the working directory. To run the snap on a system, the system needs the Snap daemon, snapd. Once snapd is running, you can install the snap:

```
sudo snap install *.snap --dangerous
```

Concerned readers might be a little worried about the `--dangerous` parameter. Don't panic! The parameter just tells Snap to skip the built-in signature validation, which ensures that only snaps from sources that the admin explicitly trusts are installed on the system.

The example shown here only scratches the surface of what is possible with Snap. The very detailed documentation, which sheds light on the possibilities and options, is available on the program website [16].

Interface System

A container app that is completely isolated from the rest of the system is of limited use. To integrate itself with the normal workflow, a container app must at least be able to access a user's personal directory. Snap solves this problem by offering standardized interfaces for different types of services [17].

For example, the `home` interface gives a snap access to user folders. The `dbus` interface allows access to the D-Bus on the host system. Some of the interfaces have

LISTING 2: snapcraft.yaml

```
name: hello
version: "2.10"
summary: GNU Hello, the "hello world" snap
description: GNU Hello prints a friendly greeting
confinement: strict

apps:
  hello:
    command: hello

parts:
  gnu-hello:
    plugin: autotools
    source: http://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz
```

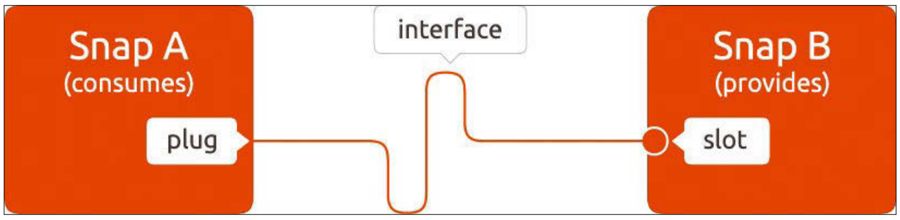


Figure 4: Communication between Snaps relies on plugs and slots, which are specified in the YAML definition.

more exotic capabilities: For example, the `camera` interface lets a snap access a connected camera; `tpm` lets the snap control a Trusted Platform Module (TPM) for cryptographic functions.

The Snap developers have also put some thought into the way snaps communicate with each other. An application that provides a service, can define a `slot` in its YAML definition – say, a MySQL database. Another snap can define a `Plug`, which then connects to the previously defined `slot` (Figure 4).

The Big Difference: A Ready-Made Online Repository

The most striking difference between Flatpak and Snap is that Canonical is already running an online marketplace for ready-made snaps (Figure 5) [18]. The store offers snaps that administrators and developers have built for different use cases.

All snaps that end up in the Canonical Snap Store are digitally signed – users can therefore be sure that they are picking up the snaps from a safe source. (Flatpak is reportedly working on a similar feature.)

Conclusions

Flatpak and Snap shake the foundations of the package management system principle, causing tremors among the stalwarts of all current Linux distributions. At first glance, the idea behind these upstart package systems makes good sense: If you are constantly dealing with backports, or if you have to resolve package dependencies manually, provisioning independent containers with applications is certainly an appealing concept.

On closer inspection, however, it turns out that the principle still has some rough edges. Snap and Flatpak are pursuing similar goals and are basically competing approaches. Fedora, the driving force behind Flatpak, insists that their system is more at home on desktops, and Canonical, maintainer of the Snap, touts the benefits of Snap as a tool for server rooms.

The user groups forming up around the Flatpak and Snap communities seem eerily similar to the split between RPM and Dpkg that has inconvenienced the Linux world for decades. It might be more sensible to agree on a common approach and to pursue it. But this hope seems futile given the dynamics of the open source world. ■■■

INFO

[1] RPM: *http://rpm.org*
[2] Dpkg: *https://wiki.debian.org/dpkg*
[3] Flatpak: *http://flatpak.org*
[4] Snap: *https://www.ubuntu.com/desktop/snappy*
[5] Cgroups: *https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt*
[6] Namespaces: *https://lwn.net/Articles/531114/*
[7] Docker: *https://www.docker.com*
[8] LXC: *https://linuxcontainers.org*
[9] Flatpak backport für Debian Jessie: *https://packages.debian.org/jessie-backports/flatpak*
[10] Flatpak for Open Suse Tumbleweed: *https://software.opensuse.org/package/flatpak*
[11] `Hello World` D-Bus example: *http://media.wiley.com/product_ancillary/30/04717761/DOWNLOAD/776130code12(freedesktop).tar.gz*
[12] Other Flatpak examples: *http://flatpak.org/developer.html*
[13] Snappy Ubuntu Core: *https://developer.ubuntu.com/core*
[14] Snapcraft: *http://snapcraft.io*
[15] Installing Snapcraft und Snapd: *http://snapcraft.io/docs/core/install*
[16] Snap documentation: *http://snapcraft.io/docs/build-snaps/*
[17] Snap interfaces: *http://snapcraft.io/docs/reference/interfaces*
[18] Snap Store: *https://uappexplorer.com/apps?type=snappy*

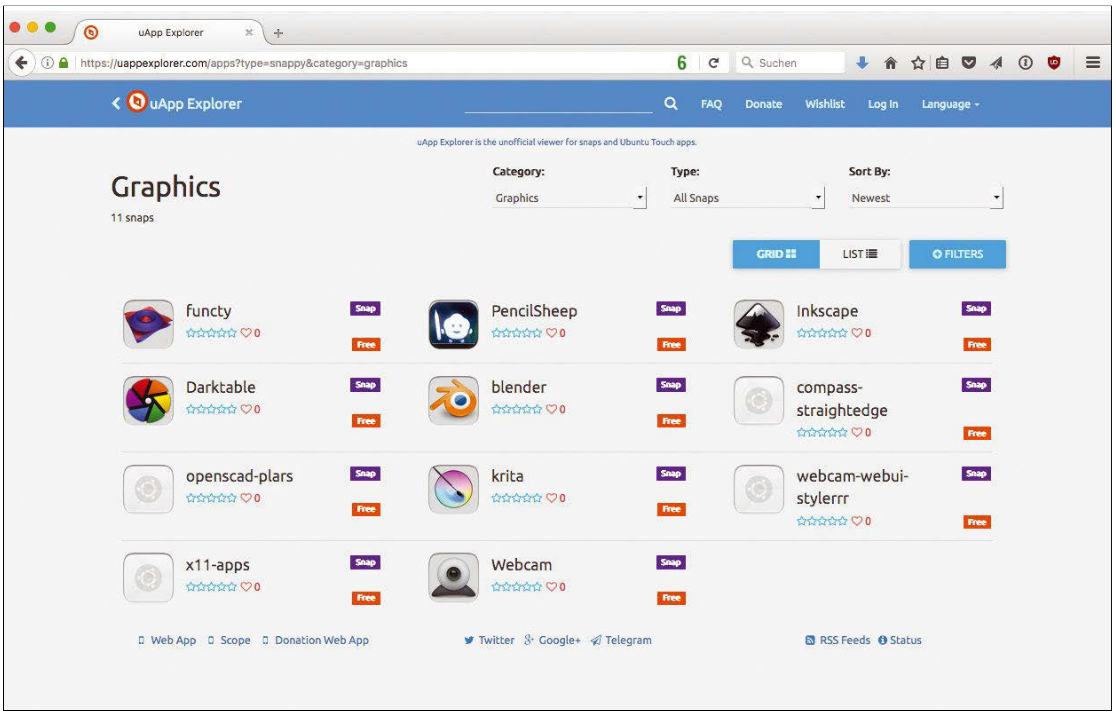


Figure 5: The Snap Store already contains various ready-made snaps that users can install at the click of a mouse.

Input Club

Open Keyboards with Style

The open source keyboard community has been hard at work developing high-quality, customizable keyboard firmware. *By Bruce Byfield*

Many of us spend eight hours or more each day at a keyboard, but keyboards have hardly evolved for decades. However, an open source keyboard community is working not only to change that, but to sell its hardware commercially under the name of Input Club [1]. The result is some of the highest quality, most aesthetic keyboards available today.

Unknown to most people except the participants, the keyboard community has been active for some years, designing and releasing keyboard firmware, printed circuit board schematics, and case designs. Andrew Lekashman, one of the founders of Input Club, lists TMK [2], QMK [3], and EasyAVR [4], as well as Input Club itself [5], as firmware developers. Input Club also develops the Keyboard Layout Language [6] "in the hopes that we could improve the overall user interface device ecosystem," Lekashman adds.

One of the first products released by the keyboard community was the ErgoDox split keyboard (Figure 1) [7], created by Dominic Beauchamp. Although not originally open source, the ErgoDox "gave us insight into the fact that people really wanted open source keyboards," Lekashman says, and a refined version of the ErgoDox has since been released as open hardware by Input Club [8].

Input Club itself got its start when Lekashman was working at Massdrop [9], a website that sells products built by online communities. Lekashman was responsible for a number of Massdrop communities, including tech, 3D printing, and e-cigarettes, but says that "I found myself, as a Bay Area startup person, using keyboards the most often. I started helping coordinate meetups with [programmer]

Lead Image © Bruce Rolff, 123RF.com

BRUCE BYFIELD

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

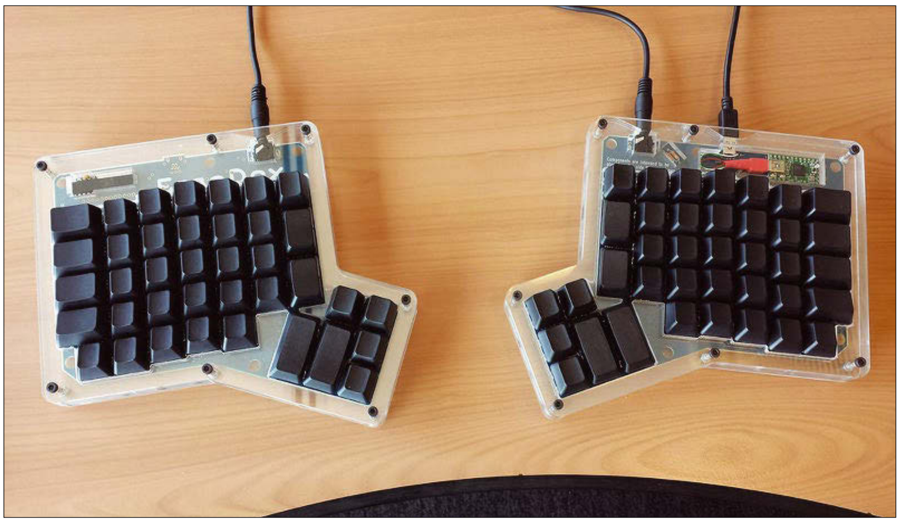


Figure 1: The ErgoDox, on of the first open source keyboards built by the community.

Figure 2: The K-Type is Input Club's first keyboard for general users rather than hobbyists.

Jacob Alexander [10], and one day we decided to build a keyboard together [and] assembled a team of people who really cared about what they were doing."

At first, Lekashman remembers, "we were just a group of people, but Massdrop's finance department wasn't comfortable sending checks for prototyping and supplies to a nonentity."

Today, Massdrop promotes and sells Input Club, and the two companies are distinct entities, although they continue to work together. The relationship appears to work – Input Club's K-Type keyboard (Figure 2), which was released on May 16, 2017, sold over 2,700 units in its first month, making Input Club one of open hardware's unsung success stories.

The Secret Is in the Switches

Input Club currently sells four keyboards: the Infinity ErgoDox Kit [11], the Infinity 60% kit [12], the WhiteFox 65% [13], and the new K-Type 60% [14]. The percentage refers to the size compared to a full-sized keyboard: The 65% keyboards include arrow and navigational keys but no number pad, while the 60% ones have neither set of keys.

The kits must be assembled by buyers using a soldering iron and Input Club's online videos. Both the backlights (Figure 3) and the keys themselves on all models can be configured for characters and macros using the web-based configurator (Figure 4) [15]. All models work on Linux, Mac OS, and Windows.

Besides being programmable, all Input Club's offerings are designed to be attractive to the eye, although none are as elaborate as Datamancer's steampunk keyboards [16], or as configurable as those being developed by Keyboardio [17]. Instead, the K-Type prototype I received for review has a minimal, Apple-like aesthetic, with white keys mounted on brushed aluminum. According to Lekashman, a shim on the bottom for angling the keyboard is optional.

However, what really distinguishes Input Club's keyboards is the attention paid to the mechanical switches used for the keys. Almost all keyboards available in stores are rubber dome, aka membrane keyboards [18], whose keys are pressure pads. However, hardcore typists and gamers have long preferred mechanical keyboards, whose keys each have a separate mechanism and offer tactile – and sometimes aural – feedback, as well as being more durable and repairable. Yet even among mechanical keyboards, Input Club's are somewhat special.

As Lekashman explains, "The most common form of mechanical switch, the Cherry MX style switches we use in our keyboards, have the benefit of a long 'travel' or distance the switch moves when you type. All too often, people overpress their keys when typing, bottoming out the key and applying stress to their fingers by repeatedly hitting the bottom of their keyboard. However, most switches 'actuate', or send a signal to the computer at the halfway mark, and it

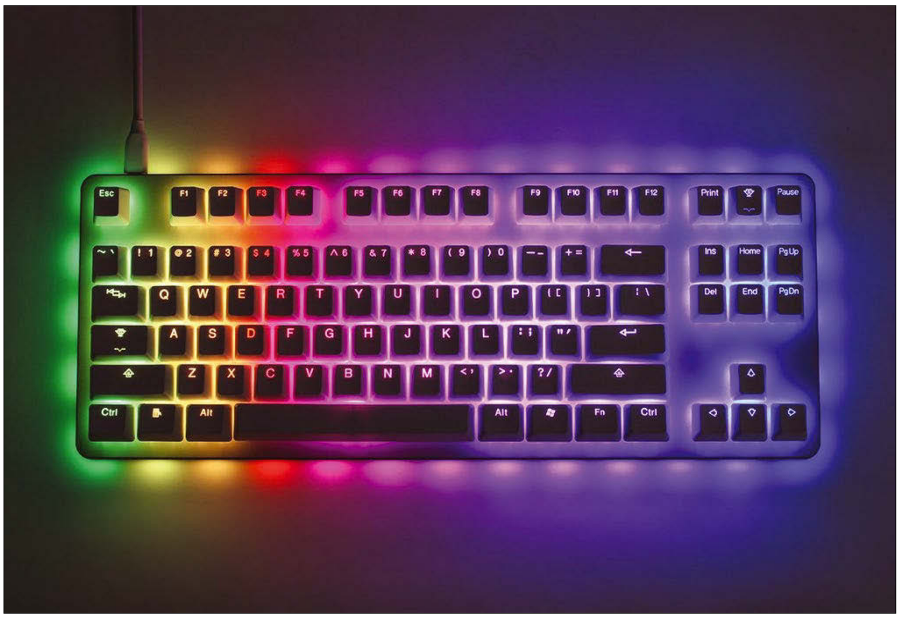


Figure 3: The K-Type comes with configurable backlights.

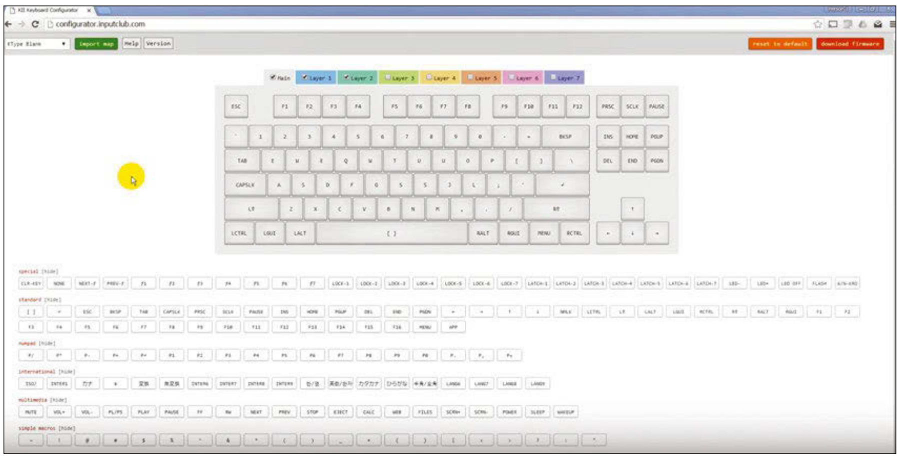


Figure 4: Input Club's online configurator customizes characters and backlights for each key, as well as macros.

really isn't necessary to continue pushing the key all the way down. Some switches provide a tactile bump right before acuation, while others [may] include a loud click to let you know when you should release a key. Many people spend most of their day at a computer typing, so investing in a keyboard with a component that gives a longer life span, a more pleasant feel, and a superior experience overall tends to make sense."

Even so, Lekashman says, "Switches are the most important and most often ignored aspect of keyboards, even by companies that make mechanical keyboards." Before picking the switches to use, Input Club tested "all of the available options" extensively [19]. The resulting review by Jason Alexander [20] begins by observing that most reviews of switches are vague or rely on analogs at best, and goes on to detail the building of a gauge to accurately measure the force required to use different switches.

With the K-Type keyboard, Input Club has released its own switches: Halo True [21] and Halo Clear [22]. Manufactured by Kaihua and design by Alexander, both the Halo switches are designed as an improvement on the popular Cherry MX switch. Input Club claims that the Halo True is designed to greatly reduce the friction when a key is first pressed, making the press a single smooth action. By contrast, the Halo Clear reproduces the firm and smooth tactile feeling of the Cherry MX Clear switch.

Lekashman explains that the Halo switches have three advantages. First, they "dramatically reduce the preload, so that when you first press a key, there is little to no resistance." Second, they reproduce the minimal effort required to depress a key found in a Topre keyboard in a Cherry MX-like switch. Third, Lekashman says, they "add a special spring that makes it difficult to fully depress the key all the way to the base of the keyboard." When they buy, aficionados can choose to buy a standard switch or the Halo model they prefer after investigating the specs, while nonexperts can trust Input Club simply to make typing easier – and, quite possibly, reduce the chances of repetitive stress injuries.

Upcoming

Although the K-Type has just been released, Input Club is already developing a full-sized keyboard named Kira in partnership with designer Angelo Tobias. Those interested in Kira can sign up to receive updates on its development [23].

"After Kira," Lekashman says, "we intend to work on a trackball, as they are an input device that hasn't received as much attention in the last 20 years as they really ought to. All of our keyboards have mouse control as a feature, and we intend to make use of a mouse and use of a keyboard a seamless and fully controllable experience."

With a price of $199 from Massdrop, and a suggested retail price of $299, a keyboard like Input Club's K-Type is several times more expensive than most rubber dome keyboards. However, the technology, aesthetics and sturdy construction of Input Club keyboards make them worth considering. As Lekashman says, "having the right tool for the job makes all the difference, and using the right keyboard can improve every workday. Keyboards are very much a matter of personal taste, which is why we put so much effort into making our keyboards so customizable. One day, it is our sincere hope that our work developing open source keyboard hardware will push the entire industry forward." Meanwhile, Input Club must be considered an early open hardware success story. ■■■

INFO

[1] Input Club: *https://input.club/*

[2] TMK firmware: *https://github.com/tmk/tmk_keyboard*

[3] QMK firmware: *https://github.com/qmk/qmk_firmware*

[4] EasyAVR firmware: *https://geekhack.org/index.php?topic-51252.0*

[5] Input Club firmware: *https://github.com//kiibohd/controller*

[6] Keyboard Layout Language: *https://github.com/kiibohd/kll-spec*

[7] Original ErgoDox keyboard: *https://deskthority.net/wiki/ErgoDox*

[8] Input Club ErgoDox keyboard: *https://ergodox.input.club/*

[9] Massdrop: *https://www.massdrop.com/*

[10] Jacob Alexander: *https://input.club/about/*

[11] Infinity ErgoDox keyboard: *https://input.club/devices/infinity-ergodox/*

[12] Infinity keyboard: *https://input.club/devices/infinity-keyboard/*

[13] WhiteFox keyboard: *https://input.club/whitefox/*

[14] K-Type keyboard: *https://input.club/k-type/*

[15] Configurator: *https://input.club/configurator/*

[16] Datamancer: *https://datamancer.com/*

[17] Keyboardio: *https://shop.keyboard.io/*

[18] Membrane keyboards: *https://en.wikipedia.org/wiki/Membrane_keyboard*

[19] Mechanical switches comparison: *https://input.club/the-comparative-guide-to-mechanical-switches/*

[20] Mechanical switches review: *https://input.club/the-problem-with-mechanical-switch-reviews/*

[21] Halo True switch: *https://input.club/the-comparative-guide-to-mechanical-switches/tactile/halo-true/*

[22] Halo Clear switch: *https://input.club/the-comparative-guide-to-mechanical-switches/tactile/halo-clear/*

[23] Kira waiting list: *https://input.club/kira-waitlist*

Get the SNIA Member rate on registration by registering at www.snia.org/sdcode and using code SDC2017200.

Don't Miss This! Register by August 11th to save $300!

Storage Developer CONFERENCE

Santa Clara, CA, September 11-14, 2017

SDC 17

Produced since 1998, SNIA's Storage Developer Conference (SDC) is the only event created **BY** storage developers **FOR** storage developers.

Register at www.snia.org/sdcode by August 11th and save an additional $300.

Register to attend SDC for technical discussions and education on the latest storage technologies and standards.

Spend 4 days with your peers, and take advantage of the opportunity to learn from experts all in one place.

If you sell a product or service to storage developers, consider customizing a sponsorship package for SDC 2017.

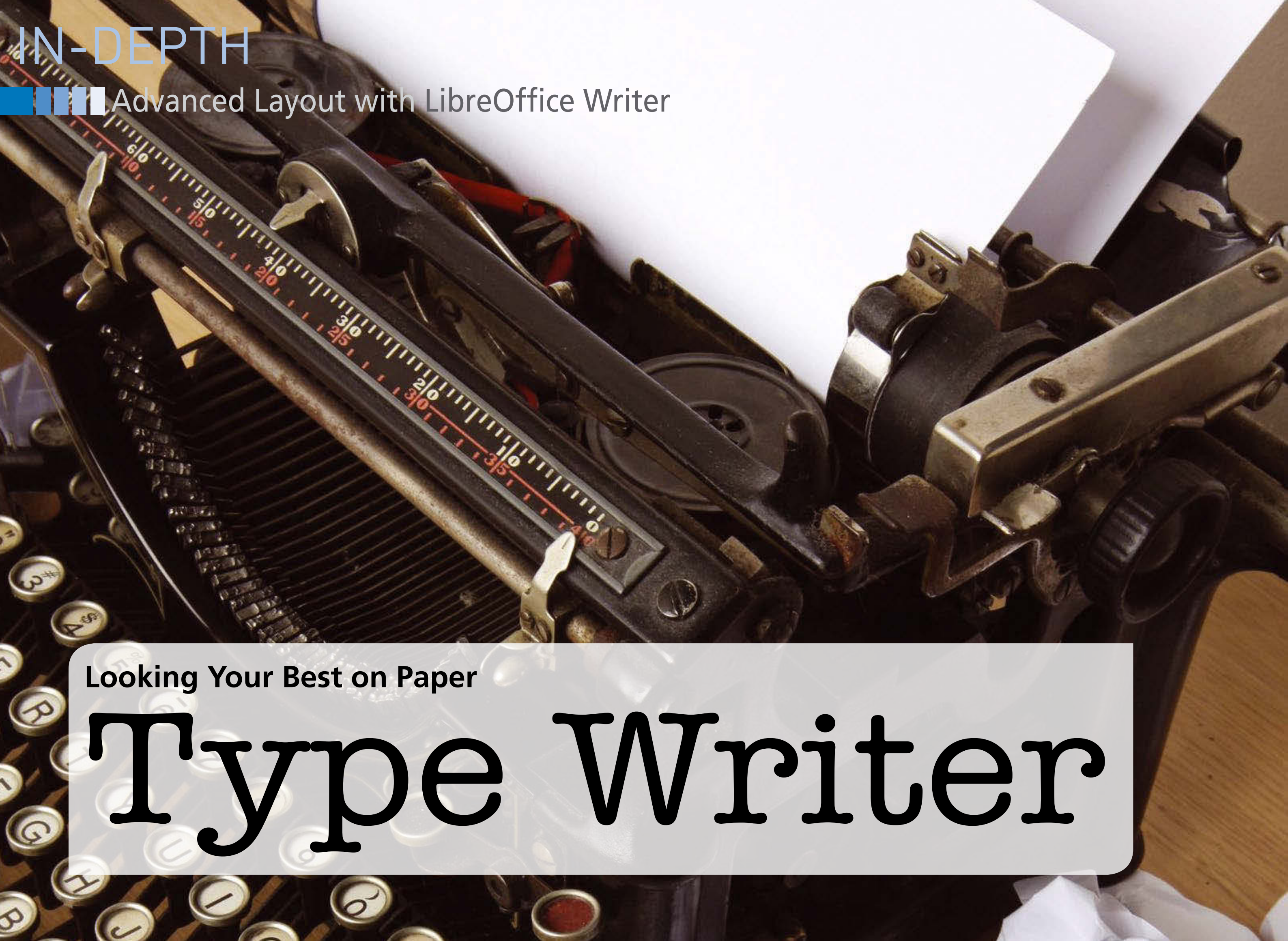
Packages include speaking and exhibiting opportunities and group registrations.

Download and watch free content from SDC 2016.

Presentations topics include Persistent Memory, Solid State Storage, File Systems, Cloud Storage, Storage Management, and more.

www.storagedeveloper.org

SNIA®



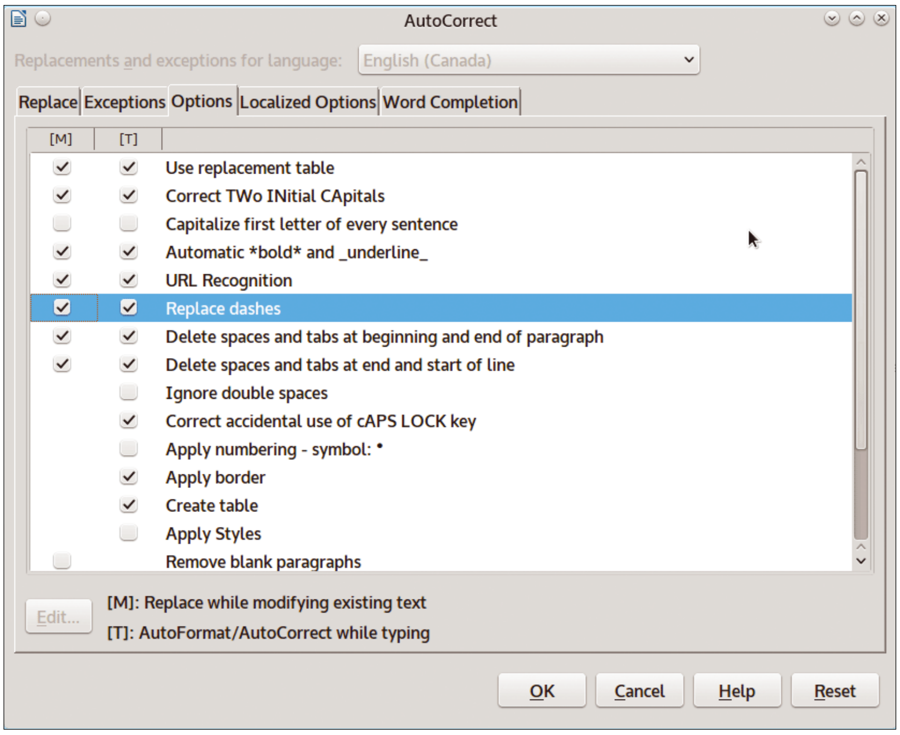
Looking Your Best on Paper
Type Writer

Some documents deserve extra attention to design and typographic detail. *By Bruce Byfield*

LibreOffice Writer does not default to advanced typography, probably because several decades of word processors has allowed a more relaxed standard for most documents. However, the days of indicating italics by underlining – a relic of the limitation of typewriters – are long past. Although most users are unaware of the fact, Writer can format text almost as well as a professional print shop. With a few default settings and some care, Writer becomes more of a desktop publisher than a word processor, transforming your documents so your words are presented to maximum advantage.

Setting Advanced Options

If you want to produce professional-looking documents in Writer, start with the options in *Tools* | *AutoCorrect Options*. In addition to numerous time-saving options, the *Options* tab has a checkbox to *Replace dashes* (Figure 1). This option automatically corrects the common habit of



AutoCorrect

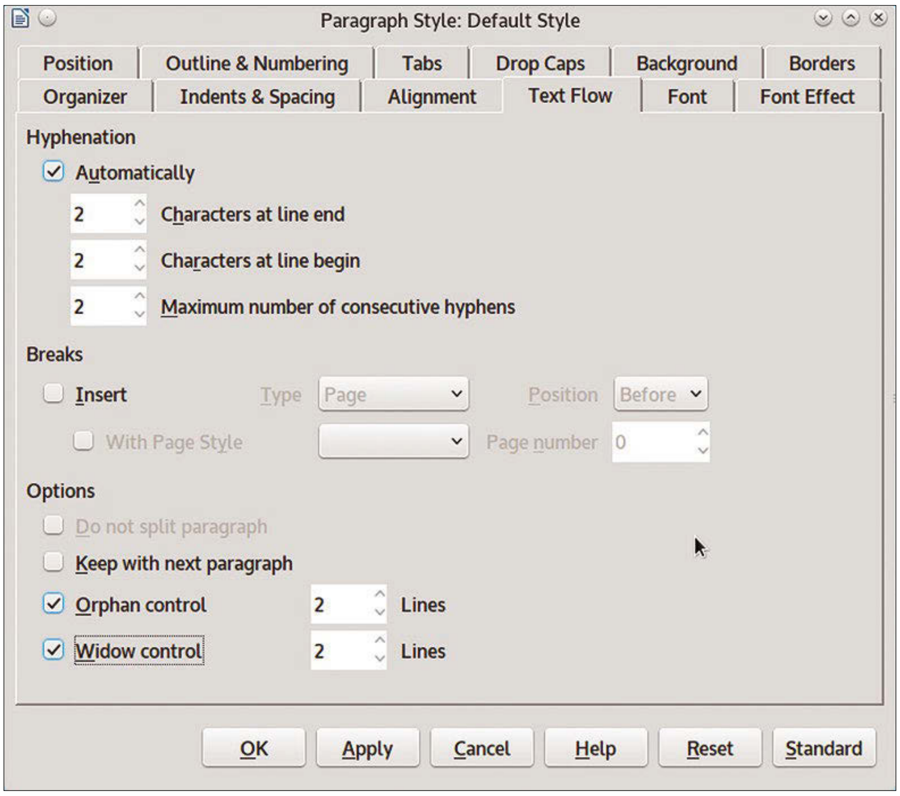
Replacements and exceptions for language: English (Canada)

Replace | Exceptions | Options | Localized Options | Word Completion

[M]	[T]	
✓	✓	Use replacement table
✓	✓	Correct TWo INitial CApitals
☐	☐	Capitalize first letter of every sentence
✓	✓	Automatic *bold* and _underline_
✓	✓	URL Recognition
✓	✓	Replace dashes
✓	✓	Delete spaces and tabs at beginning and end of paragraph
✓	✓	Delete spaces and tabs at end and start of line
	☐	Ignore double spaces
	✓	Correct accidental use of cAPS LOCK key
	☐	Apply numbering - symbol: •
	✓	Apply border
	✓	Create table
	☐	Apply Styles
☐		Remove blank paragraphs

[M]: Replace while modifying existing text
[T]: AutoFormat/AutoCorrect while typing

Figure 1: AutoCorrect includes several options that improve the typography of Writer documents.



Paragraph Style: Default Style

Position | Outline & Numbering | Tabs | Drop Caps | Background | Borders
Organizer | Indents & Spacing | Alignment | Text Flow | Font | Font Effect

Hyphenation
- [✓] Automatically
- 2 Characters at line end
- 2 Characters at line begin
- 2 Maximum number of consecutive hyphens

Breaks
- [ ] Insert — Type: Page — Position: Before
- [ ] With Page Style — Page number: 0

Options
- [ ] Do not split paragraph
- [ ] Keep with next paragraph
- [✓] Orphan control — 2 Lines
- [✓] Widow control — 2 Lines

OK | Apply | Cancel | Help | Reset | Standard

Figure 2: If you use hyphenation, the *Text Flow* tab for paragraph styles can help you use them more professionally.

replacing two hyphens with an en dash (see below for comments on whether you should use this feature). Another useful option, this time on the *Localized Options* tab, are checkboxes for replacing single and double straight quotes (' and ") with rounded smart quotes (‘ and “), of the kind used in professional publishing.

If you are using paragraph styles (and if you are interested in formatting well, you ought to be), you can control hyphenation of lines from each style’s *Text Flow* tab. You can usually leave the defaults of two characters at the end and start of a line, but you also should set *Maximum number of consecutive hyphens* to 2, which is the typographical norm (Figure 2).

While on the *Text Flow* tab, you should also enable the defaults of 2 for both widows (lines in a paragraph that fall at the top of the page) and orphans (lines in a paragraph stranded at the bottom of the page). If you are using a *Justified* alignment, which fills the entire space between the left and right margins, adding spacing as needed, you should also check that the *Last line* option on the *Alignment* tab is set to *Left*; otherwise, a short last line will have huge gaps between characters and words.

Hidden elsewhere in the menu are features for controlling hyphenation at the end of the line. These features are especially useful if your document has a *Justified* alignment so that you can hand correct awkward spacing. *Insert* | *Formatting Mark* has a submenu from which you can either prevent a line break with a *Non-breaking space* or a *Non-breaking hyphen*, or else force a hyphenated line break with an *Optional hyphen*.

Later, as you are putting the finishing touches on a document, run *Tools* | *Language* | *Hyphenation* just as you would *Tools* | *Spelling and Grammar*. The automated hyphenation does its best, but after you are finished rearranging text, it may not be optional, and running *Hyphenation* allows you to improve its best guesses.

Best Practices

Other advanced typography depends on the choices you make while formatting. Ideally, you want to use fonts that include these advanced features. Even better, you want to use fonts that are Graphite-enabled and will add many of these features automatically in the 5.3 release, or with the Typography Toolbar extension installed in earlier releases (Figure 3). Unfortunately, only a few fonts are Graphite-enabled, which means that you may have to insert features manually using *Insert* | *Special Character* or by choosing a font style from the *Font* tab of a Character style.

Formatting features that you might want to include:

- **Small Capitals.** Small caps are a font style or separate font designed completely separately from normal capital letters (Figure 4). They are used whenever more than one capital letter appears in a row, such as in abbreviations, because they fit into the rest of the text better. If your fonts lack small capitals, you can sometimes create your own with a special character style, but never use the small caps created automatically from *Font Effects* | *Effects* | *Small capitals* – they are an awkward kludge that looks nothing like either normal capitals or small capitals.
- **Ligatures.** Ligatures are special characters used to replace common combinations of letters with either awkward or poor spacing, such as ll or ff (Figure 5). The Typography Toolbar has separate buttons for *Standard ligatures*, *Historic ligatures* (apparently, overly elaborate ones), and *Discretionary ligatures* (apparently, ones that not everyone will want), although which ligatures belong to which class is not always obvious. Ligatures do not exist with monospaced fonts, for the simple reason that, when all letters occupy the exact same space, by definition there is no variable spacing that can be tweaked. However, even in variable spaced fonts, not every awkward combination of letters has a ligature to replace it, and some combinations are only awk-

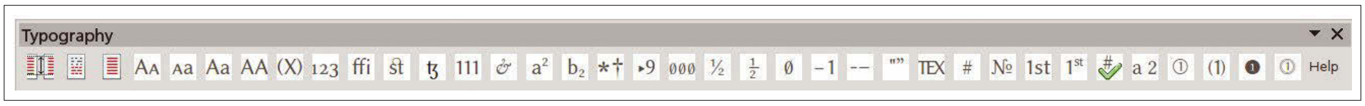


Figure 3: The Typography Toolbar extension helps you take full advantage of the few Graphite-enabled fonts that are available.

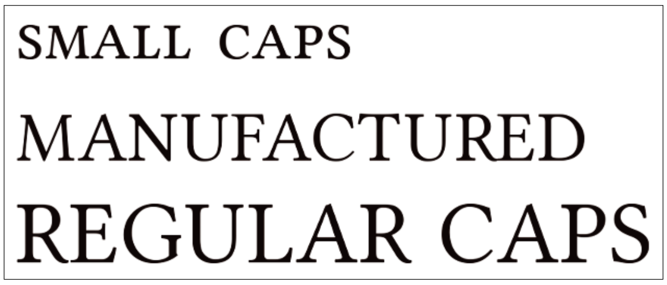


Figure 4: True small capitals (top), manufactured small capitals (middle), and regular capitals (bottom). Notice how the capitals manufactured by Writer look nothing like true small capitals.

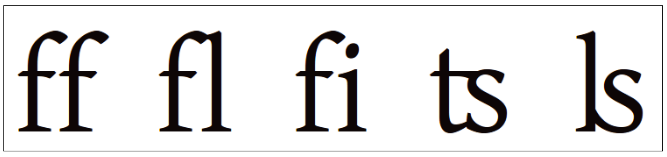


Figure 5: Ligatures are separate characters for letter combinations that are awkwardly spaced.

ward with a particular font. In these cases, you can create a character style from *Format* | *Character* | *Position* | *Spacing* to alter the spacing between selected letters.

- **Old-Style Figures.** Today, we are used to using numerals that have a common baseline. These numerals are known as ranging figures or proportional numbers. By contrast, each old-style figure has its own baseline (Figure 6). Old-style figures are considered more suitable for text and work especially well with small capitals. The only trouble is, the shifting baseline can be confusing to modern readers, especially in tables and spreadsheets, and can appear awkward beside regular capitals. For these reasons, the decision to use old-style figures should be on a font-by-font basis and not made simply because of the idea that they are more elegant.
- **Diagonal and Nut Fractions.** Fractions are often written with three separate characters: the numerator, a dividing forward slash, and the denominator. This practice is quick and convenient, but not easy to read. Diagonal and nut fractions replace these three characters with a single character. A diagonal fraction uses a forward slash to separate the numerator and denominator, whereas a nut fraction uses a horizontal line. Generally, both types are available only for common fractions like one-half and one-quarter. You might also want to avoid them when using a font of eight points or less, in which they might be too small for easy reading.
- **Em and En Dashes.** As the names suggest, an em dash occupies the space of a letter m, and an en dash the space of a letter n. The two dashes are often used interchangeably to set off an aside from the rest of the sentence. Just as often, two en dashes are used as a quick alternative to an em dash, and, as already noted, LibreOffice can automatically correct two hyphens to an en dash. However, some typographers consider an em dash inelegant and prefer to use an en dash with a space on either side of it instead. An en dash is also used to separate a range of numbers – for instance 3–4pm. It is distinct from both a hyphen and a minus sign, although in casual usage, they are usually lumped together because they look similar.
- **Borrowed Characters.** Sometimes, designers choose to replace a font's characters like ampersands (&) or question marks (?) with more stylistic versions from another font. Nothing is wrong with this practice, and the variations on what most people imagine are standard characters are surprising in number (Figure 7). The substitution can be made through the creation of a Character style, but remember: A little can go a long way. Usually, the result is less overwhelming if only one or two substitutions are made. In the same way, you may not want to use the italic or oblique style designed to accompany a regular font style. Sometimes, an italic is unimaginative, while an oblique style is simply the regular style given an angle of 45 degrees, a horror of an afterthought that is not sufficiently redesigned. Because an italic or oblique can look very different from a font's regular style, these substitutions are often easy to make.

Your Best Foot Forward

Some users might wonder what the point of advanced typography might be. Generations of typewriter users created conventions that were convenient but inelegant compared with typographical best practices. Moreover, the original Linux users were developers, which created a different set of priorities geared to working on the command line.

Those raised with these conventions might argue that they are good enough and that advanced typographical standards serve no purpose. And for quick, one-time documents, they are often right. However, if you are arguing a point of view or writing a document that will be used for years, a layout that makes for easy reading and is pleasantly designed becomes more important. An advanced design reflects on you and your thoughts, so the time taken to look your best is never wasted. ■■■

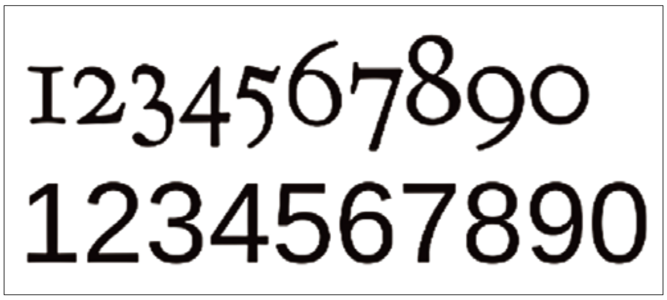


Figure 6: Old-style figures (top) are numerals designed to be placed within text.

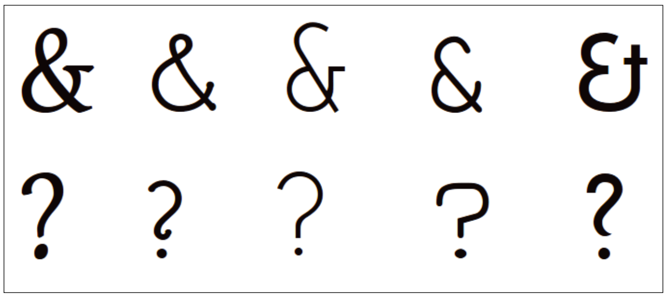


Figure 7: Sometimes, you can jazz up a font by borrowing characters from another font.

IT Highlights at a Glance

Too busy to wade through press releases and chatty tech news sites? Let us deliver the most relevant news, technical articles, and tool tips – straight to your in box. Subscribe today for our excellent newsletters:

- ADMIN HPC
- ADMIN Update
- Linux Update
- Raspberry Pi

and keep your finger on the pulse of the IT industry.

Admin and HPC:	www.admin-magazine.com/newsletter
Linux Update:	www.linuxpromagazine.com/mc/subscribe
Raspberry Pi:	www.raspberry-pi-geek.com/mc/subscribe

The sysadmin's daily grind: Brewing helpers

Free as in Beer

Columnist Charly looked into so many mash tubs during brewery tours that he wanted to try his own home brew. He did a little research and found some open source projects that could help. *By Charly Kühnast*

Everything a student of the fine art of brewing needs is likely already there in the kitchen; if not, you can get it for very little money at your local hardware store – expensive special equipment is not necessary. All you need now is a recipe: For my first experiments in my legal drug laboratory, I resorted to a ready packaged set, in which I found brewing malt, hops, and yeast in the right proportions. I only had to home brew, and I did so without automation, which is otherwise my hobby.

I thus created the mash by hand, filtered, poured in some more water, boiled the hops, then cooled, and decanted the results into a fermentation tank in the hope that the yeast would do its magic – all of this took around four and a half hours. The wort – this is the official term for the proto-beer – now needs a good week before I can bottle it. I used the time to search for open source projects that could assist me in brewing.

I started totally from scratch, that is, with the recipe and was totally amazed to discover that there's a kind of standard even for this: Beer XML [1], a description language for exchanging brewing recipes in a standardized format. Today almost every brewing software relies on Beer XML.

The quest for subtleties of home brew artistry inevitably take you to Beersmith [2]. The proprietary software is subject to a charge after a trial period, and it runs on Linux machines, in addition to Windows and Mac OS. The manufacturer supports Ubuntu, but the program also runs on many other distributions.

Beersmith relies on its database containing a five-digit number of brewing recipes, but the various calculators are even more amazing. Imagine you are putting together your own recipe: Beersmith calculates the required proportions on the basis of your ingredients and other parameters and forecasts the resulting alcohol content, bitterness, quantity of carbonic acid, and much more.

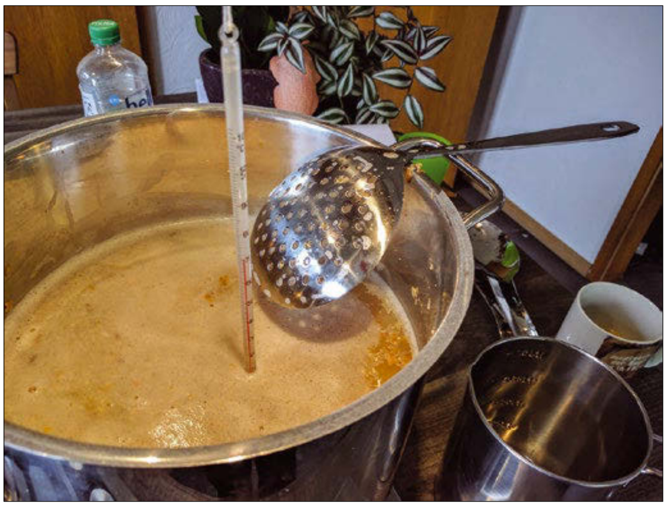


Figure 1: Demands full attention – first-time brewer Charly had to manually keep the mash at a temperature between 65 and 69 degrees Celsius for one hour in his kitchen.

Free as in Beer

There is also an open source software tool that does something similar. Brewtarget [3] is the leader of the pack here, available in Deb and RPM packages and for Windows and Mac OS for portability. Of course, Brewtarget imports and exports Beer XML and predicts many parameters of the finished beer based on the ingredients, much like Beersmith. The project is funded by donations and the sale of T-shirts.

The process of brewing almost cries out for automation. During mashing, I had to keep the mash at a temperature between 65 and 69 degrees Celsius for one hour (Figure 1). Even though I have an infinitely variable gas stove, that demanded my constant attention. Automation technology is urgently needed here. From a sober point of view, it thus came as little surprise that solutions typically relying on an Arduino microcontroller or Raspberry Pi can be used as a control unit. The Craftbeer PI [4] project is just one example among many.

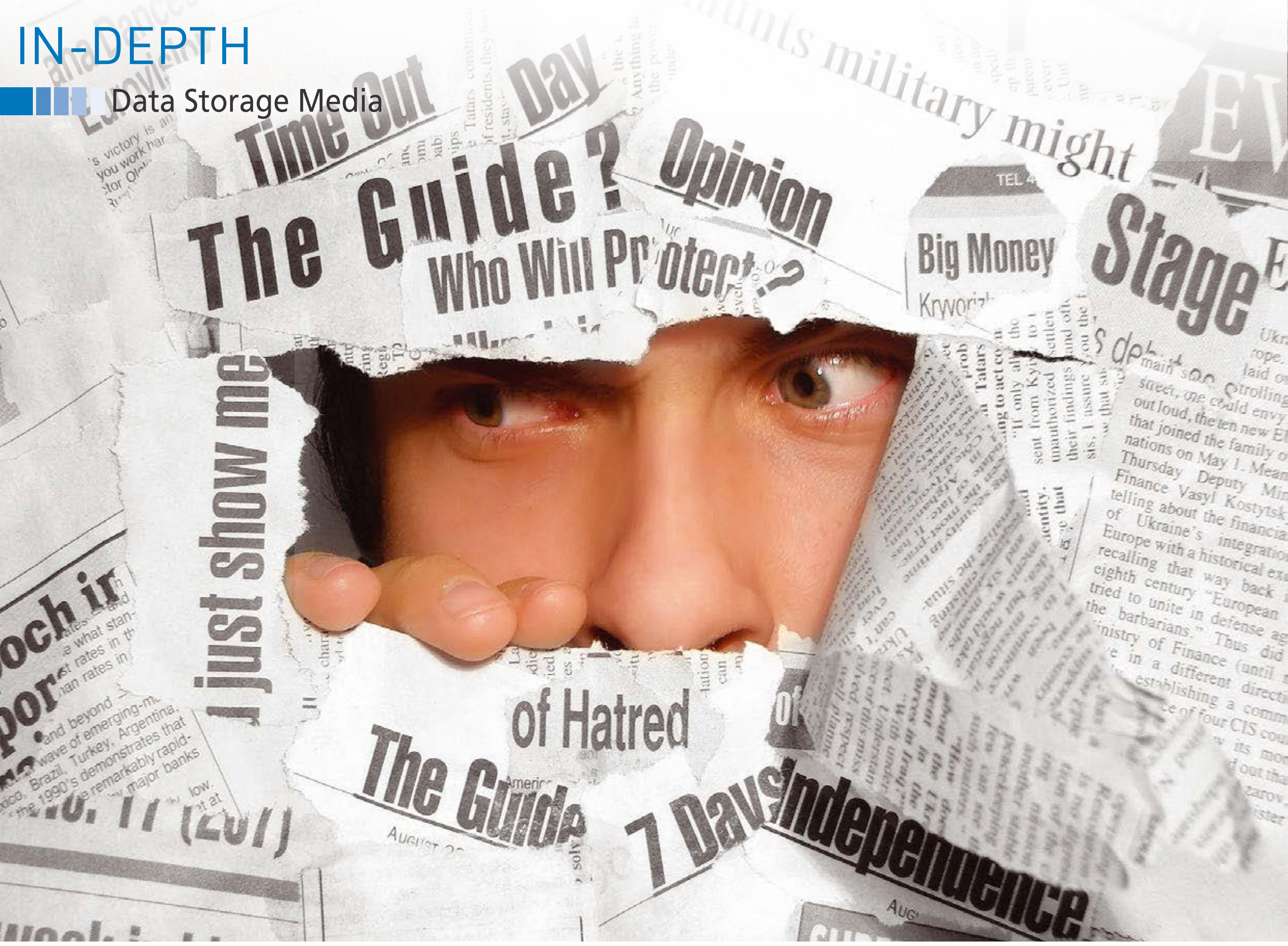
What did I learn by keeping away from the liquor store? Brewing beer is something for people with patience who are not afraid to stand in front of the stove for hours waiting for a four-liter hop smoothie to develop. Will I do it again? Definitely. And, I'll let you know all about it. ■■■

INFO

[1] Beer XML: *https://en.wikipedia.org/wiki/BeerXML*
[2] Beersmith: *http://beersmith.com*
[3] Brewtarget: *http://www.brewtarget.org*
[4] Craftbeer Pi: *http://web.craftbeerpi.com*

CHARLY KÜHNAST

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



Test real and fake disks for bad blocks

Fake Media

When it comes to cheap flash memory, buyers should beware. Fake flash memory often offers only a fraction of the advertised storage capacity. With no visible calibration mark, discerning counterfeits is problematic. Here's a test to weed out fake disks. *By Rolf Freitag*

Any customer in a bar who ordered a pint of beer would definitely complain to the barkeeper if their glass was only a quarter full. This kind of attempted fraud always fails in the real world because of the obvious lack of fluid in the glass, along with the high level of motivation of the consumer and the physical presence of the perpetrator.

Unfortunately, online buyers of flash storage media cannot hope for comparably favorable conditions, although their judgment is unlikely to be impaired by the effects of alcohol. Time and time again, you hear of buyers of USB flash drives and flash cards being duped by shady dealers on the Internet. It is not difficult to find media at surprisingly low prices where the storage capacity is wrongly stated, for example, on eBay [1, 2]. To protect themselves, sellers often make statements such as "the stick will store about 4GB of data."

In this article, I attempt to reveal these counterfeits by running the right tests. As test specimens, I use two Samsung Pro 128GB MicroSDXC memory cards (model MB-MG128) purchased in 2016 from two vendors on eBay.de. If you believe the label, they both should achieve up to 90MB/s read and 80MB/s write performance.

Look Closely

Starting with the original packaging shown in Figure 1, a visual inspection reveals interesting differences in the detail. In the case of the first card (shown on the right in Figure 1), the barcode sticker has a spelling mistake: The "o" in "Model" is missing (Figure 2). Also the (possibly fake) CE marking on the back of the first card – the second (shown on the left in Figure 1) does not have this

Lead Image © Uliya Stankevych, 123RF.com

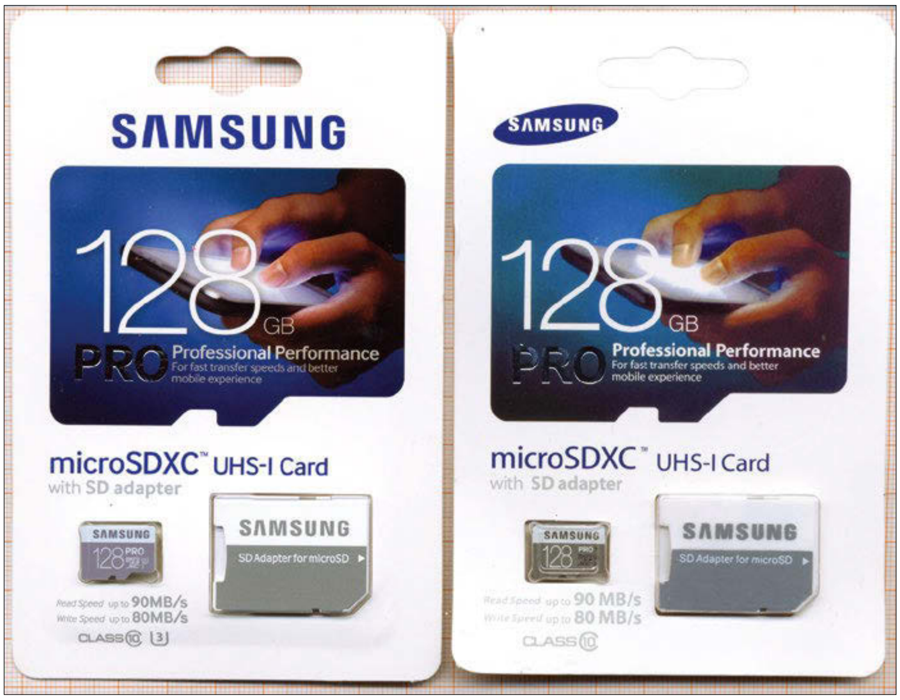


Figure 1: **Pictures of the two packaged memory cards I purchased: on the left what looks like the original product, on the right the potential forgery.**

marking – has for a long time been indicative of a fake [3].

According to the manufacturer, the two cards must achieve speed class 3 (U3), but the label on the first card shows that it is two classes below this. For the second card, a search with the exact model designation, model MB-MG128EU/EU, returned almost 6,000 matches, including the Samsung page with extensive technical information and the EAN 8806088133560. The first card, model MB-MG128D/EU, returned only one match on the website of a Polish vendor.

Reading the Registers

The memory card's metadata provides additional hints, for example, the model, manufacturer, version (revision), and serial number. Linux users can access these data using `hwinfo disc`, unless you are using a USB card reader – in this case, you will only see the reader's metadata.

This unfortunately also applies to the card register, in particular the card-specific data (CSD) register and card identification data (CID) register of SD cards. You can only read these with a direct SD interface.

Peripheral Component Interconnect Express (PCIe) connected card readers in laptops have this (e.g., the Realtek RTS5229 card reader). Or you can retrofit one as a PCIe card (DeLOCK 91485 or mini-PCIe cards from various Chinese eBay sellers).

If you have such a device, the contents of the register are found below `/sys/block/mmcblk0/device/cid` and `.../csd`. In the case of the suspected fake card, the content is:

```
00048d55534220441100001603010301
400e00327b590003e7ff7f800a404001
```

The other card returned different values:

```
1b534d303030303010a3385d7e010901
400e00325b590003b9df7f800a404001
```

If you are investigating a suspicious SSD card, you can also run `hdparm` and `smartctl` to reveal the manufacturer, product, serial number, hours of operation, and interface speed.

Forgers whose engineering expertise is just as great as their criminal energy, reproduce all the values stored in the registers of the original products in a deceptively genuine-looking way. For honest consumers, the task is therefore to look for a reliable strategy to detect tampering. You will find this in a sequential write and read of the whole data storage device without a filesystem using virtually incompressible test data.

To conceal the fake size, flash counterfeiters often manipulate the addressing mechanism of their goods. This gives users the impression that they can address the promised number of blocks and actually own a regular disk. In reality, however, the accessible data blocks are just repeated multiple times. That's why a test with a single test pattern, such as `0xaa`, may not work; instead you will want to test with large and random patterns.

This kind of testing tool is also useful for detecting incipient defects, because even authentic disks can develop faults: The isolation layer of a flash memory cell degenerates on each delete action. In the case of multilevel cell (MLC) NANDs, the original shelf life of about 10 years decreases to about a year after 3,000 erase/write cycles.

Bad Blocks vs. Bad People

The following call implements the outlined procedures and tests for faults and bandwidth at the same time:

```
device=mmcblk0
time badblocks -wsv -c 65536 -t random ↵
    -o log_$device_`date +%F_%T`.txt ↵
    /dev/$device 2>&1 | ↵
    tee logg_$device_`date +%F_%T`.txt
```

The `time` command measures the duration of the test (see Figure 3). The test candidate was a microSD card in the laptop's internal reader (`mmcblk0`). The

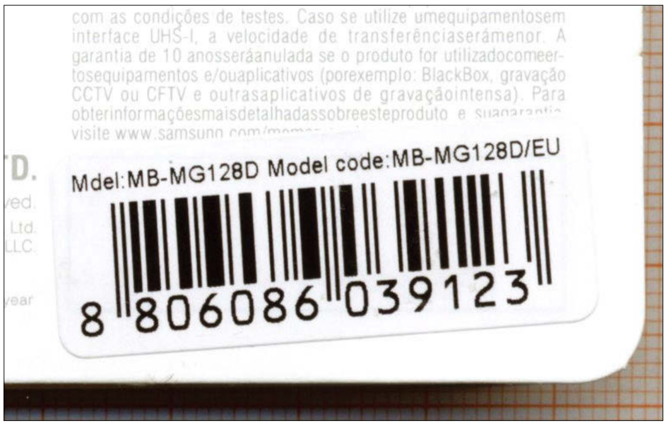
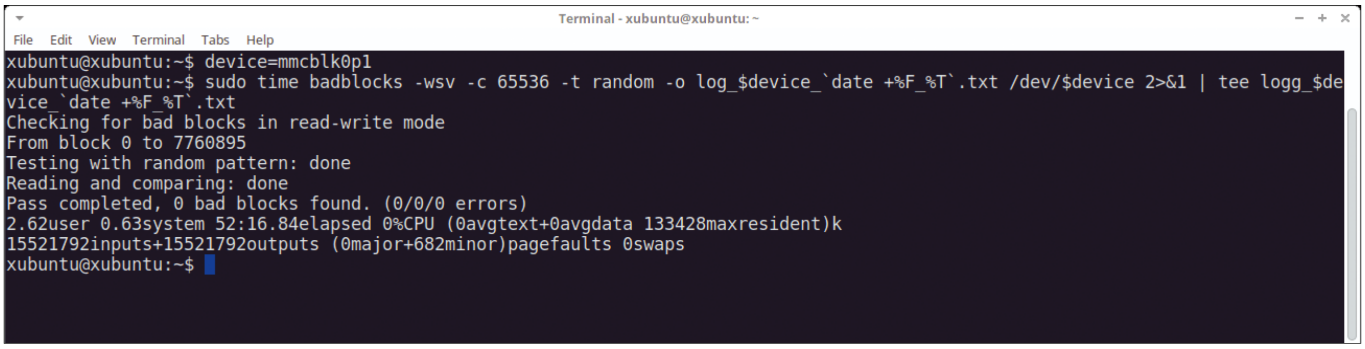


Figure 2: **The barcode on the packaging contains a spelling error.**


Terminal - xubuntu@xubuntu: ~
File Edit View Terminal Tabs Help

```
xubuntu@xubuntu:~$ device=mmcblk0p1
xubuntu@xubuntu:~$ sudo time badblocks -wsv -c 65536 -t random -o log_$device_`date +%F_%T`.txt /dev/$device 2>&1 | tee logg_$de
vice_`date +%F_%T`.txt
Checking for bad blocks in read-write mode
From block 0 to 7760895
Testing with random pattern: done
Reading and comparing: done
Pass completed, 0 bad blocks found. (0/0/0 errors)
2.62user 0.63system 52:16.84elapsed 0%CPU (0avgtext+0avgdata 133428maxresident)k
15521792inputs+15521792outputs (0major+682minor)pagefaults 0swaps
xubuntu@xubuntu:~$
```

Figure 3: A `badblocks` run looking for media defects or manipulation. The `time` command measures the time.

`badblocks` test tool, included in the `e2fsprogs` package, performs the write/read test (`-w`) with a logfile for the numbers of faulty blocks (`-o <file name>`).

It works with a random data pattern (`-t random`) the size of 65536x1024 bytes (`-c 65536`), with a progress indicator (`-s`) and verbose messaging (`-v`). `tee` sends the screen output to a second logfile. To help organize the logfiles, the file names contain both the device name (`$device`) and the date and time (`date +%F_%T`).

If you want to run this test on a data carrier that is already in use, you can use `dd` (or `ddrescue` or `dd_rescue`) to save the contents to a safe place first and write them back after the test. The disk should not be mounted during copying. Because such a test reliably overwrites all previously stored data, it is also useful as a deletion mechanism for old data. (USB sticks are often manufactured with used flash ICs and contain old data, which could include such undesirable information as child porn, top-secret files, and terrorism plans. The write-read test overwrites all old data so that undelete programs cannot recover them, keeping you out of trouble with the authorities.)

To make testing for bad blocks fun, it is important to choose the right target device. Otherwise, the test could overwrite the data on a different disk. I wrote a script that avoids such mistakes: `disktest1.sh` [4]. The only parameter it expects is

LISTING 1: Output of ./disktest1.sh

```
Warning: This should generally be run from group root (Nr. 0), but your group is Nr. 1000 !

   8       64      250880 sde
This script will do a write-read-test with device /dev/sde.
Starting in 15 s (press Ctrl-C if you want to abort).
-
output of lsblk:
NAME           MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda              8:0    0   3.7T  0 disk
|-sda1           8:1    0     1M  0 part
|-sda2           8:2    0    10G  0 part  /boot
`-sda3           8:3    0   3.6T  0 part
  `-sda3_crypt 252:0    0   3.6T  0 crypt /
sdb              8:16   0   1.8T  0 disk
sde              8:64   1   245M  0 disk
loop4            7:4    0   1.8T  0 loop
`-ob_enc4      252:1    0   1.8T  0 crypt /mnt/sbackup
zram0          251:0    0   3.9G  0 disk  [SWAP]
zram1          251:1    0   3.9G  0 disk  [SWAP]
zram2          251:2    0   3.9G  0 disk  [SWAP]
zram3          251:3    0   3.9G  0 disk  [SWAP]
zram4          251:4    0   3.9G  0 disk  [SWAP]
zram5          251:5    0   3.9G  0 disk  [SWAP]
zram6          251:6    0   3.9G  0 disk  [SWAP]
zram7          251:7    0   3.9G  0 disk  [SWAP]
Start date/time: Sun 8 Jan 22:52:46 CET 2017
Searching for defective blocks (read + write)
From block 0 to 250879
Testing with random patterns: done
Read and compare:done
Round finished, 0 bad blocks found. (0/0/0 errors)

real    2m30.322s
user    0m0.390s
sys     0m0.018s

Erledigt, Sun 8 Jan 22:55:16 CET 2017
Done, Sun 8 Jan 22:55:16 CET 2017
```

the name of the data carrier to be tested; in the example above, that's `mmcblk0`, but it's `sde` in the example in Listing 1. The script also works in a Cygwin environment, thus opening up a test option for Windows users. Linux users should run it as root; on Windows, you need to run Cygwin with administrator rights.

Time and Throughput

For a disk with 128GB capacity and an average throughput of 85MB/s, this test takes around 50 minutes. The formula for computing the test duration is `t=2*V/B` (i.e., the time in seconds multiplied by the volume in megabytes divided by the average bandwidth in megabytes per second). The factor of 2 is attributable to transferring twice for writing and reading.

This results in a mean bandwidth of `B=2*V/t`. The exact size of the data medium can be discovered by `lsblk -b /dev/$device`. If you substitute the value into the formula, you can compute the bandwidth in bytes per second.

A buyer can assume a defect or a fake if measuring the data medium reveals a figure of less than a quarter of the "up to" figure stated by the manufacturer, or less than half of the nominal value. However, you do need to use a connection with the maximum speed for the test, for example, a UHS-II reader for a UHS-II memory card.

Happy Ending

Returning to my two 128GB test cards, `bad block` reported 122,973,232 defective blocks from a total 131,071,999 blocks (of 1024 bytes each) for the first card – in other words, 94 percent of the blocks are broken. The second card, on the other hand, showed no errors. According to the formula `t=2*V/B`, the test duration should be about one hour, which was true of the second card, which passed the line after about 70 minutes. The test of the first card took too long – over five hours.

The test with `badblocks` thus exposed the following about the first card: it was a fake with many faults and was incredibly slow. When I confronted the vendor with these hard facts, I convinced him to take back the card and refund the amount paid. The second and intact card, however, works perfectly in combination with a short SD card adapter with a pull tab as the second SSD in my laptop. ■■■

INFO

[1] SD cards/counterfeits: *https://en.wikipedia.org/wiki/Secure_Digital#Counterfeits*

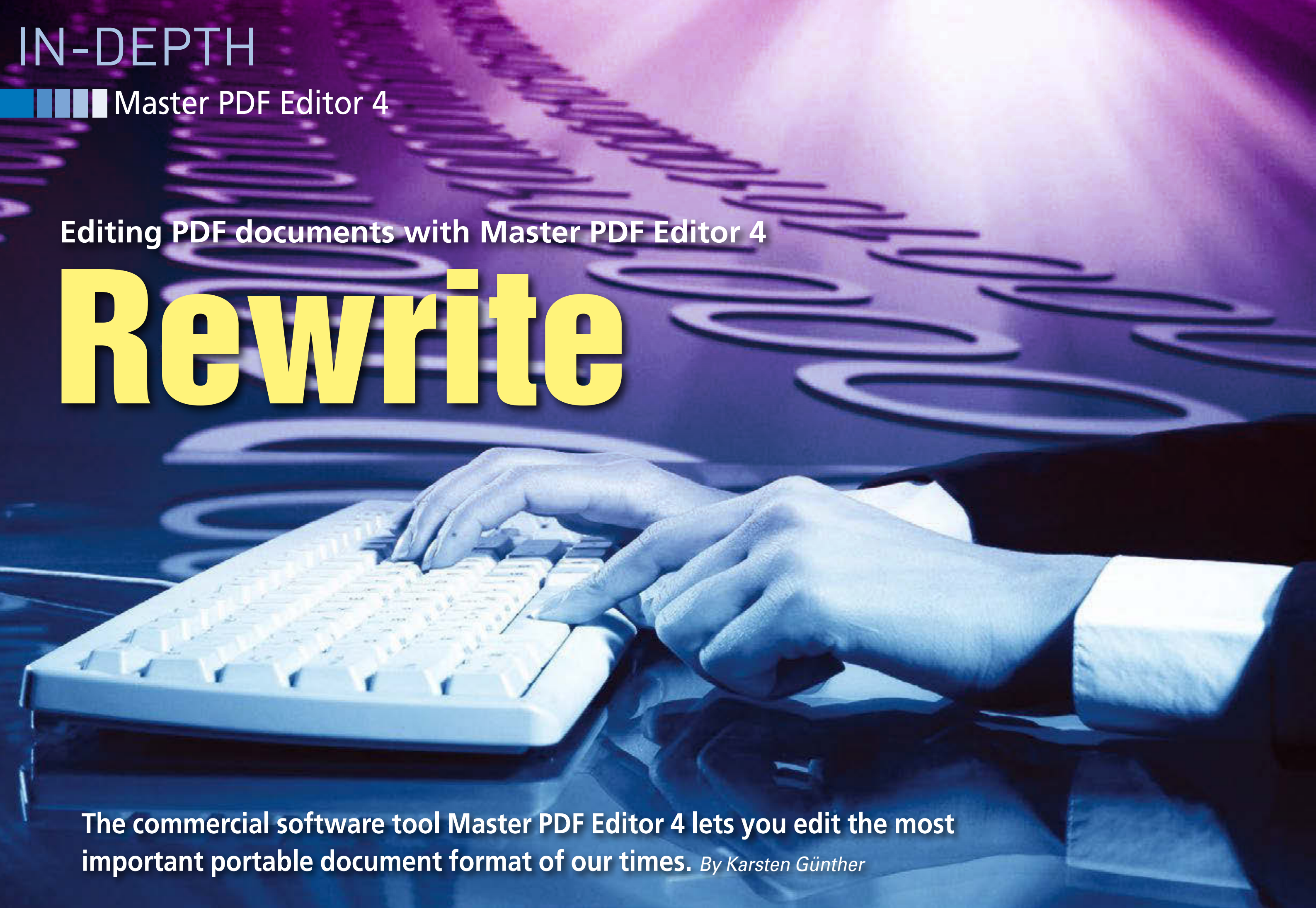
[2] USB storage/manipulated controller chips (German only): *https://de.wikipedia.org/wiki/USB-Massenspeicher#Manipulierte_Controllerchips*

[3] Identifying fakes: *http://linuxwelt.blogspot.de/2014/06/microsd-speicherkarten-falschungen.html*

[4] disktest1.sh: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/201/*

AUTHOR

Rolf Freitag has a PhD in physics and is certified in practical computer science, neural networks, and artificial intelligence. He has been programming since 1995 and is currently working for a small business in Neu-Ulm, Germany.



Editing PDF documents with Master PDF Editor 4

Rewrite

The commercial software tool Master PDF Editor 4 lets you edit the most important portable document format of our times. *By Karsten Günther*

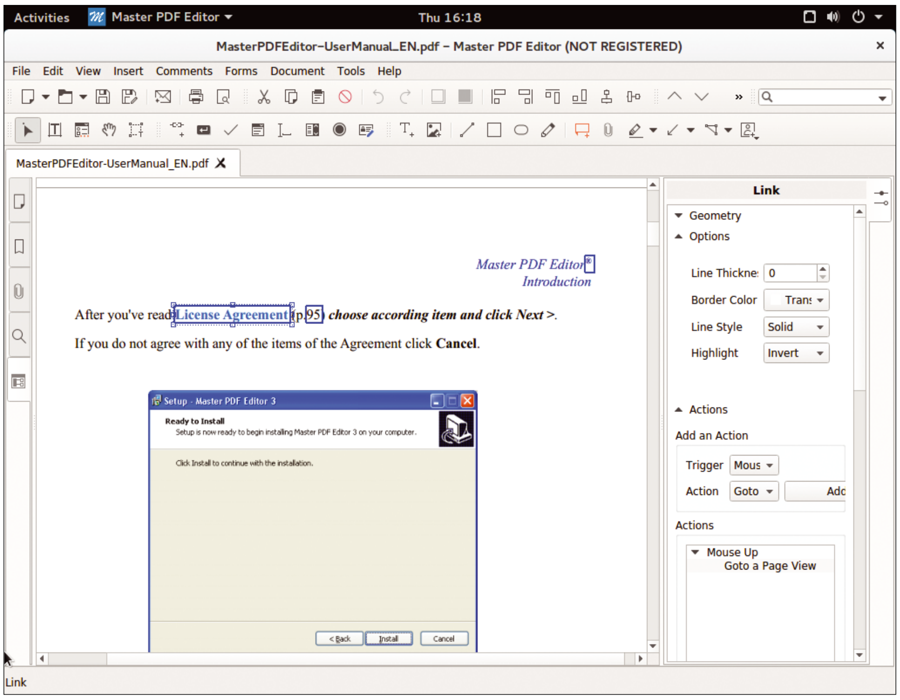
PDF was originally designed as a print format for forwarding and exchanging documents without allowing changes. It is based on the PostScript programming language, a PDF being a compiled – and thus no longer editable – version of the PostScript code. Editing PDF files therefore contradicts the stated aim of the file format: unchanged forwarding.

In practice, you often have good reasons for editing "finished" print files – whether to add minor corrections, replace individual characters, remove short sections of text, or add annotations. Thus, the desire for editors that are able to load, modify, and again save the immutable PDF documents arose at an early stage.

For a time, Ghostscript was the tool of choice: You converted the PDF to PostScript, then edited the source code in a text editor, and finally converted the document back to PDF. Today, hardly anyone uses that method: The documents are too complex, the PostScript code is anything but intuitive,

LICENSE

Master PDF Editor can be downloaded for free from the developer's home page [3]. DEB and RPM packages for 32- and 64-bit systems are available. Alternatively, a `tar.gz` archive with statically compiled binaries can be used independent of your distribution. The license for the free version of the editor allows home use, but does not provide all the available functions. For example, documents cannot be "optimized" (to save space) before exporting. The full version costs $49.95.



After you've read License Agreement (p.95) *choose according item and click Next >.*

If you do not agree with any of the items of the Agreement click **Cancel**.

Ready to Install
Setup is now ready to begin installing Master PDF Editor 3 on your computer.

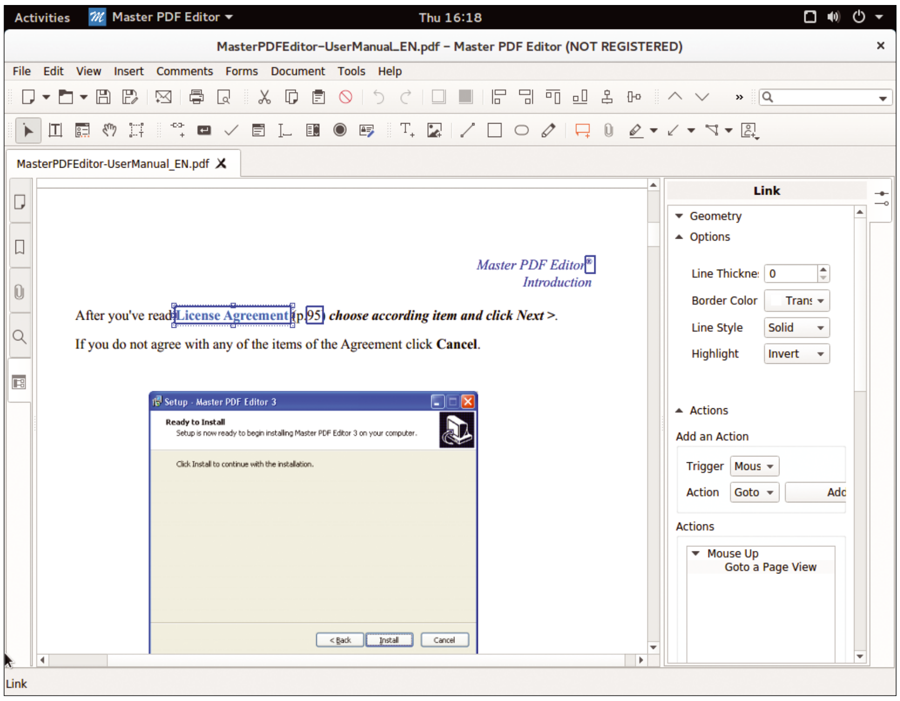
Click Install to continue with the installation.

Line Thickne: 0
Border Color: Trans
Line Style: Solid
Highlight: Invert

Add an Action
Trigger: Mous
Action: Goto
Add

Actions
Mouse Up
Goto a Page View

Figure 1: Neat and uncluttered: the Master PDF Editor graphical interface.


Activities
Master PDF Editor
Thu 16:18
MasterPDFEditor-UserManual_EN.pdf – Master PDF Editor (NOT REGISTERED)
File Edit View Insert Comments Forms Document Tools Help
MasterPDFEditor-UserManual_EN.pdf
Link

Master PDF Editor
Introduction

After you've read License Agreement p. 95 ***choose according item and click Next >.***

If you do not agree with any of the items of the Agreement click **Cancel**.

Setup - Master PDF Editor 3
Ready to Install
Setup is now ready to begin installing Master PDF Editor 3 on your computer.
Click Install to continue with the installation.
< Back | Install | Cancel

Link
Geometry
Options
Line Thickne: 0
Border Color: Trans
Line Style: Solid
Highlight: Invert
Actions
Add an Action
Trigger: Mous
Action: Goto
Actions
Mouse Up
Goto a Page View

Figure 2: **Thanks to the Object Inspector, even PDF non-experts can review and, if necessary, repair complex features in documents.**

and the pitfalls are too numerous to avoid during conversion.

Master PDF Editor

Master PDF Editor [1] is a commercial software tool available for Linux, Mac OS, and Windows. On Linux, most of the functions can be tested and used without purchasing a license (see the "License" box). The software tries to accommodate the process of editing PDF documents in an intuitive way in a graphical interface. The results were already quite impressive in version 3 of the application [2], but Master PDF Editor 4, published shortly before the end of 2016, now offers new features. For example, you can scan documents directly from within the application and convert the text via OCR.

The editor interface has not changed significantly since the last version (Figure 1).

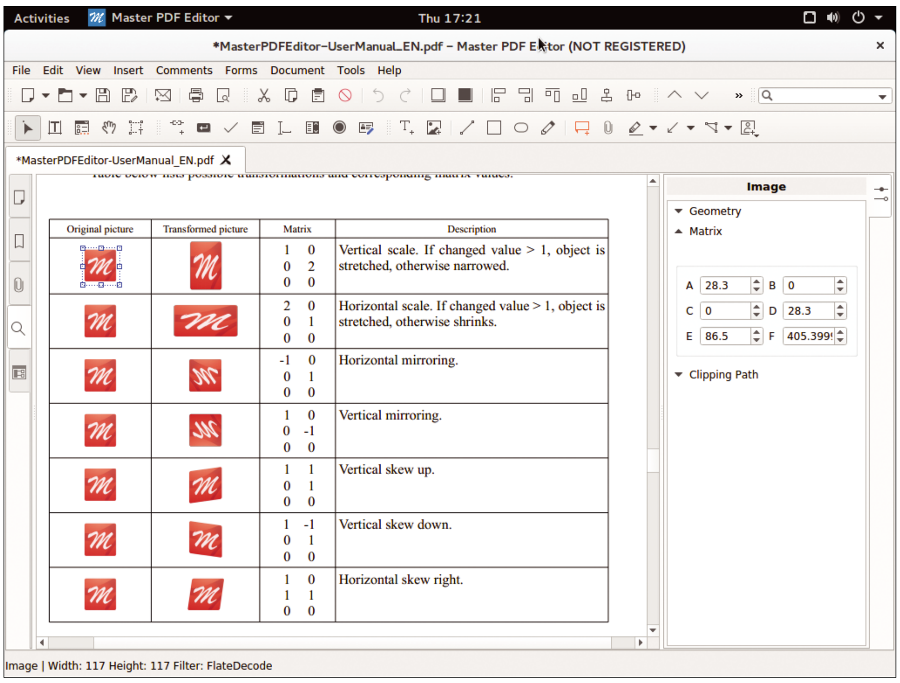

Activities
Master PDF Editor
Thu 17:21
*MasterPDFEditor-UserManual_EN.pdf – Master PDF Editor (NOT REGISTERED)
File Edit View Insert Comments Forms Document Tools Help
*MasterPDFEditor-UserManual_EN.pdf

Table below lists possible transformations and corresponding matrix values.

Original picture	Transformed picture	Matrix	Description
		1 0 0 2 0 0	Vertical scale. If changed value > 1, object is stretched, otherwise narrowed.
		2 0 0 1 0 0	Horizontal scale. If changed value > 1, object is stretched, otherwise shrinks.
		-1 0 0 1 0 0	Horizontal mirroring.
		1 0 0 -1 0 0	Vertical mirroring.
		1 1 0 1 0 0	Vertical skew up.
		1 -1 0 1 0 0	Vertical skew down.
		1 0 1 1 0 0	Horizontal skew right.

Image
Geometry
Matrix
A 28.3 B 0
C 0 D 28.3
E 86.5 F 405.399
Clipping Path

Image | Width: 117 Height: 117 Filter: FlateDecode

Figure 3: **The Matrix lets you manipulate images and text (e.g., rotate, scale, flip, skew).**

In addition to the usual menus, context menus, and toolbars, the margins play an important role. On the left are search functions, page previews, attachments, and a compressed index of the active document. The right margin has the most important universal tool of the program: the Object Inspector (Figure 2).

To use the Object Inspector, you first need to select an object (i.e., text, graphic, or link). The contents of the window displayed in the Object Inspector then change so that you can check and also change all the relevant features. Because of the underlying PDF format's object and box models, all objects include various metadata in addition to the content, such as the location of the object, its attributes, and references to other objects.

Manipulating Objects

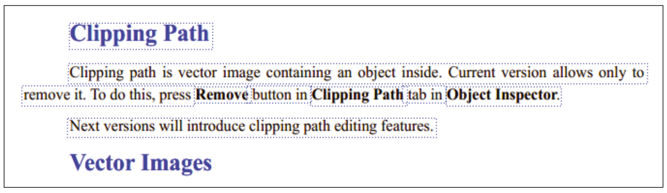
The Object Inspector displays the relevant metadata, which can be edited in the object window for the currently selected box; however, you change the content of the box directly in the main window. Although many boxes (such as text boxes) can be moved in the main window with the mouse, in practice, you should use the corresponding field in the Object Inspector. In this way, you can make sure the box is placed exactly at the desired position. Moreover, the fields in the *Matrix* area offer good options for visually manipulating the way objects display (Figure 3).

Master PDF Editor represents the inner structure of a PDF document with boxes that contain the various objects that form the basis of the document and displays them in the editor accordingly. In practice, this means that although you can correct the text, the program does not let you change the attributes (bold, italic, font) of individual characters.

A font change is displayed in a new box in the document. If the character set changes in a line, it is reflected as a box before the font change, a box with the text in the new font, and a box after the font change (Figure 4).

Special Functions

In addition to the pure editing functions, Master PDF Editor supports many features of modern PDFs – the "Features"


Clipping Path

Clipping path is vector image containing an object inside. Current version allows only to remove it. To do this, press **Remove** button in **Clipping Path** tab in **Object Inspector**.

Next versions will introduce clipping path editing features.

Vector Images

Figure 4: When multiple fonts are used in a PDF document, they are placed in separate boxes.

box summarizes the main options. If necessary, you can add new objects to an existing PDF.

For example, you can create simple vector graphics and incorporate them into the document, create new bookmarks, add notes, and even attach external files. In principle, any files – including executable programs – can be attached to a PDF and extracted again by standard PDF viewers on the local system. Fortunately, attaching documents is rarely done nowadays, because it exposes the recipient to significant security risks.

Master PDF Editor offers five editing modes that access different "levels" of enabling or selecting objects within PDF documents. You'll find all these tools at the left end of the second toolbar:

- The arrowhead-shaped *Edit Document* tool selects any object, such as the bounding line and word boxes for text.
- The *Edit Text Object* tool is specifically designed for editing text; it displays a T-bar cursor.
- The *Edit Form* tool works much like the text tool, but it is specifically designed for forms.
- The *Hand Tool* is the universal tool for manipulating content. In text, it selects the text, not the bounding boxes.
- The *Select Text* tool, symbolized by a rasterized square with a T in one corner, is for larger text areas (e.g., to copy and paste).

Selected objects are located in a stack of objects. The position in this stack determines which object covers which other objects and thus controls visibility. In the center of the top toolbar you can find functions that let you modify the object stack.

In the PDF document, the *Clipping Path* in the right margin frames the visible objects. It cannot currently be edited; Master PDF Editor only supports deletion of the frame. An option for changing the path is already on the developer's roadmap. The *Undo/Redo* functions let you remove and restore individual actions.

The program also supports active content. Its approach is similar to that of web pages, in that it allows actions to be executed when a reader clicks on a predefined area. This can be achieved, for example, using bookmarks. To create an active area, first select it with the *Hand Tool*, right-click and choose *Add Bookmark*, and edit the area (Figure 5). Under *Actions*, adjust what action triggers what behavior.

Actions implemented in JavaScript are also available for the entire document; you define these under *Document | Document Actions*. Master PDF Editor thus offers an effective and, above all, fast approach to accessing the active content of interactive PDFs and defusing it, if needed.

New in Version 4

Soon after Master PDF Editor version 4.0 was released, the developers released two minor updates in January 2017. One significant new feature is the OCR function for processing scanned documents, which is roughly equivalent to the functionality that gscan2pdf [4] or Paperwork [5] attempt to provide. All three programs use the Tesseract OCR engine, which promises reasonably good results.

Using the scan function, Master PDF Editor initially generates a PDF with images (see the "Scanning" box). The OCR function, which you will find in the *Document | OCR* menu, converts the results back into directly editable PDFs. I ran a simple test to check how well the OCR performs. The idea was for Master PDF Editor to scan a short passage from a magazine and convert it into machine-readable text.

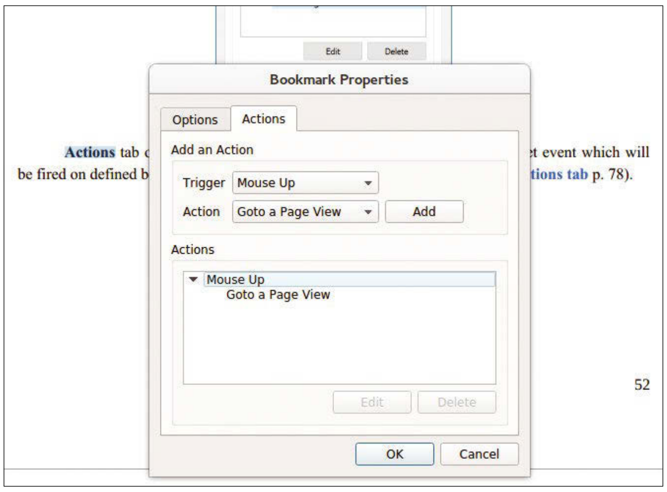
Although the scanner used for this test had a resolution of up to 600dpi, the results were not convincing. The original presented a challenge to the system: The text was not on a white background, the paper of the original was slightly wavy, and the printing was of moderate quality. However, the OCR routine should have identified these problems and correctly scanned

FEATURES

Modern PDF documents include more than just the data needed for printing, which makes it difficult to present PDF in an editable form and explains why hardly any free tools support the editing of PDF documents reliably and in detail.

Master PDF Editor supports simple actions like filling out PDF forms and deleting or inserting individual pages. Additionally, it can edit text in the document, replace images, modify and customize text attributes (i.e., formatting), encrypt and decrypt documents, and adapt digital signatures. Finally, you can add, edit, and remove bookmarks and other active content.

The less commonly used functions include JavaScript support, the ability to convert PDF to bitmap formats, and dynamic XFA forms. Optimized saving is one of the few features that is only available in the commercial version. The program scales bitmap graphics to a previously defined resolution and removes unwanted objects from the document. A document that previously weighed in at 120MB thus often shrinks to 20 or 30MB without compromising the quality.


Bookmark Properties

Options | Actions

Add an Action

Trigger: Mouse Up

Action: Goto a Page View

Actions

Mouse Up
  Goto a Page View

Figure 5: Actions can be defined with the use of bookmarks.

SCANNING

The scan function in Master PDF Editor resides in the *File | New | From Scanner* menu. The preview is at the same resolution as the final scan, which makes little sense and is unnecessarily time consuming. If you want to digitize the document using OCR after scanning, do not choose a low scanning resolution: Usually, the higher the resolution, the better the character recognition results.

OCR text. The results indicated much room for improvement.

Many OCR tools use a more-or-less intelligent spell checker that warns the user if many errors occur; this feature is missing in Master PDF Editor. The installation stores the Tesseract data locally on the hard drive, regardless of whether the files already exist in a system global installation under `/usr/share/tessdata/`, resulting in data redundancy.

If you want to understand all the features in Master PDF Editor, you can refer to the manual, which comes as a PDF or online [6]. However, it still refers to version 3.7 and thus contains no information on scanning or OCR. That said, the manual explains all the other features well and in detail. In fact, the manual contains a great deal of information about the structure of PDF files that is otherwise difficult to find.

Conclusions

Despite minor weaknesses, Master PDF Editor 4 proves to be a fine piece of software for retroactive PDF editing. In practical terms, no other free software with a similar feature set exists (see the "Alternatives" box). The latest version of the program comes with promising new functions, such as scanning and text recognition, but does not yet deliver in practice what it promises. ■■■

ALTERNATIVES

You can find a number of free tools that also attempt to edit PDF documents. The best results are currently obtained with LibreOffice Draw, but in practical terms, LibreOffice's capabilities lag miles behind those of Master PDF Editor.

INFO

[1] Master PDF Editor 4: *https://code-industry.net/masterpdfeditor/*

[2] "Master PDF Editor" by Karsten Günther, *Linux Pro Magazine*, issue 164, July 2014, pg. 46, *http://www.linuxpromagazine.com/Issues/2014/164/Master-PDF-Editor*

[3] Free version: *https://code-industry.net/free-pdf-editor*

[4] gscan2pdf: *http://gscan2pdf.sourceforge.net*

[5] "Paperwork" by Karsten Günther, *Ubuntu User*, issue 33, 2017, pg. 13, *http://www.ubuntu-user.com/Magazine/Archive/2017/33/Use-Paperwork-to-digitize-and-archive-documents*

[6] Manual: *https://code-industry.net/masterpdfeditor-help*

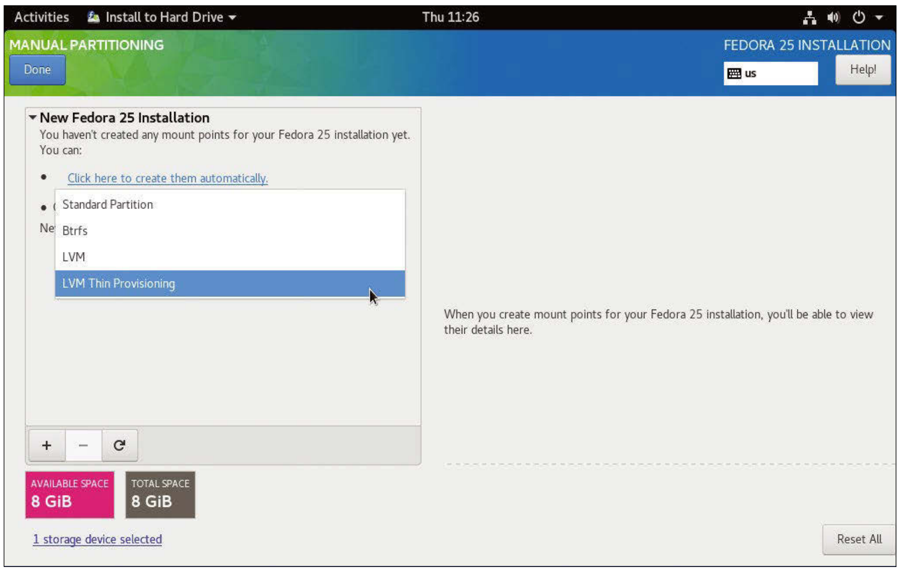
Installer framework
Calamares at a glance

Modular Kit

Calamares helps you create simplified routines for installing a distribution, but there are some pitfalls, as we explain. *By Ferdinand Thommes*

Installing Linux today is quite straightforward compared to, say, 15 years ago when the process could be quite time-consuming. That said, installers still confront newcomers with issues such as partitioning, which is similar to open heart surgery and can easily go wrong. Installing Arch Linux is the exception in this case, because there is no official installer for the system. But, if you look at Debian, Fedora (Figure 1), or openSUSE, their setups offer an abundance of possibilities, even including LVM/RAID on encrypted systems.

This diversity can easily be confusing for beginners. Additionally, these users may be accustomed from Windows or Mac OS X to always having the same installer, which you get to know better over time. Users who often switch distros on



Activities Install to Hard Drive ▾ Thu 11:26

MANUAL PARTITIONING FEDORA 25 INSTALLATION

Done us Help!

▾ New Fedora 25 Installation
You haven't created any mount points for your Fedora 25 installation yet. You can:

- Click here to create them automatically.
- Ne

Standard Partition
Btrfs
LVM
LVM Thin Provisioning

When you create mount points for your Fedora 25 installation, you'll be able to view their details here.

+ − ↻

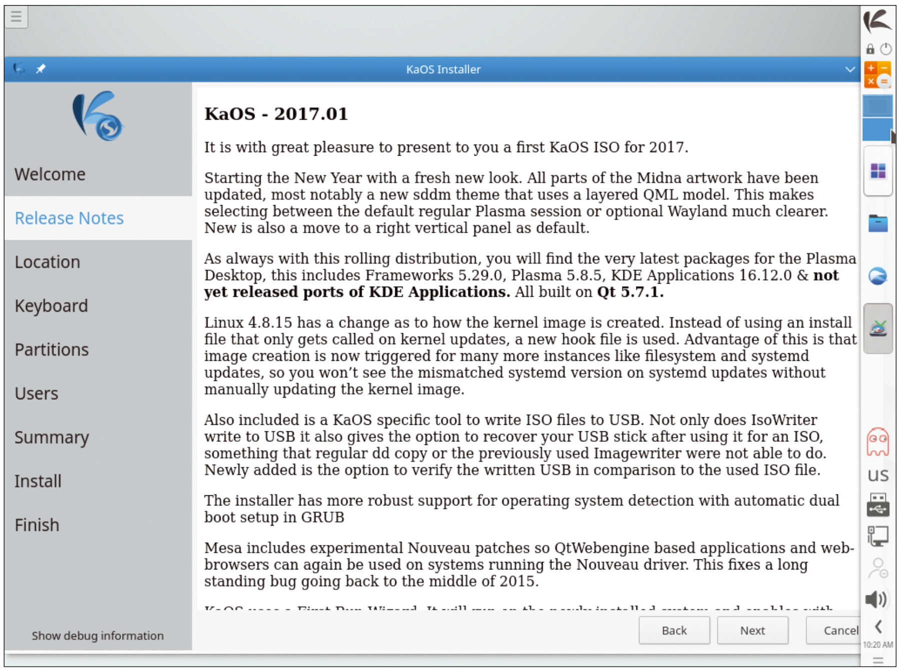
AVAILABLE SPACE 8 GiB
TOTAL SPACE 8 GiB

1 storage device selected

Reset All

Figure 1: **The Fedora installer *Anaconda* offers so many options that identifying the switches and key settings can overwhelm beginners.**

Lead Image © ChriSes, photocase



KaOS - 2017.01

It is with great pleasure to present to you a first KaOS ISO for 2017.

Starting the New Year with a fresh new look. All parts of the Midna artwork have been updated, most notably a new sddm theme that uses a layered QML model. This makes selecting between the default regular Plasma session or optional Wayland much clearer. New is also a move to a right vertical panel as default.

As always with this rolling distribution, you will find the very latest packages for the Plasma Desktop, this includes Frameworks 5.29.0, Plasma 5.8.5, KDE Applications 16.12.0 & **not yet released ports of KDE Applications.** All built on **Qt 5.7.1.**

Linux 4.8.15 has a change as to how the kernel image is created. Instead of using an install file that only gets called on kernel updates, a new hook file is used. Advantage of this is that image creation is now triggered for many more instances like filesystem and systemd updates, so you won't see the mismatched systemd version on systemd updates without manually updating the kernel image.

Also included is a KaOS specific tool to write ISO files to USB. Not only does IsoWriter write to USB it also gives the option to recover your USB stick after using it for an ISO, something that regular dd copy or the previously used Imagewriter were not able to do. Newly added is the option to verify the written USB in comparison to the used ISO file.

The installer has more robust support for operating system detection with automatic dual boot setup in GRUB

Mesa includes experimental Nouveau patches so QtWebengine based applications and web-browsers can again be used on systems running the Nouveau driver. This fixes a long standing bug going back to the middle of 2015.

Figure 2: The KaOS distro uses the interface to Calamares to display its proprietary module with release notes in the installer.

Linux, however, are repeatedly confronted with different interfaces.

The situation improved with the emergence of Ubuntu. In version 6.06, its developers presented Ubiquity [1] as a wizard for the installation of a live system. The features of the Debian installer are used in the background. The roots of the software were in the Spanish distribution Guadalinex, which received government funding.

Almost all Ubuntu offshoots subsequently moved to Ubiquity and thus unified the image of distributions based on Debian. But, in the realm of the RPM distributions and Arch Linux derivatives, wild variety prevailed and still does today.

A Brilliant Idea

About three years ago, KDE developer Teo Mrnjavac (see "Interview" box) then implemented an idea that takes much of the work off the distributor's hands. The distribution-independent Calamares installer framework [2] composes modules to create an installer with a custom look or lets you create these modules yourself. Rudimentary knowledge of Python helps you understand the code of the module – but this is not a precondition.

Mrnjavac, who previously helped develop the Amarok and Tomahawk audio applications, worked for Blue Systems at the time [3]. This company employs about a dozen well-known KDE developers and is currently supporting KDE Neon, Netrunner, and Maui distributions, as well as other Linux projects, both financially and in terms of development.

After nine months of development work, Calamares 1.0 was ready in February 2015. The result included 25 modules, covering everything from basic bootloader support, partitioning, user management, and the installation itself. There is also an interface for modules created by distributors themselves. KaOS and Siduction are two distributions that make use of this by displaying release notes directly in the installer (Figure 2).

Continuous Development

About a year later Calamares 2.0 appeared. The use of *KDE Partition Manager* (Figure 3) was new; this saw the tool revived after years of slumber. This simplified partitioning on the one hand and offered advanced options for the operation on the other. Calamares also supported operations immediately after installation, which made it possible to run commands, such as removing parts of the live session that were no longer needed.

Since Calamares 2.2, the software has also supported state-of-art NVM Express (NVME) drives. Version 2.3 added complete encryption of the system to be

INTERVIEW

Linux Magazine: *Teo, about three years ago, you had the idea of a distribution neutral installer framework that then appeared some time later as Calamares. How did you get this idea?*

Teo Mrnjavac: Before Calamares, it was customary for new distributions to take the installer from some other distribution and to adapt it to your own needs. In the end, this resulted in forks of forks, each with their own errors and shortcomings. I realized that much time was being wasted here, and that it would make more sense for distributors to collaborate on a product that everyone shared.

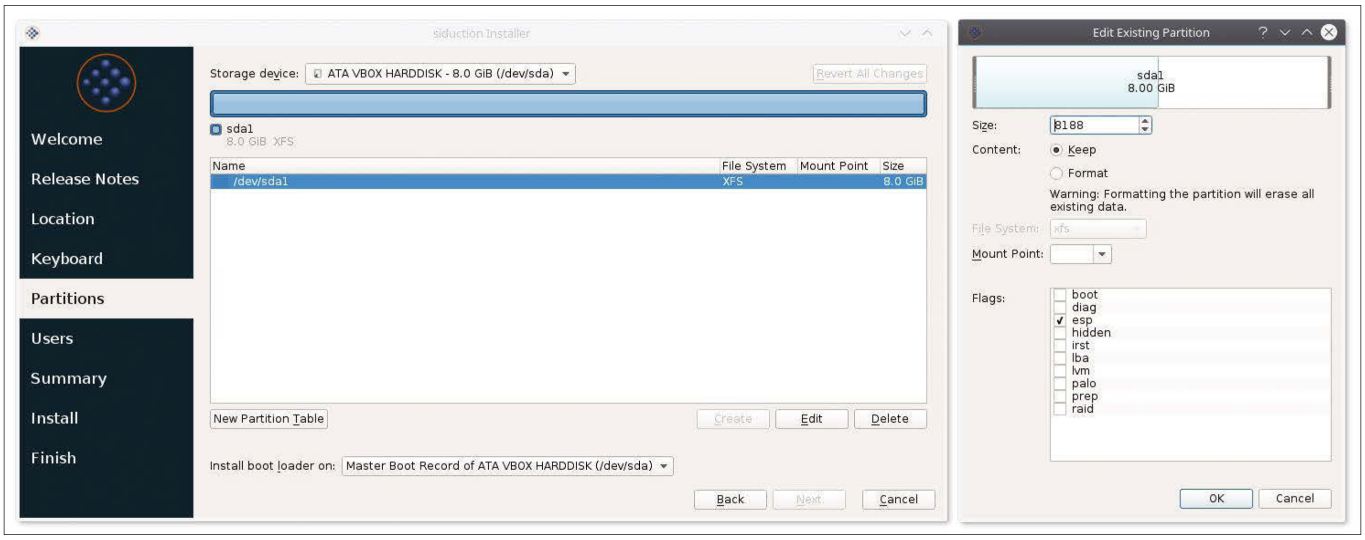
LM: *Some 20 distributions already rely on Calamares; many have collaborated in the development of the framework. And, it looks as if more distributions will be following suit. What are your future plans for the project?*

TM: Calamares is a modular tool that allows distributions to compile the installer itself, rather than a finished product. Calamares' fate thus depends on objectives and priorities, and last but not least, on the needs of the respective user. My main goal in this project is to enable distributors to implement their own ideas in terms of the installer and to ensure a trouble-free installation.

LM: *What kind of skills does a distributor need to create a deployable installer with Calamares?*

TM: Calamares is designed to support ease-of-use. In most cases, the distributor only needs to modify a few configuration files. In addition, there is the option of extending the framework with your own modules in Python or C++ – this is something you can do, but by no means have to.

LM: *Teo, thank you very much for the interview!*



Storage device: ATA VBOX HARDDISK - 8.0 GiB (/dev/sda)

sda1
8.0 GiB XFS

Name	File System	Mount Point	Size
/dev/sda1	XFS		8.0 GiB

Install boot loader on: Master Boot Record of ATA VBOX HARDDISK (/dev/sda)

sda1
8.00 GiB

Size: 8188

Content: [✓ Keep] Format

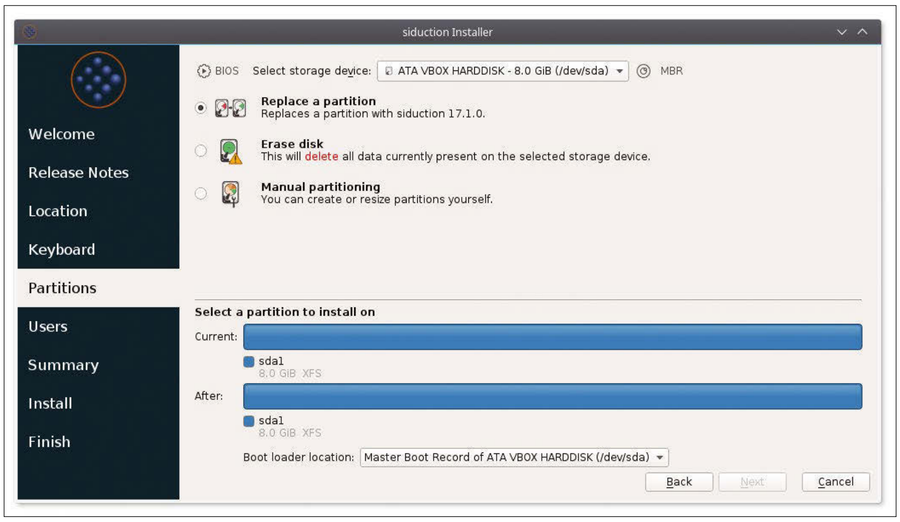
Warning: Formatting the partition will erase all existing data.

File System: xfs

Mount Point:

Flags: boot, diag, [✓ esp], hidden, irst, lba, lvm, palo, prep, raid

Figure 3: The *KDE Partition Manager* – restored to its former glory and now featured in Calamares.



BIOS Select storage device: ATA VBOX HARDDISK - 8.0 GiB (/dev/sda) MBR

● **Replace a partition**
Replaces a partition with siduction 17.1.0.

○ **Erase disk**
This will delete all data currently present on the selected storage device.

○ **Manual partitioning**
You can create or resize partitions yourself.

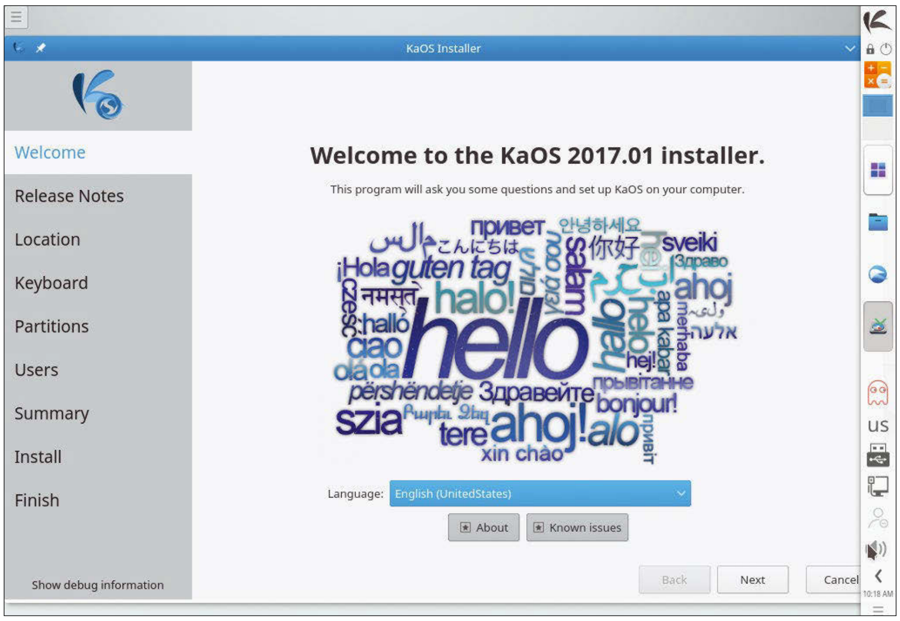
Select a partition to install on

Current:
sda1
8.0 GiB XFS

After:
sda1
8.0 GiB XFS

Boot loader location: Master Boot Record of ATA VBOX HARDDISK (/dev/sda)

Figure 4: The *Replace Partition* option lets you install without changing the type or size of the filesystem.



Welcome to the KaOS 2017.01 installer.

This program will ask you some questions and set up KaOS on your computer.

Language: English (UnitedStates)

Figure 5: Calamares offers the possibility to customize the Welcome screen. Only the language selection is standard.

installed to its repertoire, including the bootloader and the optional swap partition based on the Linux Unified Key Setup (LUKS) specification. This even applies to dual-boot environments, for example, with Windows 10.

The developers also introduced the *Replace Partition* function, which lets you reuse a partition without changes to its size or type at the push of a button (Figure 4).

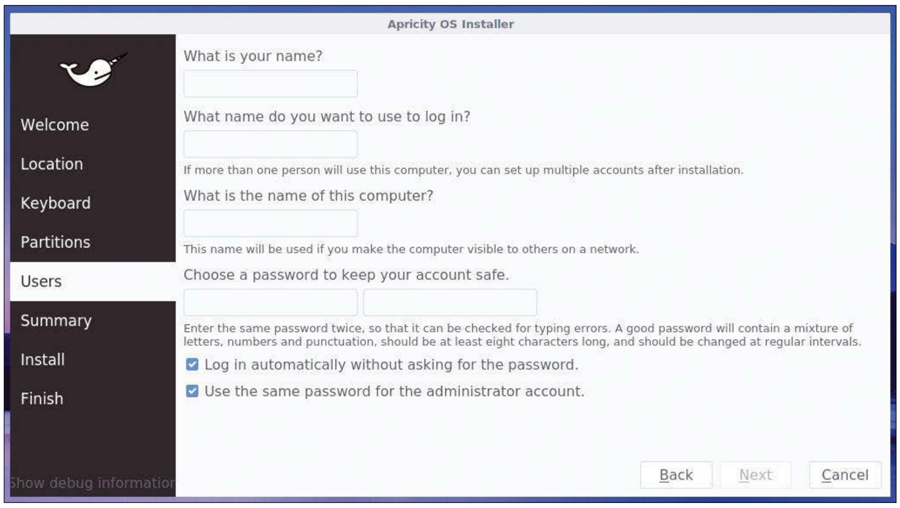
More and More

The latest version is Calamares 3.1 (from April 2017), which is already use by about 20 distributions. It is written mostly in C++; the developers used the Qt framework for the interface. Scripting in the modules is handled by Python 3, and the software configuration is done by the YAML [4] markup language. Users never get to see some of the modules; those that handle the interaction appear on the left edge of the window as tabs.

In the first tab – *Welcome* – you select the desired language (Figure 5). Calamares adjusts the locale, the time zone, and language accordingly in the second tab. The keyboard model and the layout are then set in the third tab, and you can test the parameters there, if necessary.

Processing Tabs

The nitty-gritty starts in the next tab: partitioning. In the top line, the software shows you whether the medium was booted via the BIOS or UEFI, and whether MBR or GPT was used. You also select the storage device.


Apricity OS Installer

What is your name?

What name do you want to use to log in?

If more than one person will use this computer, you can set up multiple accounts after installation.

What is the name of this computer?

This name will be used if you make the computer visible to others on a network.

Choose a password to keep your account safe.

Enter the same password twice, so that it can be checked for typing errors. A good password will contain a mixture of letters, numbers and punctuation, should be at least eight characters long, and should be changed at regular intervals.

- [x] Log in automatically without asking for the password.
- [x] Use the same password for the administrator account.

Figure 6: In the case of Apricity, a Calamares-based installer, the default setting creates the same password for root and the user.

In the simplest case, you use the *Replace Partition* option to take advantage of storage space for the installation without changing the size or the filesystem. You also have the option to erase the disk and install in the space this frees up.

The third path takes you to manual partitioning. This is where you can modify partitions in terms of size, filesystem, mount point and label, or write a new partition table. You can also determine where the bootloader – optionally systemd or GRUB – ends up.

Let Me In

The *User* tab proves less critical, but still comes with some minor pitfalls. Next to the input fields for names and passwords, there are two checkboxes on the page. One is for automatic login without a password. This is useful for machines that you use on your own, and where you always use the same session.

The second option is the Calamares default, and also the default for some distros, which envisages a password for the main user and root. Here you need to make a conscious decision about what you want to do. If you uncheck the box, this opens up another prompt for a separate administrator password (Figure 6). If you want to change this default value for a distribution to be composed, you will find the parameter in `/etc/calamares/modules/users.conf`. You can configure the sudo mechanism there, if necessary.

At a Glance

Below *Summary*, you again see all the settings you configured and their impact (Figure 7). Before beginning, the *Install* tab prompts you one last time to make sure that you want to start the process. After confirmation, a bar visualizes the progress of the installation.

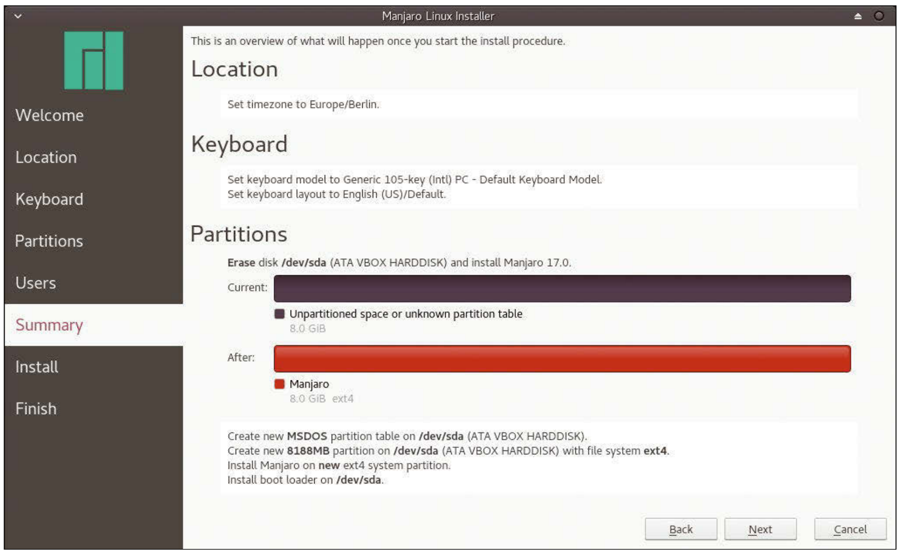
During the installation, you can display a kind of slideshow with details of the distribution and configuration. The procedure is usually completed in a couple of minutes. The last tab provides information about completing the installation and prompts you to reboot.

If you are interested in building Calamares itself, or understanding the configuration, the wiki on GitHub [5] will help. You can find the appropriate files below `/etc/calamares/` on live media equipped with Calamares.

Conclusions

Popular distributions such as Debian, openSuse, or Fedora are unlikely to drop the installers they have maintained for many years in favor of Calamares. This is especially attributable to distribution-specific features that Calamares does not offer. Fedora at least delivers Calamares as a package, which is likely to help developers of derivatives. But, it is conceivable for distributors to offer Calamares as an additional alternative for beginners along with their own installers. After all, Calamares has convinced nearly two dozen distributions who now rely on the modular model. They include Apricity, Chakra, KaOS, Manjaro, Siduction, Tanglu, and – in recent weeks – KDE Neon.

Calamares offers prebuilt modules for the appropriate steps, but it also gives you enough freedom to install your own modules and customize the look. Thus, beginners will probably see a growing number of distributions with a single installer in the future. This way, harmonization on Linux makes sense and is fun. ■■■


Manjaro Linux Installer

This is an overview of what will happen once you start the install procedure.

Location

Set timezone to Europe/Berlin.

Keyboard

Set keyboard model to Generic 105-key (Intl) PC - Default Keyboard Model.
Set keyboard layout to English (US)/Default.

Partitions

Erase disk **/dev/sda** (ATA VBOX HARDDISK) and install Manjaro 17.0.

Current: Unpartitioned space or unknown partition table 8.0 GiB

After: Manjaro 8.0 GiB ext4

Create new **MSDOS** partition table on **/dev/sda** (ATA VBOX HARDDISK).
Create new **8188MB** partition on **/dev/sda** (ATA VBOX HARDDISK) with file system **ext4**.
Install Manjaro on **new** ext4 system partition.
Install boot loader on **/dev/sda**.

Figure 7: The summary of the chosen parameters (in this case, Manjaro) shows what the installer will do with the system as soon as you initiate the setup.

INFO

[1] Ubiquity: *https://wiki.ubuntu.com/Ubiquity*

[2] Calamares: *https://calamares.io/about/*

[3] Blue Systems: *http://www.blue-systems.com/projects/*

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

[5] Wiki: *https://github.com/calamares/calamares/wiki*

Evaluate systemd logs using journalctl

Finely Filtered

The journal is the systemd component responsible for viewing and managing logfiles.

By Ferdinand Thommes

The systemd service journald creates very comprehensive logs, which allow you to analyze data in a variety of ways. Once you get used to the convenience this affords, you won't even miss the old-style logfiles.

Better structured

The *journald* daemon collects messages from the kernel, initrd, any running services and other available sources and collects them into one place. This results in a massive amount of data compared to the logfiles you're traditionally used to such as */var/log/messages* or */var/log/syslog*. You'll also find a huge amount of metadata is included, which can significantly improve your results when searching the journal.

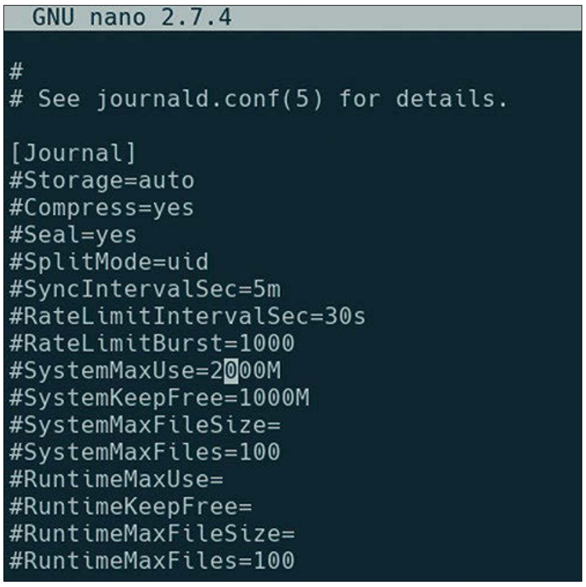
This also results in a larger amount of data which is tricky to store in traditional text files. This is why the *journald* daemon stores this information in binary files. You can read this data, using the command `journalctl`. If necessary, you can also convert the binary files to other formats for further analysis.

A little off the top

The huge amount of data being logged results in large files. While your system previously managed to trim these down using the *Logrotate* tool to compress and archive logfiles, the systemd journal allows you to set limits when files are changed.

Take a look at your own systemd [1] configuration file. For non-Debian based Linux distros, this is usually found in `/etc/systemd/journald.conf`. You may also find it in `/etc/systemd/journald.conf`. Look for the values `SystemMaxUse` and `SystemKeepFree`. The first of these determines how much space the journal files contain. The second value specifies how much free space to allocate (Figure 1).

Both values will determine the absolute limit on journal file sizes and available space. If you don't enter any values, the default amount of space permitted for journal files is 10 percent of your filesystem. The minimum amount of free space is 15 percent by default. The program caps both values at 4 GB for each.



```
  GNU nano 2.7.4

#
# See journald.conf(5) for details.

[Journal]
#Storage=auto
#Compress=yes
#Seal=yes
#SplitMode=uid
#SyncIntervalSec=5m
#RateLimitIntervalSec=30s
#RateLimitBurst=1000
#SystemMaxUse=2000M
#SystemKeepFree=1000M
#SystemMaxFileSize=
#SystemMaxFiles=100
#RuntimeMaxUse=
#RuntimeKeepFree=
#RuntimeMaxFileSize=
#RuntimeMaxFiles=100
```

Figure 1: The systemd journal can expand very quickly. To save on disk space, limit the size of the journal files.

LISTING 1: Viewing the Current Status of the Journal Daemon

```
$ systemctl status systemd-journald
 systemd-journald.service - Journal Service
   Loaded: loaded (/lib/systemd/system/systemd-journald.service; static; vendor
           preset: enabled)
   Active: active (running) since Fr 2017-03-10 14:33:37 CET; 4h 21min ago
     Docs: man:systemd-journald.service(8)
           man:journald.conf(5)
 Main PID: 10376 (systemd-journal)
   Status: "Processing requests..."
   CGroup: /system.slice/systemd-journald.service
            10376 /lib/systemd/systemd-journald
Mar 10 14:33:37 rpg-pi3b-01 systemd-journald[10376]: Runtime journal
(/run/log/journal/) is 4.7M, max 38.3M, 33.5M free.
Mar 10 14:33:37 rpg-pi3b-01 systemd-journald[10376]: Journal started
Warning: Journal has been rotated since unit was started. Log output is incomplete
or unavailable.
```

Given the size of hard drives nowadays, it's perfectly feasible to set `SystemMaxUse` to somewhere between 50 MB to 1 GB. In any case, the default values for the journal are sufficient for everyday use. Feel free to change the `SystemMaxFileSize` to limit the size of individual files if you think it's necessary, however.

When setting size limits, use the values `K`,`G` and `T` for kilobytes, gigabytes and terabytes respectively. If you want to alter the default values in `journald.conf`, make sure to uncomment each line by removing the `#` at the start, then restart the *journald* daemon.

Persistent or volatile?

Take a look at the line reading *Storage* = . While most Linux distributions are configured to store logs continuously, your logs will only be saved to `var/log/journal` if you set this to `Storage=persistent`. If you don't want to save the logs, change this to `Storage=volatile` instead. Doing this means your logs will be stored in `/run/log/journal` but will be deleted by the daemon once your active session ends.

For more options for controlling the journal size, pull up the man page for `journald.conf`. If you want to reduce the journal size while it's running, you have two choices. Your first choice is to use the following command to reduce journal size, for instance to 100 MB. While running as root, enter:

```
# journalctl --vacuum-size=100M
```

The program will delete the oldest entries until the journal reaches the desired size (in this case 100 MB). Your second option is to remove all entries before a certain time:

```
# journalctl --vacuum-time=1month
```

The above command discards all messages older than one month (Figure 2), irrespective of how large or small this would make the logfile.

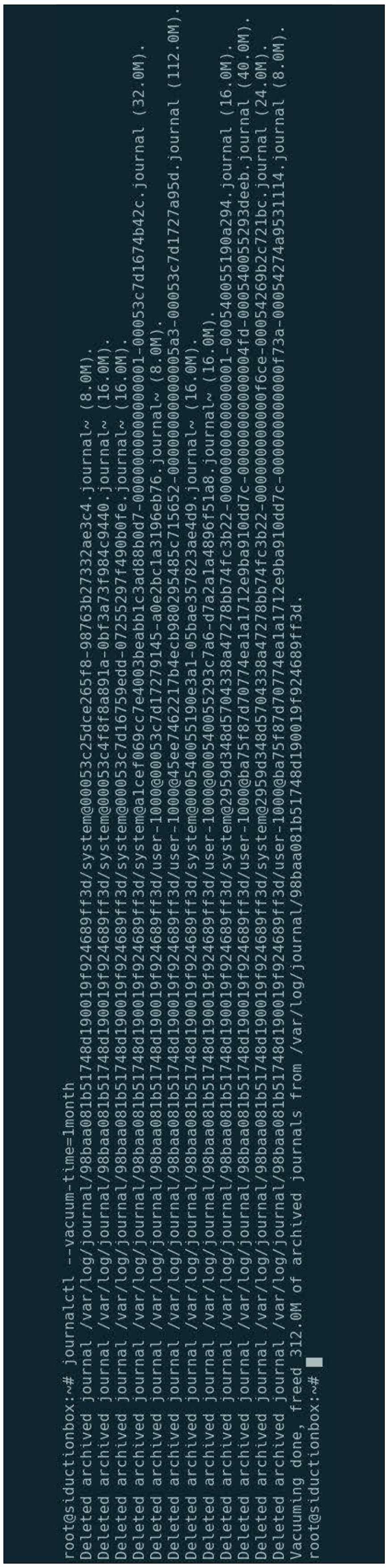
Early and Often

The *journald* daemon not only records much more data than other logging mechanisms, but actually starts up much earlier in the boot process than was previously possible. This is a huge help when narrowing down system startup problems. Readers who remember the number of photos taken of systems not booting because of a kernel panic or other boot issues on support forums will no doubt relate. Thanks to systemd this is a thing of the past.

Systems using *SysVinit* [2] do not store messages from the initial stages of the boot process, as the root file system has not yet been mounted as a writable medium. However, systems using systemd create a socket [3] at run time, from which collected messages can be read. The journal, therefore, offers some significant advantages, despite creating numerous binary files.

Status and verification

Your operating system contains a journal for each user as well as one for the sys-



```
root@siductionbox:~# journalctl --vacuum-time=1month
Deleted archived journal /var/log/journal/98baa081b51748d190019f924689ff3d/system@00053c25dce265f8-98763b27332ae3c4.journal~ (8.0M).
Deleted archived journal /var/log/journal/98baa081b51748d190019f924689ff3d/system@00053c4f8f8a891a-0bf3a73f984c9440.journal~ (16.0M).
Deleted archived journal /var/log/journal/98baa081b51748d190019f924689ff3d/system@00053c7d16759edd-07255297f490b0fe.journal~ (16.0M).
Deleted archived journal /var/log/journal/98baa081b51748d190019f924689ff3d/system@a1cef069cc7e4003beabb1c3ad88b0d7-0000000000000001-00053c7d1674b42c.journal (32.0M).
Deleted archived journal /var/log/journal/98baa081b51748d190019f924689ff3d/user-1000@00053c7d17279145-a0e2bc1a3196eb76.journal~ (8.0M).
Deleted archived journal /var/log/journal/98baa081b51748d190019f924689ff3d/user-1000@45ee7462217b4ecb980295485c715652-00000000000005a3-00053c7d1727a95d.journal (112.0M).
Deleted archived journal /var/log/journal/98baa081b51748d190019f924689ff3d/system@000540055190e3a1-05bae357823ae4d9.journal~ (16.0M).
Deleted archived journal /var/log/journal/98baa081b51748d190019f924689ff3d/user-1000@000540055293c7a6-d7a2a1a4896f51a8.journal~ (16.0M).
Deleted archived journal /var/log/journal/98baa081b51748d190019f924689ff3d/system@2959d348d5704338a47278bb74fc3b22-0000000000000001-000540055190a294.journal (16.0M).
Deleted archived journal /var/log/journal/98baa081b51748d190019f924689ff3d/user-1000@ba75f87d70774ea1a1712e9ba910dd7c-00000000000004fd-000540055293deeb.journal (40.0M).
Deleted archived journal /var/log/journal/98baa081b51748d190019f924689ff3d/system@2959d348d5704338a47278bb74fc3b22-000000000000f6ce-00054269b2c721bc.journal (24.0M).
Deleted archived journal /var/log/journal/98baa081b51748d190019f924689ff3d/user-1000@ba75f87d70774ea1a1712e9ba910dd7c-000000000000f73a-00054274a9531114.journal (8.0M).
Vacuuming done, freed 312.0M of archived journals from /var/log/journal/98baa081b51748d190019f924689ff3d.
root@siductionbox:~# █
```

Figure 2: Using command-line options such as `--vacuum-time` allows you to reduce the size of your logfiles without changing your configuration.

LISTING 2: Checking the Size and Integrity of Journal Data

```
$ sudo journalctl --disk-usage
Archived and active journals take up 4.7M on disk.
$ sudo journalctl --verify
PASS: /run/log/journal/747bced4498d729c8a19f23400000006/system.journal
```

LISTING 3: Checking Local Time

```
$ timedatectl status
Local time: Tue 2017-05-16 18:42:42 IST
  Universal time: Tue 2017-05-16 17:42:42 UTC
        RTC time: Tue 2017-05-16 17:42:42
       Time zone: Europe/Dublin (IST, +0100)
 Network time on: yes
NTP synchronized: yes
 RTC in local TZ: no
```

tem itself. If a user belongs to the group *systemd-journal*, they can access the journal and view all the data without running as root. Before you dive in and view all the data available, you may want to master a few basic but important commands.

View the current status of the journal daemon using Systemctl (Listing 1). Use `journalctl --disk-usage` to check the current journal size and `journalctl -verify` to test the integrity of your data (Listing 2).

To see whether your log is recording the correct time, run `timedatectl status`. This command lets you check that your time zone corresponds to your location (Listing 3). The top line should show the current time. If you're running the computer in a new time zone use `timedatectl set-timezone <zone>` to adjust.

In theory, you can display any data from the journal using the `journalctl` command. By default the terminal pager program *less* is used to display data. It allows you to scroll back and forth through the log. Most importantly, you can use it from your regular user account without root privileges. When you've finished examining the binary files, return to the command prompt by pressing *Q*.

The whole story

You can display the complete journal output by running the command `journalctl` without any additional options. This will show all saved logs subject to any file size limitations and the time since your last reboot.

Each time you restart the computer, the program will insert the line – *Reboot* – to break up the information. This not only makes the logs easier on the eye, but is useful to determine how long an error has been occurring. Use `journalctl -p err` to limit the output if necessary. This option will display only *ERROR* log levels from the journal.

Normally you'll most probably want to focus on issues occurring at a certain time or filter results. For instance, you can use `journalctl -b` to show all logs since the last boot. If you're interested in logfiles from the last boot but one, run `journalctl -b -1`. Use `journalctl --list-boots` to display all boot events saved in the journal (Listing 4). Use the value from the first column of the output to display information on a specific boot e.g. `journalctl -b -0`

Finding the Time

If necessary, you can filter logs to show events at exact times. This is very helpful for computers that are not restarted often. Use the options `--since` and `--until` to narrow down the time window to the minute or second as required. You can also combine these options.

You can see an example of using one of the options on its own in the first line of Listing 5. The second line contains an example of using both. If you are specifying the time, it must be in the format `YYYY-MM-DD HH:MM:SS`. You can also use more general options such as `--since yesterday`, as shown in Listing 5.

Filter by component

There are other filters you can use to search for specific events. This means you can detect messages relating to individual components. For example, if you are experiencing problems with your web server, use `journalctl -u apache2.service` to display only related events (Figure 3).

If you have an idea of where the error may be occurring within the web server, refine the filter further, as in the example query (Listing 6, line 1) of a suspect PHP module. To investigate specific processes, you can also filter by process, user or group ID.

First, look for the corresponding process ID. You can do this using the com-

LISTING 4: Displaying All Boot Events Saved in the Journal

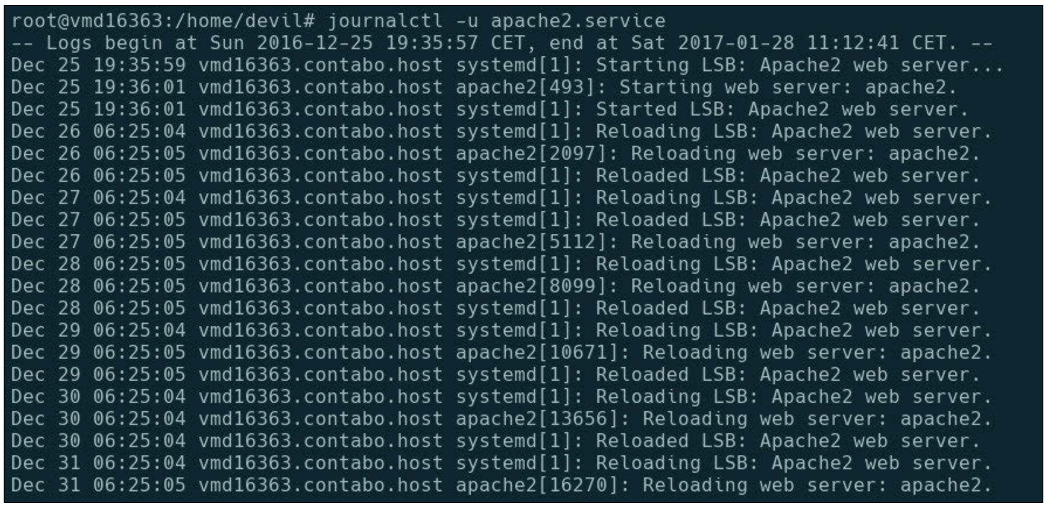
```
$ sudo journalctl --list-boots
 0 9e814cbee30a47ea85a58a5674829a95 Mi 2017-02-08 14:09:34 CET-Fr 2017-03-10
   19:03:25 CET
```

LISTING 5: Filtering Logs by Time

```
$ sudo journalctl --since "2017-05-10 12:00:23"
$ sudo journalctl --since "2017-05-10 12:00:23" --until "2017-05-10 19:00:23"
$ sudo journalctl --since yesterday
$ sudo journalctl --since 12:00 --until "now"
```

LISTING 6: Filtering by Component

```
01 $ sudo journalctl -u apache2.service -u php-fpm.service --since yesterday
02 $ pidof apache2
03 $ ps aux | grep apache2
04 $ sudo journalctl _PID=PID
05 $ id -u www-data
06 $ sudo journalctl _UID=UID --since today
```



```
root@vmd16363:/home/devil# journalctl -u apache2.service
-- Logs begin at Sun 2016-12-25 19:35:57 CET, end at Sat 2017-01-28 11:12:41 CET. --
Dec 25 19:35:59 vmd16363.contabo.host systemd[1]: Starting LSB: Apache2 web server...
Dec 25 19:36:01 vmd16363.contabo.host apache2[493]: Starting web server: apache2.
Dec 25 19:36:01 vmd16363.contabo.host systemd[1]: Started LSB: Apache2 web server.
Dec 26 06:25:04 vmd16363.contabo.host systemd[1]: Reloading LSB: Apache2 web server.
Dec 26 06:25:05 vmd16363.contabo.host apache2[2097]: Reloading web server: apache2.
Dec 26 06:25:05 vmd16363.contabo.host systemd[1]: Reloaded LSB: Apache2 web server.
Dec 27 06:25:04 vmd16363.contabo.host systemd[1]: Reloading LSB: Apache2 web server.
Dec 27 06:25:05 vmd16363.contabo.host systemd[1]: Reloaded LSB: Apache2 web server.
Dec 27 06:25:05 vmd16363.contabo.host apache2[5112]: Reloading web server: apache2.
Dec 28 06:25:05 vmd16363.contabo.host systemd[1]: Reloading LSB: Apache2 web server.
Dec 28 06:25:05 vmd16363.contabo.host apache2[8099]: Reloading web server: apache2.
Dec 28 06:25:05 vmd16363.contabo.host systemd[1]: Reloaded LSB: Apache2 web server.
Dec 29 06:25:04 vmd16363.contabo.host systemd[1]: Reloading LSB: Apache2 web server.
Dec 29 06:25:05 vmd16363.contabo.host apache2[10671]: Reloading web server: apache2.
Dec 29 06:25:05 vmd16363.contabo.host systemd[1]: Reloaded LSB: Apache2 web server.
Dec 30 06:25:04 vmd16363.contabo.host systemd[1]: Reloading LSB: Apache2 web server.
Dec 30 06:25:04 vmd16363.contabo.host apache2[13656]: Reloading web server: apache2.
Dec 30 06:25:04 vmd16363.contabo.host systemd[1]: Reloaded LSB: Apache2 web server.
Dec 31 06:25:04 vmd16363.contabo.host systemd[1]: Reloading LSB: Apache2 web server.
Dec 31 06:25:05 vmd16363.contabo.host apache2[16270]: Reloading web server: apache2.
```

Figure 3: Journalctl can query the logs for individual services.

mands `pidof` or `ps` (Listing 6, lines 2 and 3). Next, query the corresponding process using the option `_PID=` (line 4). In Debian, you can find the user for the web server using the command `id -u www-data` (line 5), for instance to display all events since midnight (line 6). To display all possible filters, use `man systemd.journal-fields`.

You can also search for notifications from misbehaving applications. To do this, simply enter the relevant path. For applications installed via your package manager, these should be located in */usr/bin*, for instance `journalctl /usr/bin/amarok`. For other apps, use `which <name>` to determine the exact path.

Including Kernel Messages

If your want to check for kernel messages, you can do so for the current session with `journalctl -k`. For information on previous sessions, use `journalctl -k -b -n`.

The option *-p* determines the priority of displayed messages. For instance, `journalctl -p err` shows error messages whereas `journalctl -p crit` displays critical messages. The log levels are the usual syslog log levels as documented in syslog(3) which you can find online, for instance on Wikipedia [4]. Both numeric and text values are accepted.

Analysis Options

Before we had systemd, if your system didn't recognise an external drive such as a USB stick, your only choice was to examine the output of `dmesg` using the command `tail -n 10`. This would show the last ten lines of output, hopefully displaying when the stick was connected. Journalctl has a built-in function for this. Use `journalctl -n <value>` to display the last x number of lines.

System admins also used to use the command `tail -f` to keep track of logfiles. On operating systems using systemd, you can now achieve the same thing by running `journalctl -f`. By way of example, run the command `journalctl -u apache2 -f`. Use Ctrl+C to interrupt the continuous output.

Output to other formats

If you want to analyze a journal in a different way, you can output data to other formats. See Listing 7 for an example of how to save all syslog *Error* levels as a simple text file. For other file types, use the `-o` or `--output` to specify the format. The default output is `short` and matches the syslog output.

The option `-o verbose` provides a more comprehensive output with all metadata and fields. Use `-o cat` as an alternative to `short`. Use the option `-o short-monotonic` for a more precise time-stamp. This will allow a more reliable comparison of outputs from different sources.

For further analysis of journal data using web tools, you can use the output options `-o json` or `-o json-pretty`. If you simply want to send the journal over the network in binary format, use `-o export`.

Safety and security

As efficient as the binary files are, if the logs do become corrupted, there's currently no way to repair them. However, their contents will usually be preserved. You can use your built-in Linux commands such as `strings` and `grep` to filter only uncorrupted data, as shown in Listing 8.

Conclusion

As soon as you've come to grips with the *journald* daemon, you'll never want to go back to a time before systemd. The logs are not only more detailed, but start much earlier during system boot. They're both more comprehensive and easier to analyze.

If you want to use syslog's non-binary format, change your configuration options in `/etc/systemd/journald.conf` to `ForwardToSyslog=yes` and `MaxLevelSyslog=debug`. In order to do this the `rsyslogd` [5] daemon must be running. You can also use the `omjournal` tool to insert messages from syslog [6]. ■■■

LISTING 7: Saving All Syslog Error Levels

```
$ sudo journalctl -b -p err --no-pager >journal.txt
```

LISTING 8: Filtering Only Uncorrupted Data

```
$ sudo strings /var/log/journal/ID | grep -i search string
```

INFO

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

[2] SysVinit: *https://en.wikipedia.org/wiki/Init#SYSV*

[3] Socket: *https://en.wikipedia.org/wiki/Network_socket*

[4] Syslog security level: *https://en.wikipedia.org/wiki/Syslog#Severity_level*

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

[6] omjournal: *http://www.rsyslog.com/doc/omjournal.html*

Reading hardware information with I-Nex

Information Desk

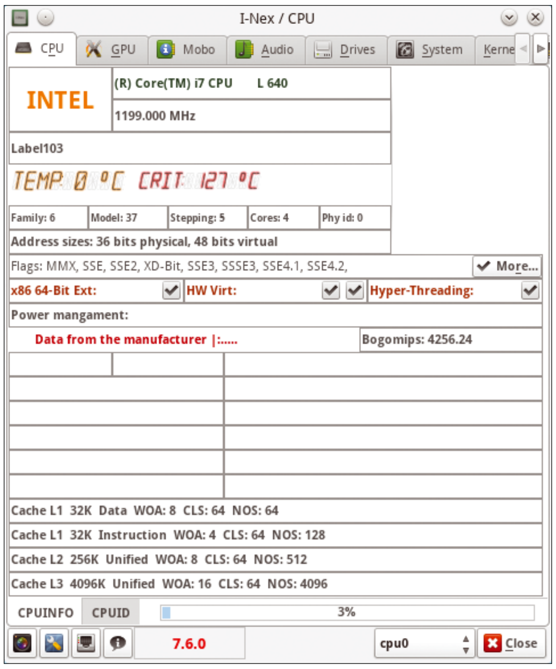
I-Nex is a graphical tool that quickly gives you a detailed overview of the hardware installed in your computer.

By Erik Bärwaldt

All major distributions and often also the desktop environments come with tools that help users discover more or less detailed information about the hardware of their computers. However, you may need to use several programs to gain information relating to drivers, kernel parameters, and hardware. Most tools deliver no technical specifications at all for certain components, such as motherboards and laptop batteries. The workaround involves a time-consuming search on the Internet.

LISTING 1: Install I-Nex

```
$ sudo add-apt-repository
ppa:i-nex-development-team/stable
$ sudo apt-get update
$ sudo apt-get install i-nex
```



I-Nex / CPU

CPU | GPU | Mobo | Audio | Drives | System | Kerne

INTEL	(R) Core(TM) i7 CPU L 640
	1199.000 MHz

Label103

TEMP 0 °C CRIT 127 °C

Family: 6	Model: 37	Stepping: 5	Cores: 4	Phy id: 0

Address sizes: 36 bits physical, 48 bits virtual

Flags: MMX, SSE, SSE2, XD-Bit, SSE3, SSSE3, SSE4.1, SSE4.2, ✓ More...

x86 64-Bit Ext: ✓ HW Virt: ✓ ✓ Hyper-Threading: ✓

Power mangament:

Data from the manufacturer |:..... Bogomips: 4256.24

Cache L1 32K Data WOA: 8 CLS: 64 NOS: 64
Cache L1 32K Instruction WOA: 4 CLS: 64 NOS: 128
Cache L2 256K Unified WOA: 8 CLS: 64 NOS: 512
Cache L3 4096K Unified WOA: 16 CLS: 64 NOS: 4096

CPUINFO CPUID 3%

7.6.0 cpu0 Close

Figure 1: Tidy and informative: the I-Nex splash screen.

I-Nex gives you a detailed overview of the hardware installed in your computer and various software parameters in a far easier and faster way. The compact tool presents all relevant information in a visually appealing form.

Installation

Most distributions have I-Nex in their repositories, so you can usually set up the program on your system conveniently with just a few clicks of the mouse. The project website [1] also provides appropriate installation instructions. On Ubuntu, you set up I-Nex from the project PPA (Listing 1).

On the I-Nex page, you can pick up the source code of the application, so that you can take a look under the hood. If you are building from the source code, I-Nex needs you to resolve numerous dependencies, including Gambas, Perl, and Python.

After successfully completing the installation, you will find two new entries in the menu structure of the desktop: *I-Nex* and *I-Nex Library*. The latter provides information about the libraries used by I-Nex, which the program loads to help it detect hardware and software. You will find the data for the computer in the graphical front-end *I-Nex* (Figure 1).

The clear-cut program window summarizes the information on the hardware, the kernel, and key components of the distribution in almost horizontally arranged tabs. In addition, there is a load indicator in the form of a colored bar with a percentage in the *CPU* tab.

Informative

Clicking on the *CPUID* button at the bottom of the program window opens a new view that informs you of the processor speed, memory size, and hardware support for certain technologies, such as command extensions for multimedia or virtualization.

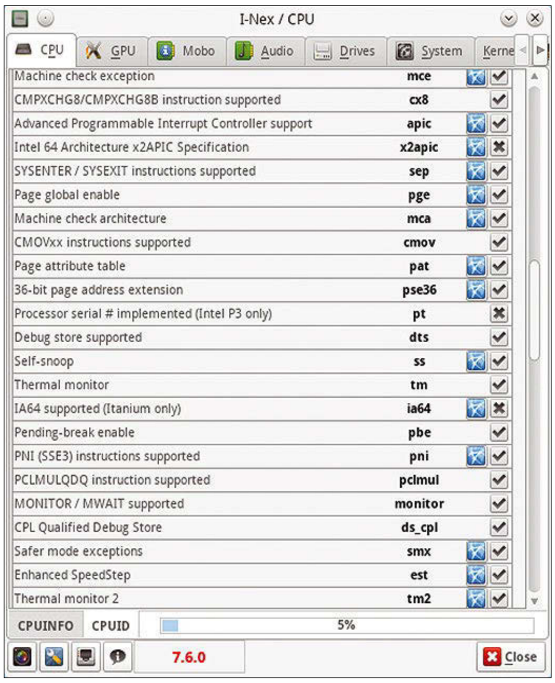
Because modern Intel processors come with a variety of firmly implemented extensions, the list is accordingly extensive.

To view information on each point, click on one of the small blue squares: Doing so follows a link that invokes the page associated with the term on Wikipedia in each case (Figure 2). Clicking on the tab *CPUINFO* bottom left in the window then returns you to the primary screen with the CPU data display.

The *GPU* tab delivers data on the graphics adapter and the appropriate drivers and modules. If you use a device with multiple graphics cards, say a laptop with a dedicated graphics chip and a chipset-integrated, power-saving GPU, you can use the appropriate selection box bottom right in the program window to toggle between them.

For detailed information about the display and its capabilities via Extended Display Identification (EDID) press the *Force* button at the bottom center in the window. Among other things, this information helps you find spare parts for notebook displays, because the panel manufacturer with accurate type data for the screen is displayed in the EDID data (Figure 3).

In the third tab, *Mobo*, I-Nex shows you all the details it has discovered for your computer's motherboard. The program shows this clearly in a table organized by the categories *Board*, *BIOS*, *Chassis*, and *Product*.



I-Nex / CPU

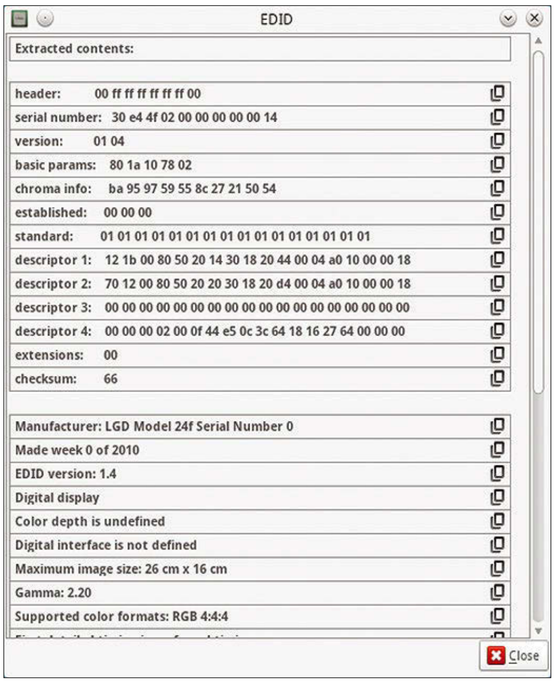
CPU | GPU | Mobo | Audio | Drives | System | Kerne

Machine check exception	mce	✓
CMPXCHG8/CMPXCHG8B instruction supported	cx8	✓
Advanced Programmable Interrupt Controller support	apic	✓
Intel 64 Architecture x2APIC Specification	x2apic	✗
SYSENTER / SYSEXIT instructions supported	sep	✓
Page global enable	pge	✓
Machine check architecture	mca	✓
CMOVxx instructions supported	cmov	✓
Page attribute table	pat	✓
36-bit page address extension	pse36	✓
Processor serial # implemented (Intel P3 only)	pt	✗
Debug store supported	dts	✓
Self-snoop	ss	✓
Thermal monitor	tm	✓
IA64 supported (Itanium only)	ia64	✗
Pending-break enable	pbe	✓
PNI (SSE3) instructions supported	pni	✓
PCLMULQDQ instruction supported	pclmul	✓
MONITOR / MWAIT supported	monitor	✓
CPL Qualified Debug Store	ds_cpl	✓
Safer mode exceptions	smx	✓
Enhanced SpeedStep	est	✓
Thermal monitor 2	tm2	✓

CPUINFO CPUID 5%

7.6.0 Close

Figure 2: This screen shows you the capabilities of your computer's CPU.



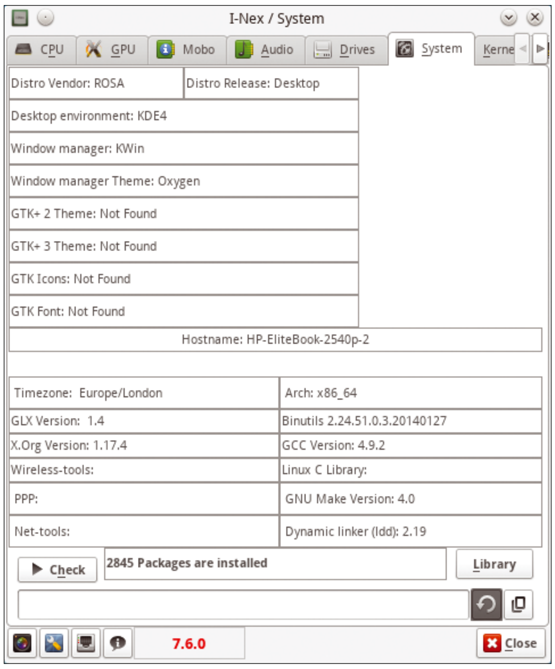
EDID

Extracted contents:

```
header:         00 ff ff ff ff ff ff 00
serial number:  30 e4 4f 02 00 00 00 00 00 14
version:        01 04
basic params:   80 1a 10 78 02
chroma info:    ba 95 97 59 55 8c 27 21 50 54
established:    00 00 00
standard:       01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
descriptor 1:   12 1b 00 80 50 20 14 30 18 20 44 00 04 a0 10 00 00 18
descriptor 2:   70 12 00 80 50 20 20 30 18 20 d4 00 04 a0 10 00 00 18
descriptor 3:   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
descriptor 4:   00 00 00 02 00 0f 44 e5 0c 3c 64 18 16 27 64 00 00 00
extensions:     00
checksum:       66

Manufacturer: LGD Model 24f Serial Number 0
Made week 0 of 2010
EDID version: 1.4
Digital display
Color depth is undefined
Digital interface is not defined
Maximum image size: 26 cm x 16 cm
Gamma: 2.20
Supported color formats: RGB 4:4:4
```

Close

Figure 3: I-Nex provides detailed data about the hardware – the display here.



Distro Vendor: ROSA | Distro Release: Desktop
Desktop environment: KDE4
Window manager: KWin
Window manager Theme: Oxygen
GTK+ 2 Theme: Not Found
GTK+ 3 Theme: Not Found
GTK Icons: Not Found
GTK Font: Not Found
Hostname: HP-EliteBook-2540p-2

Timezone: Europe/London	Arch: x86_64
GLX Version: 1.4	Binutils 2.24.51.0.3.20140127
X.Org Version: 1.17.4	GCC Version: 4.9.2
Wireless-tools:	Linux C Library:
PPP:	GNU Make Version: 4.0
Net-tools:	Dynamic linker (ldd): 2.19

2845 Packages are installed

Figure 4: **I-Nex also delivers information about the Linux distribution.**

Comprehensive

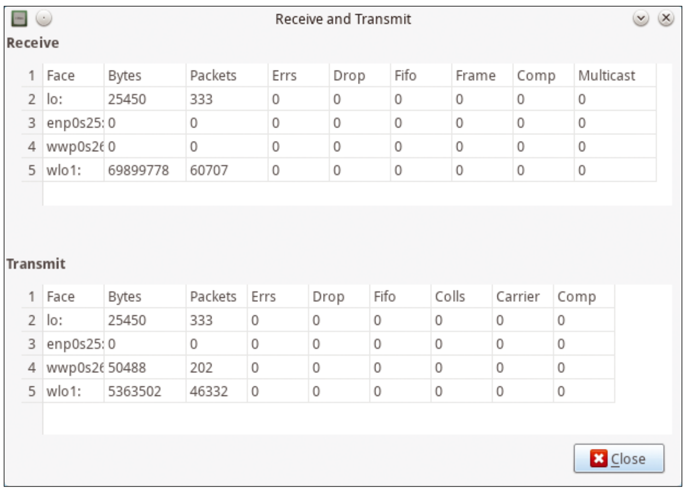
In the fourth tab, you will find information on the driver module depending on the audio hardware you have configured. At the top of the display, select the driver in question in a selection box, as long as the system supports it, if multiple sound cards are installed in the computer. These highly detailed displays are particularly useful if you want to take advantage of professional audio software and configure it optimally.

The *Drives* display is another important panel; it supplies information on the internal or externally attached storage. Combo boxes at the bottom of the window let you select the desired disk and the corresponding partition. I-Nex does not offer hardware tests for your mass storage; you will need additional software such as the Smartmontools and their graphical front-end Gsmartcontrol.

Basic info about the installed operating system is provided in the *System* tab. Here, I-Nex presents data on the desktop environment, the deployed window managers, the themes, and the installed distribution. You will also find information on the version number of the key components, such as GCC, GLX, and X.org (Figure 4). Pressing the *Check* button additionally tells the program to determine how many packages are installed on the machine.

The *Kernel* tab summarizes various pieces of information relating to the operating system kernel, including a list of boot parameters. It offers the possibility of analyzing problems caused by startup parameters in the GRUB boot manager.

The *Memory* tab displays an overview of the memory physically present in the system as well as its utilization. I-Nex permanently updates the data at intervals of a few seconds. You can thus see the size of the shared cache with buffers and the volume of data the system has outsourced to swap.



Receive

1	Face	Bytes	Packets	Errs	Drop	Fifo	Frame	Comp	Multicast
2	lo:	25450	333	0	0	0	0	0	0
3	enp0s25:	0	0	0	0	0	0	0	0
4	wwp0s26	0	0	0	0	0	0	0	0
5	wlo1:	69899778	60707	0	0	0	0	0	0

Transmit

1	Face	Bytes	Packets	Errs	Drop	Fifo	Colls	Carrier	Comp
2	lo:	25450	333	0	0	0	0	0	0
3	enp0s25:	0	0	0	0	0	0	0	0
4	wwp0s26	50488	202	0	0	0	0	0	0
5	wlo1:	5363502	46332	0	0	0	0	0	0

Figure 5: **I-Nex clearly summarizes data on the network interface.**

Network

The *Network* tab summarizes the details of the network interfaces. Select the respective interface in a selection box top left. In case of problems with the network connection, pressing *Show Receive and Transmit* displays a table with a list of all transmitted packets, sorted by interface and package type. The more packages that occur in the *Errs* and *Drop* groups, the more urgent the need is to validate your network configuration (Figure 5).

In the next tab, *USB | Input*, you can see information about USB devices and additionally check the function of various buttons. These are interesting, in particular, in combination with ACPI events on mobile devices: Some hardware still has problems with the correct configuration of various ACPI states.

The *Battery* tab is reserved for mobile devices and convertibles: This is where you will find detailed information about the physical properties of the battery/batteries installed in the device. If the battery electronics return the appropriate value, you can check the *Cycle Count* to see how many charge cycles the battery in your laptop has gone through. This is a good way of anticipating an imminent replacement.

Because there is still no binding firmware standard for the battery electronics, the corresponding fields are occasionally empty because of a lack of support by the manufacturer. The values that appear below the headings *Charge full:* and *Charge full design:* give you evidence of defective cells in the battery, or the battery's impending failure in conjunction with the charge indicator at the bottom of the program window.

Conclusions

I-Nex brings together a huge amount of information about the hardware and essential data for the operating system under a single, nicely designed interface. The software itself requires very little in the way of resources. Thanks to this useful compilation of data, the program also provides information on possible causes of issues. The software is thus especially useful for users who want a detailed graphical overview of their systems. ■■■

INFO

[1] I-Nex: *http://i-nex.linux.pl*

Multilingual programming for retrieving web pages

Tower of Babylon

We show you how to whip up a script that pulls an HTTP document off the web and how to find out which language offers the easiest approach. *By Mike Schilli*

Few programming tasks illuminate the differences between commonly used languages as clearly as that of retrieving a web document. When it comes to shell scripts, admins often turn to the `curl` utility, which transfers the data behind a URL without much ado and sends them to the standard output.

MIKE SCHILLI

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

But, what if the URL points to a black hole? Or the server denies access? And what if the server returns a redirect? For example, `curl http://google.com` does not return the expected HTML page with the search form but just a note that the desired page may be available on `www.google.com`. Armed with the `-L` option, however, `curl` follows the reference and then returns the data from the source it finds there.

What happens with a huge file like a 4K movie containing many gigabytes of data? Will the process exhaust your RAM because it attempts to swallow everything in a single gulp? Does encryption work automatically for an HTTPS URL using the SSL protocol, and does the utility check the server's certificate correctly so that it does not fall victim to a man-in-the-middle attack? Similar to good old `curl`, popular programming languages offer all of this, although often only as an add-on package and often requiring quirky approaches.

Go, Hipster Go!

The relatively new Go comes with web support out the box; it is included in the `net/http` package with exemplary SSL support. Go programmers are required to handle any errors that occur immediately

Lead Image © Maksym Shevchenko, 123RF.com

LISTING 1: http-get.go

```
01 package main
02 import "fmt"
03 import "net/http"
04 import "io/ioutil"
05
06 func main() {
07     resp, err := http.Get("http://google.com")
08
09     if err != nil {
10         fmt.Printf("%s\n", err)
11         return
12     }
13
14     if resp.StatusCode != 200 {
15         fmt.Printf("Status: %d\n",resp.StatusCode)
16         return
17     }
18
19     defer resp.Body.Close()
20     body, err := ioutil.ReadAll(resp.Body)
21     if err != nil {
22         fmt.Printf("I/O Error: %s\n", err)
23         return
24     }
25
26     fmt.Printf("%s\n", body)
27 }
```

LISTING 2: http-get.py

```
01 #!/usr/bin/python3
02 import requests
03
04 try:
05     r = requests.get("http://google.de");
06     if r.status_code == 200:
07         r.encoding = 'utf-8'
08         print(r.text)
09     else:
10         print(r.status_code)
11 except Exception as exp:
12     print("Error: " + str(exp))
```

LISTING 3: http-get.rb

```
01 #!/usr/bin/ruby
02 require 'net/http'
03
04 url = URI.parse('http://www.google.com')
05
06 begin
07   rsp = Net::HTTP.get(url)
08   puts rsp
09 rescue Exception => e
10   puts "Error: " + e.message
11 end
```

after function calls and cannot excuse any lapses by claiming that any exceptions that are thrown will be dealt with somewhere down the line, be it as hard-to-read stack traces when the program terminates. This is no doubt a philosophical question with a similar scope to finding your perfect partner or choosing between the `vi` and `emacs` editors.

Listing 1 [1] shows three different error checks, because various things can go wrong while fetching a document: the act of generating the request (e.g., unsupported protocol), the server returning an error status code (404 for document not found), or an error occurring when retrieving the data via the convoluted pipes of the Internet, which could lead to the data stream terminating abruptly. Functions in Go in general like to return two parameters, a result and an error structure, whose value is set to `nil` if everything has gone smoothly.

It is also interesting to note that the `net/http` package first executes the request, and then receives the status code of the server in line 14, but still allows some time before scooping off the body data. In fact, the data is grabbed later on by `ReadAll()` in line 20, courtesy of the *io/ioutil* package. The `Close()` method on the object body then finally indicates that request processing has finished and that Go can dispose of the data. The related command already can be seen in line 19 but is delayed till the end of the currently executing function by the nifty `defer` keyword.

Lean Python

The Python 3 implementation in Listing 2 using the state-of-art `requests` library is more compact, because Python throws exceptions that developers can test for later on in a central location. If the request specifies a nonexistent protocol (e.g., `abc://`), verification of the server's SSL certificate fails, an I/O error occurs, then an appropriate exception is thrown, which the code either checks separately or in one fell swoop as shown in line 11.

This approach is certainly more convenient, but the typing you saved here can come back like a boomerang later on if an exception is washed up from the depths of the code, like from inside a library developed by someone else. Also the readability suffers because it is not entirely clear which line in the `try` block threw the exception. The topic has started already countless, hard-to-settle, and virtually infinite discussions.

It is also funny that `r.encoding` states that the page is encoded in ISO-8859-1 after a request for `google.de`. Only manual shift to `utf-8` in line 7 causes the following `r.text` to output the content with UTF-8 encoding. A status code other than 200 does not throw an exception by default and needs to be checked manually. Programmers who like things even more compact can use the `raise_for_status()` method to throw an exception if the server reported something other than 200.

Ruby Catchall

Ruby also comes with a built-in HTTP module named `net/http`; however, before sending requests, it needs to analyze the URL with another class, `URL`. This is far too much work for Ruby-on-Rails fans and prompted the developers to create some Ruby Gems that do the whole thing in a single action. Like Python, Ruby throws exceptions from the depths of the code, and developers can thus deal with errors further upstream.

Listing 3 shows the catchall block, which starts with `begin` in line 6, catches exceptions as of `rescue`, and finally terminates with `end`. The standard library does not follow redirects in its default setting. It interprets the Google.de website as `ASCII-8BIT`, which is probably equivalent to the ISO-8859-1 encoding identified by Python in this case.

Functional Node.js

If you want to retrieve a URL in a snippet of JavaScript in the browser, or do something similar in Node.js code on the server side, for example, on an Amazon Lamda Server [2], you need to toggle your brain to functional programming mode. After all, event-based systems do not follow the paradigm of “Do this, wait until it is finished, then do that.” Instead, they want to receive their instructions in the form of “Do this, then this, then this… and go.”

The reason for this is the event loop, which can only perform short callbacks and then wants the control back. It then drops in again when the data slowly flutters in from external interfaces. This structure complicates the readability of your code and requires much experience in the design of software components so that they interact well and in an easily maintainable way.

The dreaded pyramid of doom [3], composed of nested callbacks, can be resolved by several helper constructs. Node 7.6 now even comes with support for the `async` and `await` keywords, which force asynchronous code into a synchronous straightjacket to make things look tidier [4].

Listing 4 shows a `get` call of the HTTP module in Node.js. In addition to the URL for the web document, it expects a function. This is called later with a response object and defines a closure with a variable (`content`) and three callbacks for the events `data`, `error`, and `end`.

The `data` event gets triggered whenever a bunch of data arrives from the server. It collects the data chunks one by one and reassembles them in the `content` variable. The `error` callback gets involved in case of an error and writes the reason to the log in Line 11. When the server signals the end of the transmission, the event loop jumps to the `end` callback, which in line 15 outputs the content of `content`, where all the body data in the HTTP response is now located. The Node.js `http` library automatically follows redirects.

Good Old Perl

Good Old Perl traditionally retrieves web documents with the CPAN LWP::UserAgent module. SSL support is not automatic but gets magically added if the admin retroactively installs the CPAN LWP::Protocol::https module, which depends on the availability of an OpenSSL installation and a list of root certificates.

Listing 5 shows also a peculiarity as well as correct error handling: Like some other libraries presented here, it automatically follows redirects and identifies the encoding of `google.de` as `ISO-8859-1`, but it returns a UTF-8 string from `decoded_content()` (as opposed to `content()`). That is a good thing, because processing the data in the program code often relies on UTF-8 and otherwise causes ugly-looking mangled text problems.

To output a UTF-8 string as such without modification using `print`, the script first needs to tell stdout to select on UTF-8 mode with the help of `binmode`. This rather elaborate procedure is owed to compatibility reasons and at least ensures that old scripts from the early days of Perl’s UTF-8 support don’t freak out when they meet the new versions of Perl.

Yeah, old age is not a piece of cake, when all of your joints are aching and the young folks are turning somersaults! ■■■

LISTING 4: http-get.js

```
01 var http = require('http');
02
03 http.get("http://google.com", function(res) {
04     var content = '';
05
06     res.on('data', function(data) {
07       content += data;
08     });
09
10     request.on('error', function(err) {
11         console.log( err );
12     });
13
14     res.on('end', function() {
15       console.log( content );
16     });
17   });
18 }
```

LISTING 5: http-get.pl

```
#!/usr/local/bin/perl -w
use strict;
use LWP::UserAgent;

my $ua = LWP::UserAgent->new;

my $resp = $ua->get( "http://google.de" );
if( $resp->is_error ) {
    die "Error: ", $ua->message;
}
binmode STDOUT, ":utf8";
print $resp->decoded_content;
```

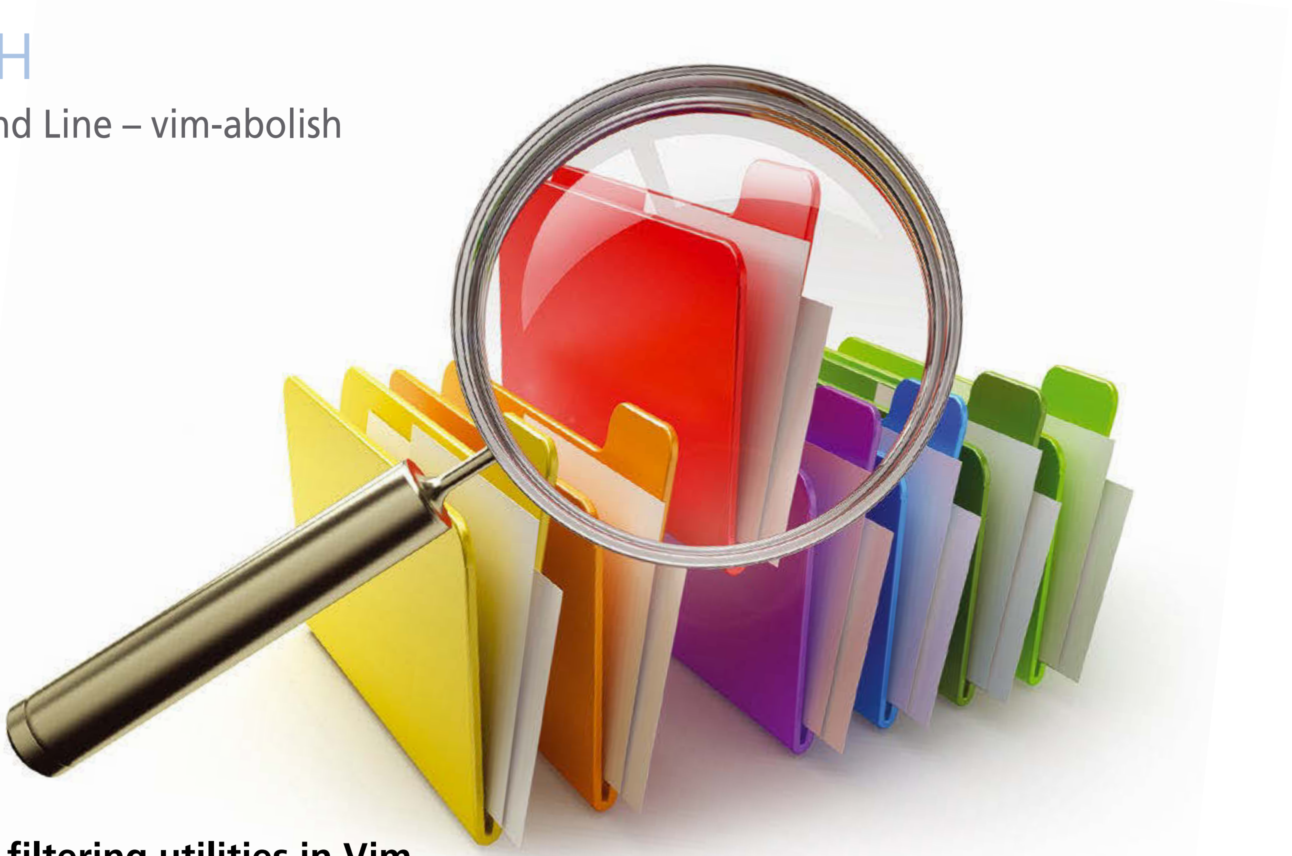
INFO

[1] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/magazine/201*

[2] “Equipping Alexa with Self-Programmed Skills” by Michael Schilli, *Linux Magazine*, issue 199, June 2017: *http://www.linux-magazine.com/Issues/2017/199/Programming-Snapshot-Alexa*

[3] “Pyramid of Doom” by Mike Schilli, *Linux Magazine*, issue 170, January 2015: *http://www.linux-magazine.com/Issues/2015/170/Perl-Asynchronous-Code*

[4] “Node 7.6 Brings Default Async/Await Support” by Sergio De Simone: *https://www.infoq.com/news/2017/02/node-76-async-await*



Advanced filtering utilities in Vim

Efficient Search and Replace

Whether you are writing code or text, vim-abolish can help you customize search and replace functions in Vim. *By Bruce Byfield*

Traditionally, vim-abolish is introduced by saying that what it does is hard to explain. This tradition dates back to its creator Tim Pope, who introduced it by saying that "I've deferred release, primarily because it's so gosh darn hard to explain" [1]. However, while exactly what is being abolished is uncertain, vim-abolish can best be described as advanced search and replace functions, roughly analogous to but less powerful than the command `sed`. It consists of four utilities: abbreviation (word completion), substitution, search, and coercion (case change). With a bit of organization, you can set up these utilities to use multiple forms of a word, such as different tenses, prefixes, suffixes, or spellings. With these tools, vim-abolish is equally useful for writing code or text documents.

Vim-abolish is installed by downloading the plugin from its homepage [1] to `~/.vim` or `~\vimfiles`, depending on your distribution. If you are using the plugin manager Pathogen, you can install Git and run the commands:

```
cd ~/.vim/bundle
git clone git://github.com/tpope/⮐
  vim-abolish.git
```

The command installs the executable to `~/.vim/bundle/vim-abolish`. From within Vim, you can run `:help Abolish` for examples of how to use each utility (Figure 1), including some complex search patterns [2].

The Command Structure

Vim-abolish offers two command structure choices. The most verbose command structure begins with `:Abolish` and varies with each utility. For example, if you are using the abbreviation utility (see below), the structure would be:

```
:Abolish [options] {abbreviation} ⮐
  {replacement}
```

However, if you are using the search utility, the structure becomes:

```
:Abolish! -search OPTIONS PATTERN
```

The other command structure option, `:Subvert`, can also vary, but tends to be shorter and more consistent than `Abolish`. It can be used for substitution, search, and coercion, but not abbreviation. For example, `Subvert`'s basic search command structure is:

```
:Subvert/PATTERN/FLAGS
```

Even more importantly, the command can be abbreviated to

BRUCE BYFIELD

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

```
:S/PATTERN/FLAGS
```

which also has the advantage of using fewer, lesser used keys that might tempt you to look down at the keyboard rather than the screen.

With both the `Abolish` and `Subvert` commands, you can use comma-separated strings in curly braces so that the command covers variations of the basic pattern, such as:

```
:S/gues{s,sed,stimate}/
```

In this example, the basic pattern of *gues* is supplemented by the flags or variations of *s*, *sed*, and *stimate*. With this structure, the command will find *guess*, *guessed*, and *guesstimate* in both lowercase, title case, and uppercase. A command can be any number of patterns, each with its own completion flags. For instance, the help file gives the example

```
:Abolish {despe,sepa}rat⮒
  {e,es,ed,ing,ely,ion,ions,or}
```

which finds about 50 words, including *desperate*, *separate*, *separates*, and *separated*.

Such patterns make vim-abolish concise but also require careful planning. At the risk of not using the plugin's full power, new users should probably begin with simple patterns, avoiding more complex ones until they are comfortable with the command structure.

Abbreviation

Abbreviation is a more powerful version of Vim's substitution command. It creates autocorrect entries that can be stored for use with other files. Abbreviation entries can save typing or correct typos, serving as a personalized spell checker.

The command structure is:

```
:Abolish OPTIONS ABBREVIATION REPLACEMENT
```

For instance, once you enter the command:

```
:Abolish cmns communications
```

cmns is changed to *communications*. If you type *Cmns*, it becomes *Communications*, and *CMNS* becomes *COMMUNICATIONS*. No extra command structure is needed for this support of different letter cases. However, both the abbreviation and the correction must be a single word, and you cannot use single or double quotation marks to make the command treat a phrase as a word.

Similarly, if you regularly type *fro* for *for*, you can correct it automatically with:

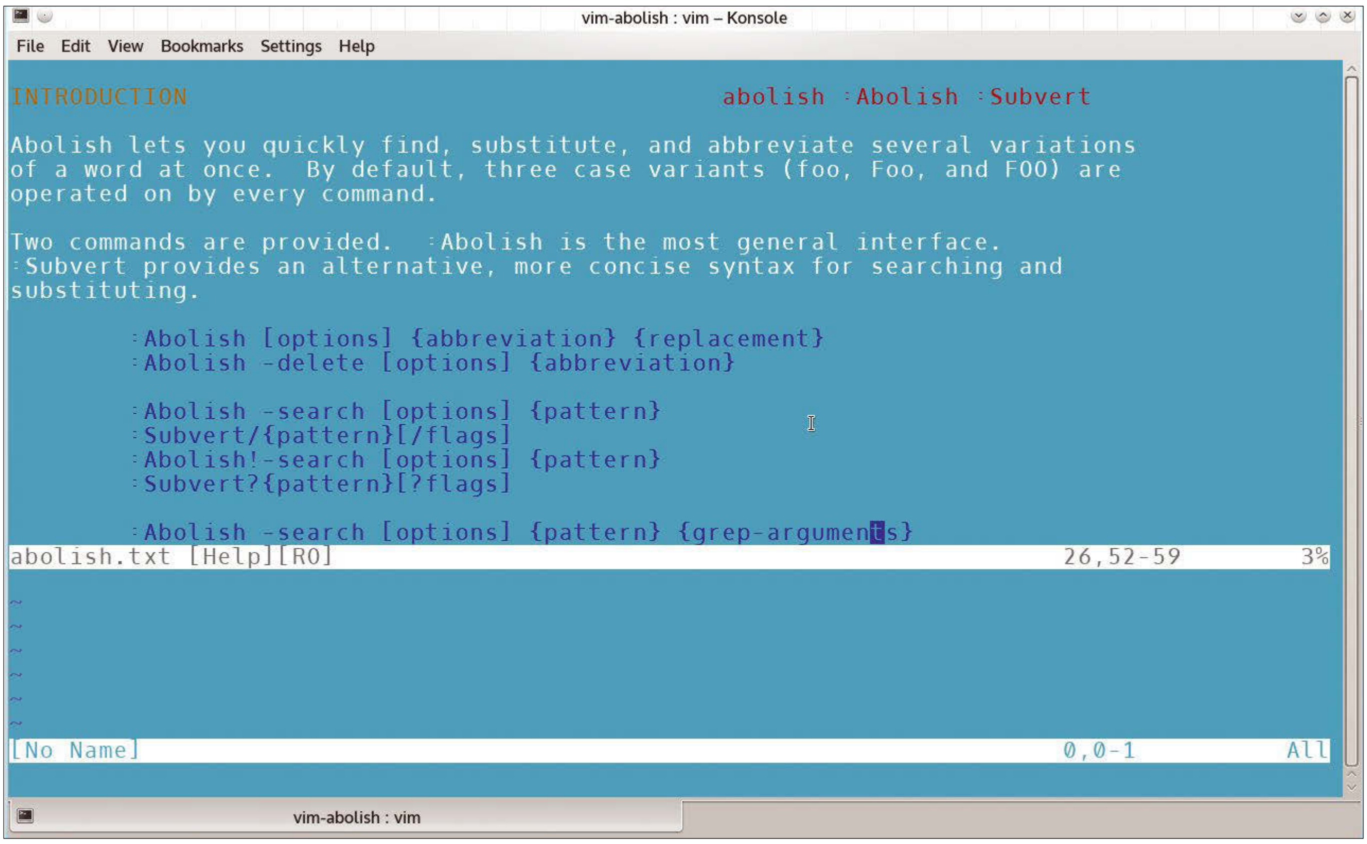
```
:Abolish fro for
```

By default, abbreviations work only in Vim's Insert mode while the current document is open. However, if you add the option `-buffer` after the basic command, the abbreviation will work throughout the current buffer as well.

Similarly, adding the option `-cmndline` will make the abbreviation work in the Vim command line, whereas adding an exclamation mark to the basic command (`:Abolish!`) saves the abbreviation for general use in Vim. Abbreviations are saved in `~./.vim/after/plugin/abolish.vim`.

To delete an abbreviation as thoroughly as possible from within Vim, enter the command:

```
:Abolish --delete -buffer -cmndline ⮒
  ABBREVIATION
```


vim-abolish : vim – Konsole
File Edit View Bookmarks Settings Help
vim-abolish : vim

```
INTRODUCTION                                    abolish :Abolish :Subvert

Abolish lets you quickly find, substitute, and abbreviate several variations
of a word at once.  By default, three case variants (foo, Foo, and FOO) are
operated on by every command.

Two commands are provided.  :Abolish is the most general interface.
:Subvert provides an alternative, more concise syntax for searching and
substituting.

        :Abolish [options] {abbreviation} {replacement}
        :Abolish -delete [options] {abbreviation}

        :Abolish -search [options] {pattern}
        :Subvert/{pattern}[/flags]
        :Abolish!-search [options] {pattern}
        :Subvert?{pattern}[?flags]

        :Abolish -search [options] {pattern} {grep-arguments}
abolish.txt [Help][RO]                                          26,52-59       3%
~
~
~
~
~
~
[No Name]                                                       0,0-1          All
```

Figure 1: Vim-abolish's help contains detailed examples of how to use the plugin's four utilities.

TABLE 1: c Flag Confirmation Options

`y`	Make the change.
`n`	Do not make the change.
`a`	Change the current instance and exit.
`q`	Exit making no other changes.
`l`	Exit, changing the current instance and returning to the first instance.
Ctrl+E	Scroll up the file.
Ctrl+Y	Scroll down the file.

TABLE 2: Letter Cases for Coercion

`crc`	CamelCase (the first letter of each syllable is capitalized)
`crm`	Mixed case (the word combines uppercase and lowercase, in no special order)
`crs` or `cr_`	Snake case (syllables are separated by an underscore)
`cru`	Lowercase Snake case
`cr-`	Dash case (syllables are separated by dashes; cannot be changed again)

The `-buffer` and `-cmndline` options, of course, can be omitted if you are sure they were not used in creating the abbreviation.

Search

The command structure for search is either:

```
:Abolish -search STRING
```

or

```
:S/STRING/
```

By default, both commands search forward from the present cursor position in the document, looping back to the beginning and continuing to the starting point. Alternatively, both `:Abolish!` or `:S?TERM?` search backward from the cursor position, looping back to the end of the document to complete the search.

Searches can be further refined with `-flags=OPTION` inserted after the `Abolish` command

```
:Abolish -flags=OPTION -search STRING
```

or placed after the search term in the abbreviated `:Subvert` command:

```
:S/STRING/ -search STRING
```

With both `Abolish` and `Subvert` searches, the `I` option disables support for case variations so that only the exact string entered is located. Additionally, `v` finds only variable names in code and `w` only complete words. These options are not always needed, but they can sometimes help to avoid false positives.

Vim-abolish's `search` command also supports grep options and regular expressions. For instance,

```
:S/gues{s,sed,stimate}/*.txt
```

finds the results *guess.txt*, *guessed.txt*, and *guesstimate.txt*.

Substitution

Substitution uses much the same options as Vim's built-in `substitute` command, which you can read about with the command `:help:substitute`. The command structure is either of the following:

```
:%Abolish -substitute -flags ⮐
  SEARCH_TERM REPLACEMENT
:%S/SEARCH-TERM/REPLACEMENT/FLAG
```

The substitution options include `I`, `v`, and `w`, just like searches. Additionally, substitution uses the `g` option to find and replace all instances of the search string (which is also the default behavior). The `e` option suppresses error messages, and `n` finds the number of instances but does not replace them.

The most useful flag is `c`. Ordinarily, vim-abolish replaces all instances without confirmation, but when the `c` flag is set, each replacement needs to be confirmed. Table 1 shows the `c` flag confirmation options. The last two might not be available, depending on how your copy of Vim was compiled.

Coercion

Vim-abolish uses the unusual term "coercion" for changing the letter case of a selected word. Search for a word, and, with the cursor still in the word, enter one of the abbreviations in Table 2 to change the word to the desired case.

If you regularly have multiple instances of words needing case changes, you might follow the advice in the `README.markdown` file, and install the command `repeat.vim` [3] to make your letter case editing easier.

Learning vim-abolish

There is no denying that vim-abolish is eccentric. Its name, as well as the names of some of its utilities, has no clear relation to its function. Nor does there seem to be a reason, other than user choice, for the support of two – three, actually – different command structures. Moreover, its four utilities have little more in common than the fact that they are all efficient ways of searching files.

Still, the eccentricity is not nearly as great as it first appears. Abbreviation is a more powerful version of Vim's substitution, and the vim-abolish `search` is an enhancement of Vim's version. You can also increase the consistency of vim-abolish by using `Subvert` rather than `Abolish` to structure your commands.

Undoubtedly, the biggest obstacle to using vim-abolish is writing advanced search patterns. Although the options are straightforward, thinking out the permutations of different word variations can be challenging. You can expect some trial and error, both to eliminate false positives and to ensure that every instance of a string is found.

Yet despite these obstacles, vim-abolish can save you time and customize Vim to your liking. These advantages make vim-abolish one of the essential plugins for many users, regardless of whether they use Vim for programming or as a general text editor. ■■■

INFO

[1] GitHub page: *https://github.com/tpope/tpope-vim-abolish*

[2] Help text: *https://github.com/tpope/tpope-vim-abolish/blob/master/doc/abolish.txt*

[3] Instructions for `repeat.vim`: *https://github.com/tpope/vim-repeat*

Ben Everard

In the technology world, there's often a constant push for something new. Companies need to tempt us with shiny products, so we continue to buy them, and they continue to make a healthy profit. One of the things that fascinates me about the open source movement is that without the profit motive, there's no need to constantly acquire the "latest-and-greatest" and no need for marketers to tell us that we need to replace perfectly working products with something new. Because of this, we keep stuff that works, even when it's old, and develop new stuff when it's needed. Modern Linux is a fascinating blend of old and new sitting side by side in a way that seems both utterly mad and perfectly sensible.

In this month's Linux Voice section, Valentine Sinitsyn shows you how to modernize your shell scripts by adding a graphical interface. Simon Phipps takes a look at a technology that manages to be both old and new – MP3s. They're 20 years old, but the simple fact of their age means that there are no longer any patent restrictions on them, so their newfound freedom has given them new relevance. Mike Saunders dives into Markdown, which is a simple formatting system that has recently become popular and given a new lease on life to ASCII (or UTF) text.

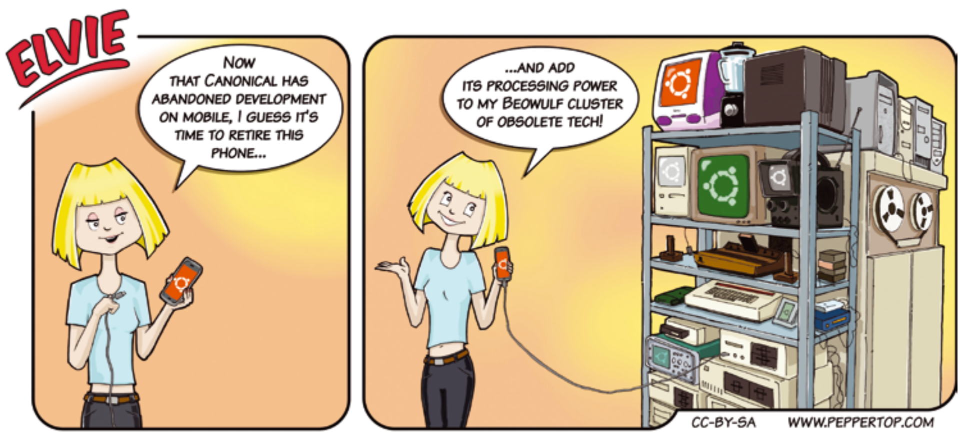
Perhaps the article most emblematic of this old-new combination of free software is in Graham Morrison's FOSSPicks. Here, there's a terminal application for viewing online maps and a re-implementation of a 20-year-old video game that runs in a web browser – old and new technologies that seem almost complete opposites, yet they sit together in the open source world, and it all just seems to make sense.

– *Ben Everard*

Andrew Gregory

Graham Morrison

Mike Saunders


ELVIE
NOW THAT CANONICAL HAS ABANDONED DEVELOPMENT ON MOBILE, I GUESS IT'S TIME TO RETIRE THIS PHONE...
...AND ADD ITS PROCESSING POWER TO MY BEOWULF CLUSTER OF OBSOLETE TECH!
CC-BY-SA WWW.PEPPERTOP.COM

LINUXVOICE

NEWSANALYSIS

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

Opinion

MP3 Is Dead, Long Live MP3

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).

Back in May, there was an unexpected surge in press coverage about the MP3 audio file format. What was most unexpected was it all declared that the venerable file format is somehow "dead." Why did that happen, and what lessons can we learn?

What actually happened was that the last of the patents on the MP3 file format and encoding process have finally expired. Based on earlier work, it was developed by the Moving Pictures Expert Group (MPEG) built on the doctoral work of an engineer at Fraunhofer Institute in Germany. Many companies held patents on the standard, and it was not until April that the last of them expired. There's no easy way to ascertain whether a patent has expired even after the date one might expect, so the wave of news arose from announcements by Fraunhofer Institute.

Framing this as an "ending" fits the narrative of corporate patent holders well, but it does not really reflect the likely consequences. Naturally the patent holding companies would rather everyone "upgrade" to the newer AAC format, which is still encumbered under a mountain of patents necessitating licensing. But for open source software, the end of patent monopolies signals the beginning of new freedoms.

MP3 has been obstructed by patents throughout its life, limiting adoption in contexts where software is community developed. Open source projects avoid patented technologies. They create challenges for both developers and users, such as

- A need to evaluate risks
- A need to identify whether any permissions are needed and, if so, to seek permission to use the format, which may lead to
- A need to obtain a license, which itself may result in
- A need to establish a business relationship – for example, to count unit shipments or user numbers

Permission-seeking steps always obstruct open source as every user of the code has to take them. That's directly opposite to the nature of open source, which is to grant all necessary permission in advance. Noncommercial terms are of no help. Open source explicitly permits commercial use of code.

Although the patents on decoding MP3 expired a while ago, it's not enough for MP3s to be free to own, free to listen to, or even free to create for personal use. As soon as there is any permission-seeking stage in the development chain, the format is compromised. Patents on the encoding process are as much as an obstacle to adoption as those in the replay process. Those are the patents that have now expired on MP3.

Now that MP3 files may be both created and played without the need to worry about patent holders, developers worldwide are finally free to innovate without needing to ask for permission. The widespread availability of both storage and bandwidth means the appeal of newer, more efficient successor formats like AAC is unlikely to be compelling. They are imperceptibly better for the average user, and they come with a need to worry about patent-holders. I think it's likely we will see MP3 encoding and decoding capabilities being more widely used than before.

Far from being dead, the MP3 format is only now finally free to live to the full. The patents are dead – long live MP3! ■■■

Framing this as an "ending" fits the narrative of corporate patent holders well, but it does not really reflect the likely consequences.

Gold in Them Thar Patients

Drying the tears of WannaCry

BY ANDREW GREGORY

Another month, another catastrophic computer hack. WannaCry, the ransomware that has cost at least £100m in canceled UK National Health Service (NHS) operations, brought the world of Ye Olde Windows XP machines to its knees, to the bafflement of many and consternation of all.

The advice given by the authorities was the same as always: Update Windows, upgrade to Windows 10 if you haven't already, and don't download any dodgy email attachments. Most media types use Macs, so the accompanying image to illustrate these stories was most often a shiny new MacBook; hardly the most likely machine to become infected with a virus written to exploit holes in Windows.

The preceding two paragraphs could have been written at any time in the last 10 years and will all too predictably be recycled in some form on many occasions over the next 10 years. But things are changing.

First of all, the argument that Windows only gets hacked more often than Linux because it's more widespread, or offers the biggest attack vector, is obsolete. If hackers wanted to attack the greatest number of machines, they'd go after Android or iOS. We can demonstrate that popularity is not the only thing making Windows vulnerable, and that a relative lack of popularity is not what's keeping Linux safe.

Also, there's genuine anger against the powers that be this time. By attacking the NHS, WannaCry's writers have hit the British public where it hurts. We'll remember, and lash out at the idiots who thought it a good idea to "save money" by running life-saving machinery on old proprietary software. The next politician who puts forward the idea of migrating to Linux will win votes, because now we (and by "we" I mean the masses, not we, the Linux users who've understood the implications of open vs. closed source software since the beginning of time) understand the consequences of taking a risk with patient records.

Linux. Oh, Linux. Now you have a competitive advantage. The business case for Linux desktop deployments is unanswerable in any organization that expects to be around for any more than 20 years. If only there were some sort of foundation dedicated to promoting Linux that could put forward a solution other than "upgrade to Windows 10." If only we had someone willing to be interviewed who could constantly use the term *Windows virus* rather than let the whole industry be libeled with the phrase *computer virus*.

Andrew Neil, who, as a journalist covering UK politics, is quite brilliant, admitted that he wanted to go into more depth on WannaCry but didn't feel he had the technical know-how to ask the right questions. He's typical of the open door against which our new foundation needs to push. Just a few press releases, a couple of quotes, a microsite, and a few estimates of how much it would cost to migrate a health trust to Linux (and of the costs of not doing so) would contribute so much to public discourse and give the right people the ammunition for when they go to bat against a government minister, health trust chief executive, or Microsoft representative.

Come on, People's Popular Foundation of Linux: We can do this! ■■■

The Ups and Downs of Ubuntu

Canonical has pitched Ubuntu as an OS for many purposes – desktops, servers, convergence, and the cloud. We look at the distribution's checkered history and where it's heading.

BY MIKE SAUNDERS

Making money from Linux is no easy task. Sure, Red Hat has done extraordinarily well over the years, and after a few bumps on the road, SUSE is looking healthy as well. But, for every financially successful distro vendor out there, you can find 10 other companies that have fallen by the wayside. Think about Xandros, Lindows (aka Linspire), and Mandriva, just to name a few. Those distros all had their strengths and benefits, but ultimately there just wasn't enough consumer demand to keep them going.

So, what have Red Hat and SUSE done right? Both companies started off with distributions aimed at hobbyists and home users – if you've been around the Linux world for a while, you may remember the chunky boxed sets with CDs and manuals you could have delivered by mail. But in the long run, Red Hat and SUSE have achieved success in large enterprises, providing long-term support and other add-on services for their distros.

With Ubuntu, Canonical has tried to get a foothold in many markets: home desktops, mobile devices, servers, and (most recently) the cloud. Some pundits have accused the company of behaving somewhat erratically over the years, jumping onto the latest bandwagon but never fully committing to a specific market. Others suggest this was the right approach – dabbling in new territories to see where money could be made.

Recently, Canonical has drastically reduced its involvement in mobile devices and convergence and now plans to focus more intensely on the cloud market, where it's the most popular operating system for virtual machines. Given this change of direction, we'll look at some highs and lows in the distribution's history to get more of a perspective of where it came from. We'll also look at some important spin-offs and what the future could hold.

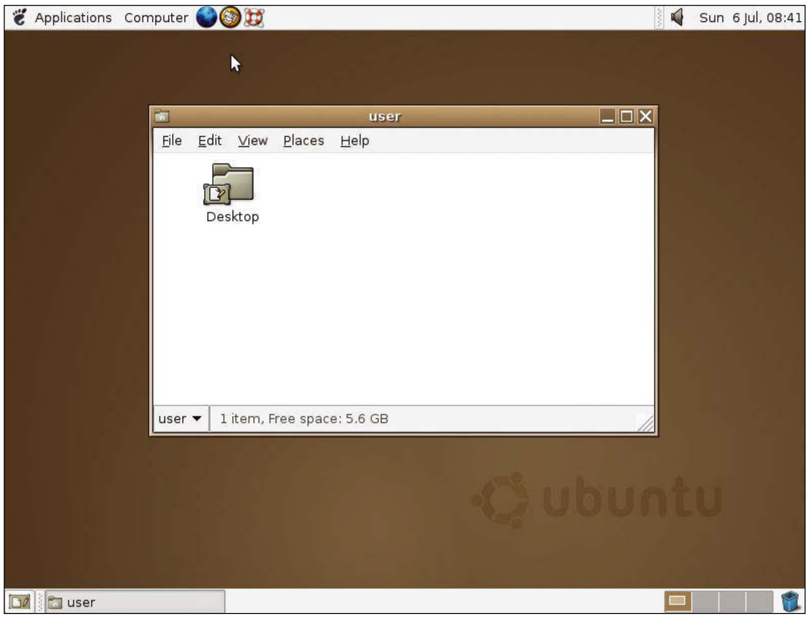


Figure 1: Here's the very first ever Ubuntu release. We quite liked the simple, no-frills desktop at the time. (Image: Wikipedia)

Not So Humble Origins

Many distros start life as one-person, part-time projects, but Ubuntu was different. From the start, in 2004, it enjoyed the backing of a man with deep pockets: Mark Shuttleworth. Having made $575 million from selling his digital signature company (Thawte) to Verisign in 1999, Shuttleworth ended up as an exceptionally rich geek, looking for new goals to achieve and projects to start. In 2002, he became the world's second space tourist, spending $20 million for a trip on a Russian Soyuz spacecraft to the International Space Station.

What prompted Shuttleworth to start Ubuntu Linux and Canonical? Well, he had previously been involved in Debian GNU/Linux – on which Ubuntu is based – back in the 1990s. In early 2004, he went on an icebreaker voyage to Antarctica, bringing with him six months of Debian mailing list archives. Shuttleworth sifted through the mails, identified potential employees for his new company, and contacted them on his return. And Ubuntu was born.

Now, I remember being rather unimpressed by the first version of Ubuntu – 4.10 – in late 2004. It wasn't a bad distro per se, but it was largely a vanilla Debian setup with some newer packages and a bit of desktop polish. It didn't have its own distro-specific tools (like SUSE's YaST configuration system or Mandriva's for-the-time extremely snazzy installer). No, it was just a passable Debian spin-off.

What did capture our attention, though, was the marketing. Shuttleworth invested $10 million into the distro, expressed his commitment to free and open source software, and worked on Ubuntu's branding and presentation. It wasn't the first distribution to be pitched as a mass-market desktop operating system, but with the financial backing, it could do a lot more. For instance, Canonical offered free CDs of Ubuntu from its website: End users without fast Internet connections (or CD burners) could receive shiny, well-presented disks in the mail, without paying a penny. Long-time Linuxers could pass on these "ShipIt" CDs to other potential users.

Consequently, it didn't take long for Ubuntu to build up a small community of followers. Even though the distro didn't have lots of custom features, it pretty much just worked, taking a reliable Debian base and providing a pretty decent out-of-the-box experience. The very early releases of Ubuntu used a rather low-key brown color scheme – like in Figure 1 – but Ubuntu 6.10 brightened things up considerably, as you can see in Figures 2 and 3.

Another attractive aspect of the distro at the time was its regular release schedule. Every six months, almost without fail, a new version would be available – with new features, updated packages, and other bits to play with. Many distros had battled with "it's ready when it's ready" syndrome, which often helps to improve stability but can cause huge delays that put off users (and potential contributors).

Canonical also designated some Ubuntu releases as "LTS" – meaning "long term support." Whereas most versions were only supported with bug fixes and security patches for 18 months, LTS versions received a whole five years of support (for server components). Similar lengthy support contracts had been available in other distros, such as Red Hat Enterprise Linux, but Ubuntu's LTS releases were easily available for all.

Controversial Moves

As time went on, however, Canonical made some decisions that furrowed the brows of many Ubuntu users (Figure 4). The distro started to feel less like a well-polished Ubuntu variant without massive changes to the upstream code and more like a very focused Canonical platform that just so happened to use freely available open source software.

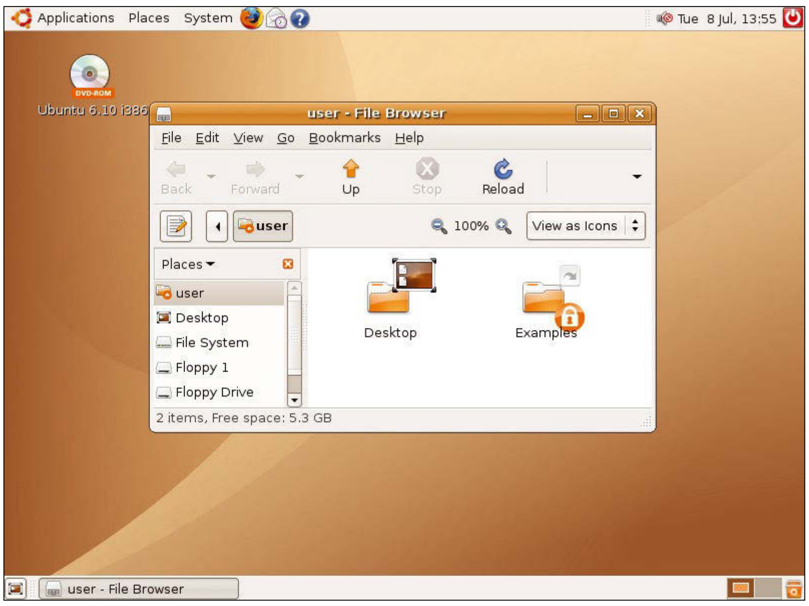


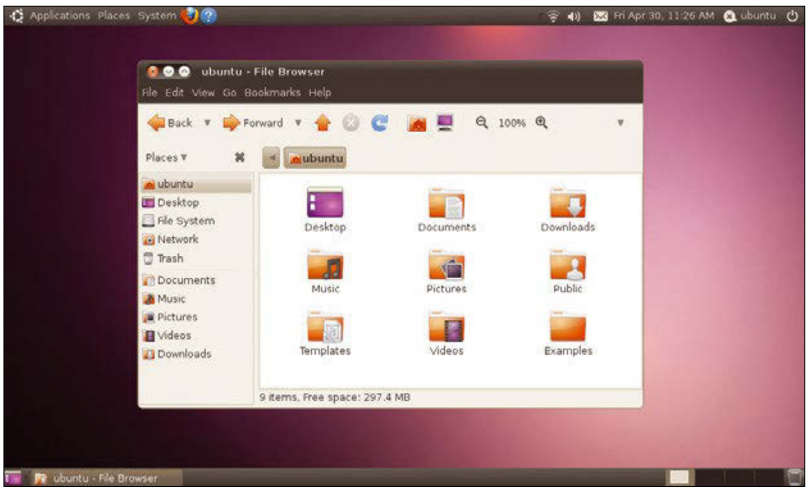
Figure 2: With Ubuntu 6.10, Canonical freshened up the design with brighter oranges… (Image: Wikipedia)

For example, in Ubuntu 9.10, Canonical introduced Ubuntu One, a file hosting service and music store that was open source on the front end but closed on the back (i.e., on the server). Users could get 5GB of storage for free but had to pay for extras. This caused plenty of consternation in the Ubuntu community: Was Canonical just going for a cash grab, having established Ubuntu as one of the most popular distros? Shouldn't the company fully commit to open source? What would happen next? (See "The Upstart That Was Upstart" for other information.)

Additionally, Canonical started to make changes to the default software. It created a new Software Center to replace Synaptic (and the various other package managers doing the rounds). Gimp was dropped from the Ubuntu 10.04 CD in favor of the simpler, but feature-limited, F-Spot, which in turn was replaced by Shotwell in Ubuntu 10.10.

Then came Unity. Since the birth of the distro, Ubuntu had used Gnome as its default desktop environment – but with Ubuntu 11.04, that

Figure 3: …but later the theme was changed again, focusing on black and purple. (Image: Wikipedia)



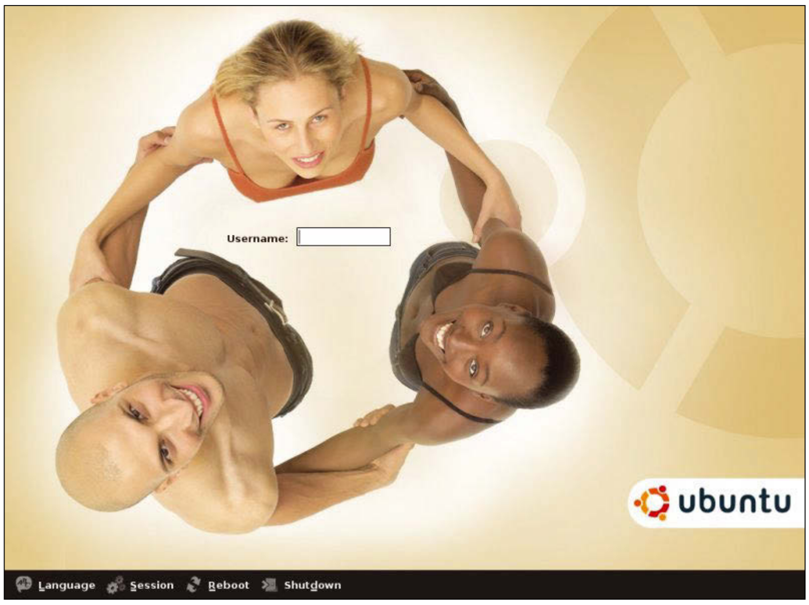


Figure 4: Earlier releases of the distro focused on "humanity" – the original definition of the word Ubuntu.

changed. Unity, a home-grown desktop shell from Canonical, was criticized as being "beta quality" and "rushed." The regular Gnome desktop was still available in the login screen as "Gnome Classic," but some long-time Ubuntu fans were dismayed that Canonical would make such a drastic change without more end-user testing.

Subsequent releases of the distro provided smoother and more refined versions of Unity, and much of the initial anger subsided. Then Canonical managed to cause another kerfuffle in Ubuntu 12.10: the Amazon search lens. When using the distro, if you tried to search for something via Unity (like a document or an application), you'd also see results from Amazon via the "More suggestions" bar. This outraged many users who saw it as a personal affront and completely against the spirit of free software. Others argued that Canonical couldn't simply live off Mark Shuttleworth's megabucks forever and would need to become a healthy, profit-making company one way or another.

We at Linux Voice remember the flame wars at the time. Shuttleworth defended the idea, saying "We're not putting ads in Ubuntu" and followed up with: "We picked Amazon as a first place to start because most of our users are also regular users of Amazon, and it pays us to make your Amazon journey get off to a faster start." He underlined the fact that search details were anonymized and sent via Canonical rather than directly to Amazon, but that didn't placate many people. This "feature" remained in place until Ubuntu 16.04, when it was disabled.

Although Unity and the Amazon search results bothered end users, some developers were getting frustrated as well. The X Window System, the base graphical layer used by every distro since the early 1990s, was due to be replaced by something lighter, faster, and more suited to modern graphics hardware. Consequently, many long-time X developers were turning their attention to Wayland as a future replacement. Canonical chose a different route, however, putting resources into a home-grown display server called Mir.

There were some valid technical arguments for this, but many in the community felt like Canonical wasn't being a team player and that, even though Mir was open source, it was an attempt to lock users and developers into a Canonical ecosystem. Shuttleworth hoped that Mir would be available to replace X in Ubuntu 15.10, but that never happened – and it never became the default in any Ubuntu release. Since Canonical's change in direction (more on that in a moment), Mir has a questionable future; Shuttleworth says "the code continues to receive investment" for embedded devices, but it will probably never be used on desktop PCs.

The Upstart That Was Upstart

Another Canonical-backed technology that caused a ruckus in the community was Upstart, a replacement for the aging SysVinit boot system that had long been used by most Linux distros. Upstart was designed to be more flexible and event-based, responding more elegantly to hot-plugged hardware and network events. It was initially released in 2006 and included in Ubuntu soon thereafter – and a few other distros (e.g., Fedora) adopted it as well.

In our opinion, Upstart was a pretty good piece of technology: It could run classic SysVinit scripts, so it was fairly easy to grasp and still felt very much Unix-like. Ultimately, though, it lost out to systemd – the mightily powerful (but arguably much more complex) boot and daemon management system that has since been adopted by almost every major distro.

Time to Make Money

Throughout the mid-2010s, Canonical had been striving to establish Ubuntu as the ideal "convergence" operating system – that is, a single OS that works well on everything from smartphones and tablets to desktops and smart TVs. Imagine having a minimalist Unity interface on your phone, to do work on the move, and when you plug the phone into a large monitor the look and feel is still very much the same. Given that smartphones were becoming mind-bogglingly powerful, we could see the potential here.

Lesser-Known Ubuntu Spin-Offs

You almost certainly know about Kubuntu, Xubuntu, and Lubuntu – versions of Ubuntu that use the KDE, Xfce, and LXDE desktops, respectively. But there are many others as well. Gobuntu [1] was a short-lived effort to make an Ubuntu-based distro that's based entirely on free software, so no binary blobs, firmware drivers, or other not-100%-free software was included. Once Gobuntu was shelved as a standalone distro, the option to install with only free software was rolled into the Ubuntu installer itself.

Cubuntu [2] is a French distro that lets you "enjoy the best of the Cinnamon desktop" – which sounds rather cool, except that Linux Mint already does very much the same thing (and appears to be more up to date). A few religion-oriented Ubuntu flavors have appeared over the years, such as Ubuntu Christian Edition [3], which seems to have stalled with version 12.04, along with Sabily, which stopped development in 2011. Both provided extra software for studying religious texts along with additional parental controls.

To read more about the myriad Ubuntu derivatives, see the wiki [4]. Many of them have fallen into bitrot now – which is understandable, given that the majority don't need to be standalone distros, but could actually be implemented as post-install scripts that tweak some settings, add some packages, and remove some others.

Canonical tried and tried, even getting a phone out of the door (Figure 5), but ultimately the duopoly of Android and iOS was hard to beat. On April 5, 2017, Shuttleworth wrote a blog entry that sounded positive on the surface (growing Ubuntu, excellent year for the company, etc.) but ultimately contained bad news for some people, saying "We will end our investment in Unity… We will shift our default Ubuntu desktop back to Gnome for Ubuntu 18.04 LTS."

After years of working on Mir, Unity, and related convergence technologies, Shuttleworth had had enough. He freely admitted that he was "wrong" to think his projects would be "widely appreciated both in the free software community and in the technology industry." You can sense the frustration and sadness in the blog post – after all, from Shuttleworth's viewpoint, he and Canonical had spent a lot of time and money trying to develop these technologies, making them open source, only for the market to largely ignore them.

Figure 5: The Ubuntu Edge was a proposed smartphone that Canonical tried to fund on Indiegogo – but the target wasn't reached.

So, Ubuntu has a new focus: the cloud. Shuttleworth is adamant that the distro will still be well supported on desktops and laptops, but cloud infrastructure and Internet of Things (IoT) are the "areas which are contributing to the growth of the company." It makes sense to concentrate on these markets, given that Ubuntu is twice as popular as all other OSs combined on Amazon's Elastic Compute Cloud.

But what's to come? We don't like peeking into crystal balls at Linux Voice, but we do suspect that Linux on the desktop has some mileage ahead. Yes, we know the jokes about "THIS will be the year of Linux taking over the desktop," and Windows isn't going anywhere, any time soon. But, with the massive WannaCry ransomware attack, forced upgrades to Windows 10, and governments becoming more aware that they should have control over their IT infrastructure, we hope Canonical still keeps working on end-user Linux distributions and doesn't just switch its attention to the cloud completely. (See the "Lesser-Known Ubuntu Spin-Offs" box for more info.) ■■■

Info

[1] Gobuntu: *https://wiki.ubuntu.com/Gobuntu*

[2] Cubuntu: *http://cubuntu.fr*

[3] Ubuntu Christian Edition: *http://ubuntuce.com*

[4] Ubuntu wiki: *https://wiki.ubuntu.com/DerivativeTeam/Derivative*

MADDOG'S DOGHOUSE

Understanding how optimization works is just as crucial to fast code as a good compiler. 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®.

Optimization

I have written about the importance of knowing how things work "from the bottom up." Usually, this includes understanding (at least at an overall level) how the hardware works and how compilers and operating systems accomplish their tasks.

Recently a university student I know was taking a course on embedded systems, and they were writing a device handler in the C language. The student had defined a global variable to hold a value that would be filled in by the hardware itself, and not initialized by any other statement. Their professor told them that they had to declare the variable "volatile," or else the compiler might "optimize" the variable outside of the scope of the module. From the code the compiler could "see," there was no way that the variable's value would change, so normal optimizations might assume that the value in the variable was the same as the first time it was set, and that registers loaded with the contents of that variable would still be valid.

Wrong.

One main way that this variable might change is if the global variable was used in an interrupt, and the hardware itself (not the programmer's code) had asynchronously changed the value inside the global variable. The second way would be if the code was meant to be multithreaded and another thread had changed the value. This would not be seen by the compiler, because in that thread the value of the thread was untouched.

In both cases, the designation as "volatile" tells the compiler not to do various optimizations to that variable and to always take the value from the variable itself even if "it has not changed."

Multithreaded programming is more common than it was years ago; there are many fewer programmers who understand (and realize the coding issues) of interrupt handling. Without the word "volatile" on that variable, the compiler might silently move the statement using the variable's value out of the scope of a loop or even a module, with the compiler assuming that it was never updated. Most programmers would look at the source code for many weeks without understanding what was happening.

Another good "optimization" story comes from Digital Equipment Corporation's (DEC) early Unix products, and particularly the days of the X Window System.

In the years 1986 to 1988, the GNU compilers were still evolving. While many people used GNU because it was readily available and had the same language syntax and semantics across many hardware architectures and operating systems, the code GNU generated was not the most optimized in the world. As an example, a commercial compiler could generate code up to 30 percent faster in execution than the code GCC would generate. However, as people saw that CPUs were getting faster and faster with costs of good programmers continuing to climb, people often just "got a faster CPU" to make up for the lack of optimization performance.

Asking the compiler to generate lots of optimizations for any architecture or operating system also made the compilation go slower and introduced "bugs" as the optimizations changed the way the code worked.

In any case, the VAX C compiler engineers decided to change their compiler from accepting just ANSI C to accepting ANSI C and GNU C, in order to recompile not only the X Window System code but perhaps even the Unix kernel itself (based upon BSD Unix v4.1 and heavily modified by DEC engineers).

It took many months and much work to get the VAX C complier to understand the nuances of the GNU C suite, but finally the VAX C people finished the project and decided to recompile the X Window Server to see how much performance improvement they could achieve.

It was close to zero improvement.

Curious about why the "optimizing compiler" did not get better performance, they looked at the source code for the X Window System Server. They realized that it had been written by very, very expert coders who anticipated what the GNU compiler was going to do and wrote their code to do "hand optimizations" to make their code very, very efficient even if the compiler was not doing that.

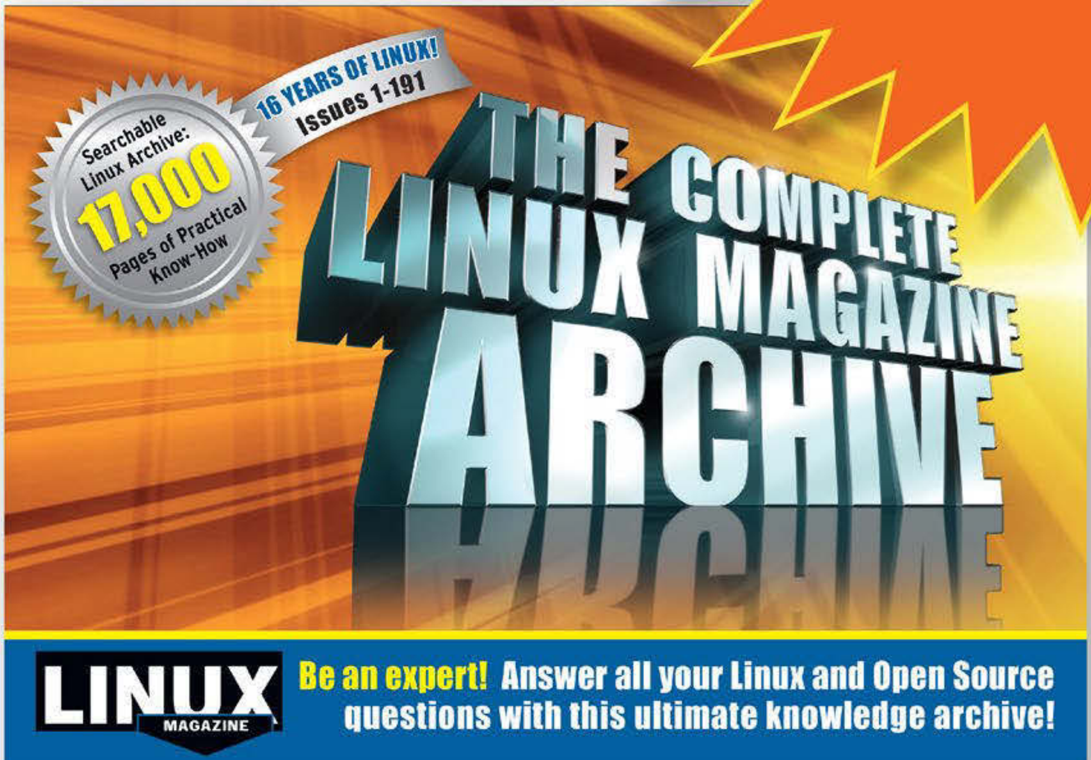
Granted, other code in the X Window System had not been written by such skilled programmers, and that code did improve with the optimizing compiler, but pieces of critical code typically did not see much improvement.

Times have changed. Computer architecture has become more complex with multiple levels of cache, multicore CPUs, and other issues that make it really difficult for even the most experienced programmer to do this type of "optimization" in their source code. However combining a knowledgeable programmer with a good optimizing compiler (and the GNU compilers have improved dramatically) can make a great deal of difference in the speed of your code. ■■■

Happy 25th Anniversary to Linux!

Help us celebrate 25 years of Linux with the All-Time Archive DVD of Linux Magazine!

Order today and get 191 issues of Linux Magazine on one handy DVD!


16 YEARS OF LINUX!
Issues 1-191
Searchable Linux Archive: 17,000 Pages of Practical Know-How
THE COMPLETE LINUX MAGAZINE ARCHIVE
LINUX MAGAZINE
Be an expert! Answer all your Linux and Open Source questions with this ultimate knowledge archive!

You get 17,000 pages of practical know-how on one searchable disc – that's 191 issues including 40+ issues that were not included in the previous release.

Order Now! Shop.linuxnewmedia.com

FAQ

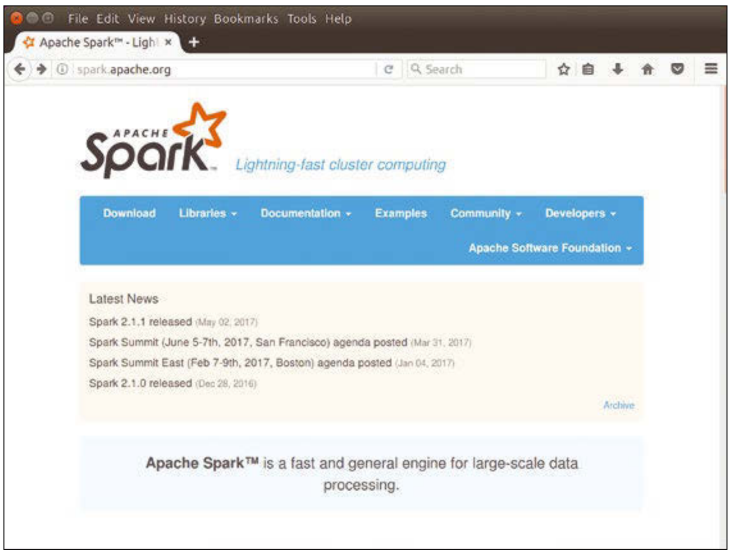
Apache Spark

Spread your processing load across hundreds of machines as easily as running it locally.

BY BEN EVERARD

Q Apache Spark? I've wanted to set fire to my Apache web server more than a few times – usually when I'm elbows-deep in a config file that just refuses to work as I want it to. Is that what it's for?

A I've been there, too, but no. The web server commonly know as Apache is officially called the Apache HTTP Server. The Apache Software Foundation manages hundreds of projects, and only a few of them are related to web servers. Apache Spark is a programming platform (Figure 1).



Latest News

Spark 2.1.1 released (May 02, 2017)
Spark Summit (June 5-7th, 2017, San Francisco) agenda posted (Mar 31, 2017)
Spark Summit East (Feb 7-9th, 2017, Boston) agenda posted (Jan 04, 2017)
Spark 2.1.0 released (Dec 28, 2016)

Apache Spark™ is a fast and general engine for large-scale data processing.

Figure 1: The project website (spark.apache.org) has all the information you need to get started.

Q But, there are hundreds of programming platforms already. What does Apache Spark have that others don't?

A To understand Spark, you have to understand the problem that it's designed to solve: Processing Big Data.

Q Ah, Big Data! The buzzword du jour of the computing industry. Let's start with the obvious question: Just how big does Big Data have to be to need Spark? Gigabytes? Terabytes? Petabytes … er … jibbly-whatsitbytes?

A With Big Data, it's better not to think of the size in terms of absolute numbers. For our purposes, we'll say that data becomes big once there's too much to process on a single machine at the speed you need it. All Big Data technologies are based around the idea that you need to coordinate your processing across a group of machines to get the throughput you need.

Q What's so hard about that? You just shuffle the data around to different machines, run the same command on them, and then you're done. In fact, give me a minute; I think I can put together a single command to SCP data to other machines and then use SSH to run the commands on them.

A You don't even need to write a complex command if you just want to load balance data processing across various machines, the Gnu Parallel tool has support for that baked in. Although this approach can work really well for some simple examples, (e.g., if you want to recompress a large number of images), it can very quickly get complex or even impossible. For example, if the processing of the data involved uses the results of some other aspects of the processing.

Q It sounds like you're just making up problems now. What real-world issues does this help with?

A Probably the best example of a problem that Spark solves is machine learning.

Q As in artificial intelligence?

A Yep. Machine learning can work in many ways, but consider this one. You push data through an algorithm that tries to intuit something about the data. However, at the same time, it's learning from the data that it sees. The two separate items, learning and processing, are happening at the same time. For this to work, there has to be a link between the separate computers doing the processing (so that they can all learn in the same way), but at the same time, they want to distribute their processing as much as possible.

Q This sounds a bit like black magic. How does Spark manage to keep a consistent, yet changing model across multiple machines?

A The concept that sits at the heart of the Spark platform is Resilient Distributed Datasets (RDDs). An RDD is an immutable collection of data. That means that once it's created, an RDD doesn't change. Instead, transformations that happen to an RDD create a new RDD. The most basic use of RDDs is something you may be familiar with: MapReduce.

Q Ah, yes. I've heard of that. I can't remember what it is though!

A Very simply, MapReduce is another paradigm for processing Big Data. It goes through every item in your dataset and runs a function on it (this is the map), and then it combines all these results into a single output (this is the reduce). For example, if you had a lot of images and you wanted to balance their brightness and then create a montage of them, the map stage would be going through each image in turn, and the reduce stage would be bringing each balanced image together into a montage.

Spark is heavily inspired by MapReduce and perhaps can be thought of as a way to expand the MapReduce concept to include more features. In the above example, there would be three RDDs. The first would be your source images, the second would be created by a transform that balances the brightness of your images, and the third would be created by the transform that brings them all together to make the montage.

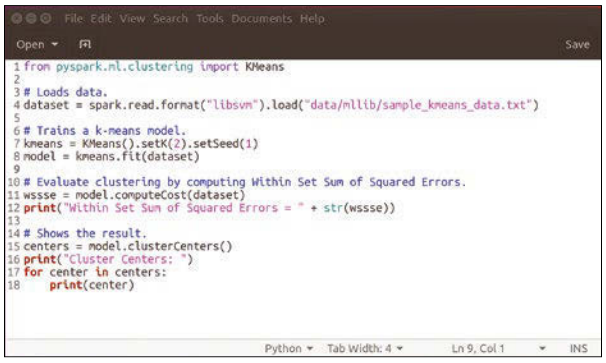
Q Ah, OK. How does this work with artificial intelligence again?

A Well, there's two answers to that. The first is complex and doesn't fit in a two page FAQ; the second is: "don't worry about that, someone's done it for you."

Q Ah, there's a library to use?

A To call Spark's MLlib a machine learning library seems to undersell it, but because that's what it's actual name stands for, we can't really argue with it. Essentially, it is a framework that comes complete with all the common machine learning algorithms ready to go. Although it does require you to do some programming, 10 lines of simple code is enough to train and run a machine learning model on a vast array of data split over many machines (Figure 2). It really does make massively parallel machine learning possible for almost anyone with a little programming experience and a few machines.

Figure 2: Distributed machine learning in 10 lines of Python (plus a few comments). It doesn't get much easier than this.



```python
from pyspark.ml.clustering import KMeans

# Loads data.
dataset = spark.read.format("libsvm").load("data/mllib/sample_kmeans_data.txt")

# Trains a k-means model.
kmeans = KMeans().setK(2).setSeed(1)
model = kmeans.fit(dataset)

# Evaluate clustering by computing Within Set Sum of Squared Errors.
wssse = model.computeCost(dataset)
print("Within Set Sum of Squared Errors = " + str(wssse))

# Shows the result.
centers = model.clusterCenters()
print("Cluster Centers: ")
for center in centers:
    print(center)
```

Q OK, so that's machine learning. Are there any other tasks that Spark really excels at?

A There's nothing else that Spark makes quite as easy as machine learning, but one other area that is gaining popularity is stream data processing. This is where you have a constant flow of data in and you want to process it as soon as it comes in (and typically send it on to some time-series database). The Spark Structured Stream framework makes it easy to perform just this sort of processing.

It's wrong to think of Spark as just for machine learning and streaming, though. These are just two areas that happen to have frameworks on Spark. The tool itself is general-purpose and can be useful for almost any distributed computing.

Q So, all I need to create my own streaming machine learning system is a couple of machines, a Linux distro, and Spark?

A Not quite. You also need some way of sharing the data amongst the machines. Hadoop's Distributed File System (HDFS) is the most popular way of doing this. This provides a way of handling more data than any one machine can handle and efficiently processing it without moving it between computers more than is necessary. HDFS also provides resilience in case one or more machines break. After all, the more machines you've got, the higher the chances are that one breaks, and you don't want your cluster to go down just because one machine has some issues.

Q Right. I'm off to build a business empire built on computers that are more intelligent than I am.

A Good luck. ■■■

Valentine Sinitsyn works in a cloud infrastructure team and teaches students completely unrelated subjects. He also has a KDE Developer account he's never really used.

CORE TECHNOLOGY

Some shell scripts are silent; others communicate to users extensively. Learn how to make their dialog smoother with, er ..., dialogs.

BY VALENTINE SINITSYN

Interactive Scripts

As an administrator, you may view shell scripts as wordless minions that do their job and die silently – unless an error occurs, of course. At least, that's what we expect from a well-behaving Unix command. There are good reasons for that (think pipelining), but it doesn't mean you must have it that way all the time.

In this Core Tech, we'll learn some tricks to make shell scripts interact with the user via command prompts and dialog boxes. This is how you got Slackware installed, right? (You did try Slackware, didn't you?) Yes, the Slackware installer is basically a shell script, and with some tools under your belt, you can do no worse than Patrick Volkerding. Sound good? If so, let's go and have some text-mode fun.

Command Prompt

Perhaps the simplest form of interactivity you can have in your scripts is a command prompt. That's how shell itself is made interactive, after all. Getting pieces of data from the user is possible with the `read` built-in command [1]. But, to make things fancier, let's throw some history support and other readline goodies into the mix.

To enable readline support in `read`, pass the `-e` switch to the built-in. This provides a bunch of keyboard shortcuts that you got used to in the shell. Tab completion works; you can move word-wise forward and backward with `C-f` and `C-b` (where `C` is typically the Control key); using `C-a` and `-e` bring you to the beginning and the end of the line, respectively. The Home and End keys should work, too. You can even use `C-r` to search the history. These are just a few examples; readline is really powerful. It comes with its own configuration file, usually `~/.inputrc`, and the complete list of commands, including default key bindings, is in the Bash man page [2].

While readline typically binds history-related keyboard shortcuts like `C-R` automatically, it needs something extra to fill in the actual history contents. First, you read the history file via `history -r ~/.script_history` at the beginning of your code. Using `history -s args` stores `args` as a single line of history. When your script ends, you can save updated history to the file via `history -w ~/.script_history`. You can also customize Tab completion behavior in your scripts, but that's another story.

Next, let's consider a simple example. What you see in a Listing 1 is what could be a command loop in some installer. Remember those proprietary packages that come as an executable shell script combined with a tarball? The script implements several install stages and rudimentary state transitions. Other functions, such as `show_license` and `ask_install_root`, are irrelevant and not shown here. As installation is typically a one-

Listing 1: Read Built-In Usage Example

```
STAGE=begin
while [ "$STAGE" != end ]; do
  case "$STAGE" in
    begin )
      show_license
      read ANSWER
      if [ "$ANSWER" = yes ]; then
        STAGE=ask_install_root
      else
        STAGE=end
      fi
      ;;
    ask_install_root )
      ask_install_root
      read -e INSTALL_ROOT
      STAGE=install
      ;;
    install )
      # Unpack to $INSTALL_ROOT
      STAGE=end
      ;;
  esac
done
```

Listing 2: Dialog Call

```
dialog --clear --exit-label Next
  --textbox /usr/share/common-licenses/GPL-3 20 80
  --and-widget --yesno "Do you accept the terms of the license?" 5 50
if [ $? -ne 0 ]; then
  echo "You must accept the license to continue"
fi
```

shot procedure, I don't make use of the history feature here. And, because it's rather trivial to type "yes" and "no" (and because our legal department wants this answer to be very explicit), I haven't enabled readline in that stage. I do enable it on the next stage though, as Tab completion for filesystem paths is typically a good idea.

Making Dialogs

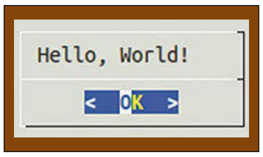
Although command prompts certainly work, they are hardly intriguing in 2017. Today, we demand a dialog-based "experience," preferably a nice graphical one, but an old-school ncurses-based approach would also do. Bash can't play such tricks on its own (at the very least, it's counter-Unix-way). But there are external commands that can.

Of course, I'm referring to the `dialog`. Although it's not the only choice, the `dialog` tool is ubiquitous and usually comes installed by default. If not, you can probably find it in the distribution repositories under this very name.

The `dialog` tool provides interface components (or widgets) you'd expect from a typical UI toolkit. I mean menus, checkboxes, text and password inputs, progress bars (called "gauges" here), file selectors, and so on. There are many widget types, but not everything is necessarily compiled into the tool. You tell `dialog` which widget you want via a command-line argument, and it is possible to display more than one widget in a single command invocation. In turn, `dialog` dumps user selections (if any) into stderr (or whatever file descriptor you supplied via `--output-fd`). Simple choices, such as "yes" or "no," are returned as exit codes. You can use environment variables to define which codes are actually returned [3]. The default is 0 (i.e., success) for buttons such as Yes or OK, 1 for No or Cancel, 2 for Help, and a few others as well.

Let's go through a simple "Hello, World!" example (Figure 1):

```
dialog --msgbox "Hello, World!" 5 20
```



```
Hello, World!
   < OK >
```

Figure 1: An obligatory "Hello, World!" example.

That's pretty straightforward. You tell dialog you want a plain message box (`--msgbox`); 5 and 20 are the dialog box height and width, respectively. For a modern flat look, consider adding the `--noshadow` argument to disable shadow.

Now, back to our wannabe-installer script. Remember the stage where it shows you the license and asks if you accept it? It's doable with a single `dialog` call (Listing 2).

Let's go through this step by step. First, we tell the first widget that we want it to clear the screen after itself. This is what you typically expect from a shell script. However, omitting it creates an interface that feels more like a real, multi-window one (Figure 2). There are some options you can use to tweak it even further, as shown in the man page [3].

Then, we redefine the label that is shown on exit buttons. It defaults to "EXIT" (surprise!), which is not what we want here, because in this case, the exit button doesn't terminate the script.

Then comes the textbox widget itself. It's basically a file viewer. Here, we show the GPL v3 license text that comes with many Debian-based distributions. Licenses are a long read, so this time the dialog is whopping 80x20 in size. And, of course, the text is scrollable.

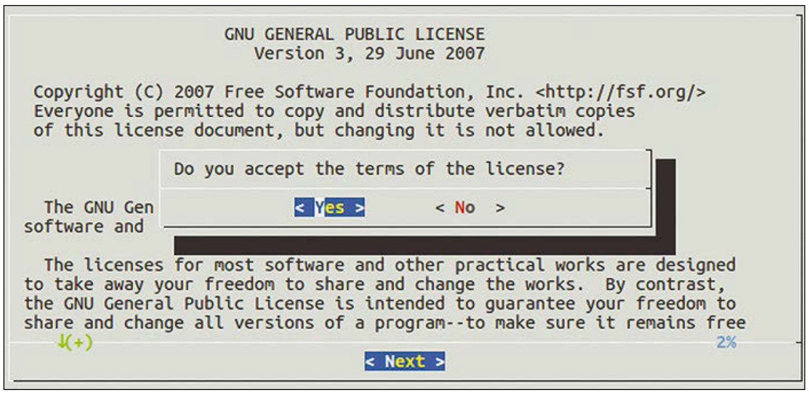
To display (unconditionally) the second widget, you prefix it with `--and-widget`. Here, it is a "yes/no box" which asks a simple question. It is also possible to override the "Yes" and "No" button labels, yet we don't need it here. To determine whether or not the user agreed, you can check the exit code. Zero means the license was accepted.

You Choose

In previous examples, interactivity was somewhat limited: a dialog box, yes or no, and that's it. This might be good for a wedding, but it doesn't build a complete user interface. It's time for users to make some real decisions.

Perhaps the most common form of choice in user interfaces is a menu. Most GUI applications these days provide a menubar, or at least a button (or ribbon). With `dialog`, you are limited to a `--menu` widget. Listing 3 explains how you create one.

The first three arguments to `--menu` should be already familiar. The next one, dubbed menu-height, is the number of menu items to show simultaneously.



```
                  GNU GENERAL PUBLIC LICENSE
                     Version 3, 29 June 2007

  Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
  Everyone is permitted to copy and distribute verbatim copies
  of this license document, but changing it is not allowed.

                 Do you accept the terms of the license?

   The GNU Gen               < Yes >        < No  >
 software and

   The licenses for most software and other practical works are designed
 to take away your freedom to share and change the works.  By contrast,
 the GNU General Public License is intended to guarantee your freedom to
 share and change all versions of a program--to make sure it remains free
       ↓(+)                                                          2%
                              < Next >
```

Figure 2: The `--textbox` widget and chaining two dialogs in one call. Note that in absence of `--clear`, dialogs are overlaid.

Listing 3: Menu Dialog Sample

```
choice=$(mktemp /tmp/choice.XXXXX)
dialog --menu 'Select an action:' 10 50 3
    Backup "Backup data"
    Restore "Restore from backup"
    Quit "Exit the script" 2> "$choice"
case $(<"$choice") in
  Backup ) ;;
  Restore ) ;;
  Quit ) exit 0 ;;
esac
```

If your menu has more, `dialog` will scroll the contents automatically.

What follows is a number of pairs consisting of a tag and a caption. You can think of a tag as the menu item identifier. When the user selects anything, you get the tag in stderr or to wherever `--output-fd` points.

Here, we build a simple four-item menu, which allows a user to either back up or restore data, get help, and exit the script. Only the latter option is really implemented. However, `--dselect` and `--fselect` [3], the directory and file selection widgets, are here to implement two remaining options. The `dialog` tool also displays both tags and captions and assigns unique single-letter hotkeys (shown in red in Figure 3) to each option. Pressing hotkeys selects the corresponding option but doesn't confirm the selection: you need to press Enter or click OK for that. I'd suggest not to using spaces within tags, as it makes things a bit more complicated. We redirect stderr to a temporary file and read it back to learn the user's choice. If it is empty, it means the user pressed Escape or otherwise canceled the script.

Menus are great for setting up one choice. But, sometimes you may want to select several options (Figure 4). The checklist widget is one way to do it with `dialog`. It is very similar to menu except it places a checkbox next to each item. You can mark items with Space and confirm your choice with Enter. Mouse should work as well (and it's not limited to a checklist widget, naturally).

Now, please have a look at Listing 4, which shows a complete example. The first four options describing the dialog appearance are the same

Listing 4: Checklist Dialog Sample

```
choice=$(mktemp /tmp/choice.XXXXX)
dialog
  --separate-output
  --checklist 'What would you like?' 10 50 3
    Eclair "A few eclairs" "on"
    Froyo "Some Frozen Yoghurt" "off"
    ICS "An Ice Cream Sandwich" "on"
    2> "$choice"
```

both for menu and checklist. Consult the description above for the refresher. Setting contents is a bit different, though. For a menu, tag and item suffice; for a checklist, we also need a way to tell which options are initially on. So, each checklist entry comes via `tag item status` triplet, where `status` is `on` if you want the option checked.

After a user is done with the selection, `dialog` dumps selected items' tags into stderr or `--output-fd`, as usual. By default, all tags go on a single line separated by spaces and quoted as necessary. To make parsing easier, we tell `dialog` to dump tags one per line with `--separate-output`. You can use our good old friend `read` (sans `-e` this time) to learn the user's selection.

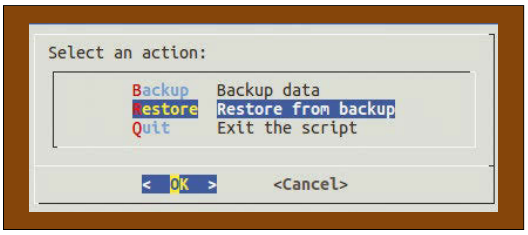
```
while read tag; do
    # Do something
done <"$choice"
```

Make Them Dynamic

Until now, we were reading what `dialog` says. It's time to make data flow the opposite direction. You send it to `dialog` via a pipe. This facilitates dynamic dialog contents, and there are several widgets that can benefit from it. Here, I'll cover two that I like most: You can find the rest in the man pages [3].

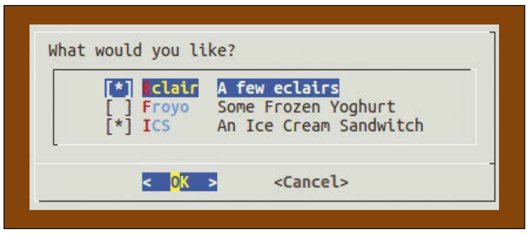
Suppose your script starts a time-consuming operation, such as creating a tarball from a large directory. It's a good idea to give the user some visual feedback of the execution's progress. As you know, `tar -v` prints filenames as it processes them. Why not reuse this handy feature?

The `--programbox` widget makes it a breeze. It reads lines of data from the standard input (unless you redefine this with `--input-fd`) and shows them in a dialog, scrolling as necessary. When the command finishes, the OK button appears so a user can



```
Select an action:

        Backup   Backup data
        Restore  Restore from backup
        Quit     Exit the script

        <  OK  >       <Cancel>
```

Figure 3: The `--menu` widget in `dialog` is well-suited for choosing one of the mutually exclusive options.

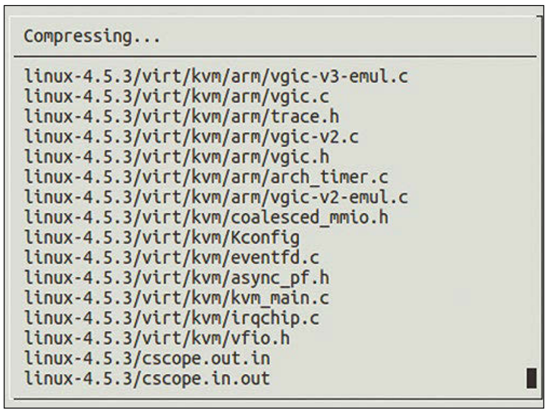


```
What would you like?

      [*] Eclair  A few eclairs
      [ ] Froyo   Some Frozen Yoghurt
      [*] ICS     An Ice Cream Sandwich

        <  OK  >       <Cancel>
```

Figure 4: Checklists are like menus that let you choose more than one option.

Listing 5: A Web Download Script

```
wget -q --progress=dot --show-progress $url 2>&1 |
    sed -un '/%/s,^.\+ \([0-9]\+\)% .\+$,\1,p' |
    dialog --gauge "Downloading..." 7 50
```



```
Compressing...
linux-4.5.3/virt/kvm/arm/vgic-v3-emul.c
linux-4.5.3/virt/kvm/arm/vgic.c
linux-4.5.3/virt/kvm/arm/trace.h
linux-4.5.3/virt/kvm/arm/vgic-v2.c
linux-4.5.3/virt/kvm/arm/vgic.h
linux-4.5.3/virt/kvm/arm/arch_timer.c
linux-4.5.3/virt/kvm/arm/vgic-v2-emul.c
linux-4.5.3/virt/kvm/coalesced_mmio.h
linux-4.5.3/virt/kvm/Kconfig
linux-4.5.3/virt/kvm/eventfd.c
linux-4.5.3/virt/kvm/async_pf.h
linux-4.5.3/virt/kvm/kvm_main.c
linux-4.5.3/virt/kvm/irqchip.c
linux-4.5.3/virt/kvm/vfio.h
linux-4.5.3/cscope.out.in
linux-4.5.3/cscope.in.out
```

Figure 5: The `--programbox` widget is a way to display the program's output in a neat dialog box.

review the output. Unfortunately, there is no support for scrolling at this point, which makes `dialog --programbox` a little less useful than it could be.

Try this:

```
tar czvf /dev/null smth/big | ↩
  dialog --progressbox "Compressing..." 20 70
```

You'll see file names scrolling by quickly (as `/dev/null` is "fast and web-scale," you know), and the end result should look similar to Figure 5.

This sort of feedback is good, but it doesn't tell you how far the operation has already progressed. In other words, you might want percentages. `dialog` handles this as well via `--gauge`. It works in a similar fashion: You pipe integer values between 0 and 100, and `dialog` displays them along with a meter bar. There is also a way to change the text displayed on the widget in run time, so you can switch between "Installing files..." and "Finalizing the setup..." if you wish (again, the man pages [3] have more information).

This is not something we need for the next example, however. Calculating percentages could be tricky (and `dialog` doesn't help you there), so we'd opt for something more straightforward – a web download script (Figure 6). Tools like `wget` already do the math for us, and we can readily pump numbers into `dialog`. It is as easy as the pipeline shown in Listing 5.

The first line invokes `wget`. The `-q` option tells it we just want a progress bar, and two other switches force a particular progress bar appearance. The `wget` command prints progress information to stderr, and we redirect it to a pipe.

Then comes `sed`, which is barely readable (as usual). I recommend turning off buffering with `-u`. This is needed so we get percentages one per line, one at the time. With buffering on, `sed` may send them in large batches, and the progress bar won't go smoothly. The `-n` option tells `sed` not to print any line unless explicitly requested. Then I look for lines having a percent sign embedded, delete everything but a number before the sign, and print the line. Note that this won't work for those HTTP servers that do not report file sizes; in which case, `wget` won't report any percentages.

Next, the values are piped into `dialog`. All options to `--gauge` should be self-explanatory. The only nuance is that you can specify the initial percentage via the fourth option.

The `dialog` tool (see also the "Dialog, the GUI Way" box) may not be as sophisticated as Qt or GTK+. On the other hand, you probably don't want to drive that complex interfaces from a shell script either. Keep `dialog` in mind if you are making multipurpose reusable shell scripts, which will interface with a human. Clicking on menus is much easier than remembering command-line options, even if you can do the latter. ■■■

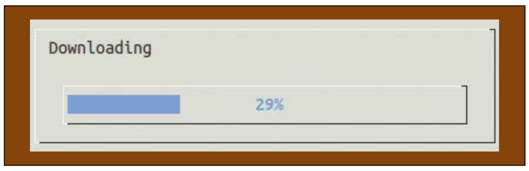


Figure 6: Progress bars are great indicators of how far a long-running operation has progressed.

Dialog, the GUI Way

All examples in this article were terminal-based, because `dialog` itself is a ncurses-based application. However, this doesn't mean you can't create graphical user interfaces in shell scripts. In fact, a few options are available.

For starters, there are Xdialog and gdialog. Both are a decade old yet retain a good degree of compatibility with `dialog` at the command-line syntax level. You are unlikely to find both in your package manager, so let's turn to newer alternatives.

If you are in the KDE camp, have a look at KDialog. It provides the same basic set of widgets as `dialog`, but the syntax is somewhat different. As this is a graphical, KDE-based application, it relies on a higher-level IPC mechanism: D-Bus. This is how you set progress values, for example. User choices are returned via exit codes or stdout.

Finally, there is Zenity. It's a Gnome project, and you should take a look at it if you need GTK+ dialogs. The command-line syntax is again different and pipes are used for IPC as in plain `dialog`. Among the tools we discussed, Zenity is the only one that can provide a notification bar icon. On a similar note, none of these programs require you to specify dialog geometry, like `dialog` does. Layout managers in GTK+ and Qt do a great job here.

Info

[1] builtins(7) man page: *http://man7.org/linux/man-pages/man1/bash.1.html*

[2] bash(1) man page: *http://man7.org/linux/man-pages/man1/bash.1.html*

[3] dialog(1) man page: *https://linux.die.net/man/1/dialog*

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

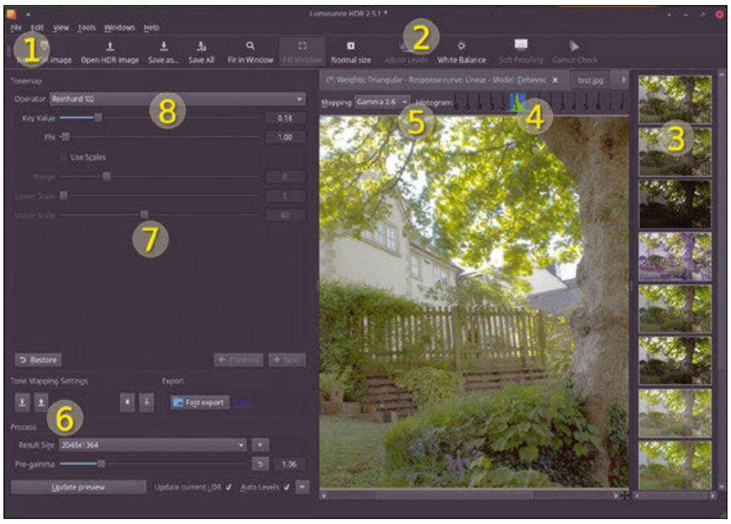
Photo processor

Luminance HDR 2.5.0

Over the past few years, the once specialized effect of HDR-style photography has become mainstream. From estate agents using HDR to add refinement to their interior images, to the latest movies and games using HDR to add impossible levels of detail – even smartphones casually include pseudo-HDR modes with their cameras. And yet, generating real HDR output is still a relatively complex and convoluted process. You need a camera that supports exposure bracketing, and ideally, RAW output: The open source Magic Lantern firmware for many Canon EOS camera adds this facility. You then take a set of photos at various exposures, from underexposed to overexposed, usually from a tripod because of the shutter speed. Software first aligns the images and then combines them into a special kind of data set that combines these exposure values alongside the standard details. Finally, additional clever algorithms are used to apply tone maps for the new composite image and to generate the final output.

My favorite image processing tool, Darktable, recently added the ability to stack and combine images with different exposures to produce excellent HDR output. It's the easiest way I've seen to generate this effect, and it's wonderful being able to create HDR from the same application you use to change exposure and shadow brightness. Darktable, however, doesn't offer in-depth control over any of the separate steps, such as alignment or how tone maps are processed. That's why this major update to Luminance HDR is so significant. Version 2.5.0 comes more than three years after 2.4.0, spanning the huge growth in HDR photography. It's possible that this growth is partly thanks to Luminance being open source, as it has had a major part in making previously costly HDR processing so popular. Luminance is a one-stop tool that will load a stack of RAW images, calculate the alignment even when the camera has moved, and let you apply all kinds of tone maps before rendering the final output. And, it does this without relinquishing user control. It can be difficult and unpredictable, but it's also capable of producing the best results.

The latest version still starts with the image import wizard, which steps you through adding the bracketed image set as well as alignment, if required. You can change the image exposure and create common profiles for setups you want to recreate. You're then dropped into the main application window, which now works well with high DPI screens and even has a dark theme for us vampires. Most significantly, there are two new tone maps: ferradans and mai. Tone maps are intrinsic to the way HDR works and looks, because they map the potential colors in the photo stack to what you see on screen. As such, they define the character of the output. The two new tone maps are excellent. Mai doesn't have a single additional operator and yet generates excellent output, and ferradans is almost as easy to use. These tools are just the tip of the iceberg. The release has many more new features and everything feels a lot quicker than earlier versions, keeping Luminance HDR at the top of its game.



1 Import wizard: Easily stack and align RAW images. **2 Levels and balance:** No need to use an external editor. **3 Thumbnail previews:** Quick-view the output from adjustments. **4 Histogram:** Drag and resize the range used in the image. **5 Preview:** Previews can take a while to render, but their size is adjustable. **6 Tone mapping:** Save tone map presets and update the preview. **7 Tone map options:** Each tone map offers a different set of options. **8 Tone maps:** Luminance HDR now supports 11 different tone maps.

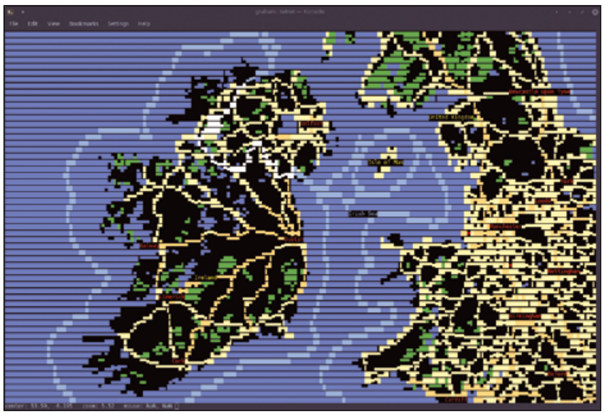
Project Website
http://qtpfsgui.sourceforge.net/

Console maps

mapscii

It always feels a bit of a cheat covering something written with Node.js, because the typical Node.js application often feels like a web application, or even a website. That's because Node.js helps you run things locally written in JavaScript that would typically run on a server on the web. But JavaScript itself has become so popular, perhaps akin to being the BASIC of the web, that using JavaScript with a runtime environment like Node.js is no different to a run-time environment for many other languages, and you get the ability to run your application both offline and online almost for free. How's that for convergence!

This is exactly what's happened with mapscii. It's a brilliant little map tool you can run on the command line via Node.js, and because it's written in JavaScript, you can also do something crazy – connect to an instance of mapscii running on the developer's own telnet server. Yes, telnet. It's a little like something from the *War Games* movie, except the world view zooms into a street view, thanks to OpenStreetMap. It's remarkably quick and responsive, and going from an Earth-sized scale down to street level takes just seconds. Cursor keys with A and Z to zoom are used to control the map, but you can also use the mouse. The mouse wheel can even be used to zoom, and click-dragging the tiles on the background lets you move around the map just as you would a map in a browser. Even if that sounds like a gimmick when you have your ordinary web browser open, mapscii has another trick: Pressing B will remap the output to the braille unicode font, and you can run the whole thing offline or run your own server and access it from your own devices. Perfect for fast low-bandwidth private navigation.


It may seem like a hack, but mapscii is genuinely useful. You can press B to enable Braille mode, for example.

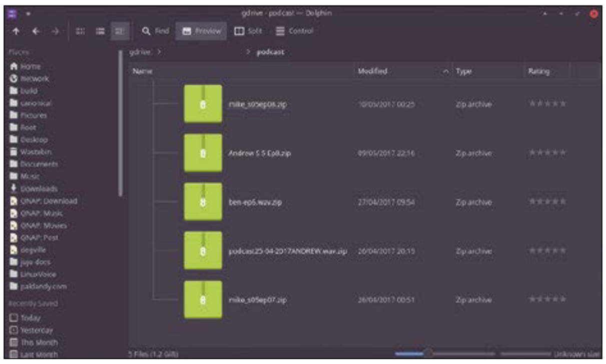
Project Website
https://github.com/rastapasta/mapscii

File sharing

Google Drive for Plasma

Although many of us have moved away from relying on public cloud storage for all our files, switching to alternatives like Nextcloud, few of us have given up the habit completely. This is because services like Dropbox and Google Drive are genuinely useful. They allow you to share large files, collaborate on document editing, and easily search through the content of files and your editing history. Used with careful consideration of your own privacy, as with cloud-based email, cloud file storage is a useful addition in the connected world. And yet, in the case of Google Drive, a native Linux client has never been forthcoming.

As a KDE user, this makes Google Drive slightly more inconvenient, especially when trying to avoid opening a web browser. And, access to Google Drive's files through a web browser is never ideal. It promotes the idea that you never have to manage your files, never have to delete them. A native-Linux client has always been badly needed. Fortunately, there's now Google Drive integration with KDE's Plasma. It's so well implemented that there's very little to see and not very much to write about. It works as a KIO slave, which means when you use the URL `gdrive://` in any KDE application, your Google Drive will be accessed as the destination. The first time you do this, you'll even be stepped through the authentication process, although it can be accomplished from the system-wide accounts page, too, which is also where you can disable the facility. With authentication added, you'll see your Google Drive files just as you would your local files, and they can be opened and saved in the same way. Unfortunately, this doesn't work offline, but it does make copying online files to your local storage trivially easy.


Even when Nextcloud is used for personal files, access to Google Drive is still a convenient upgrade.

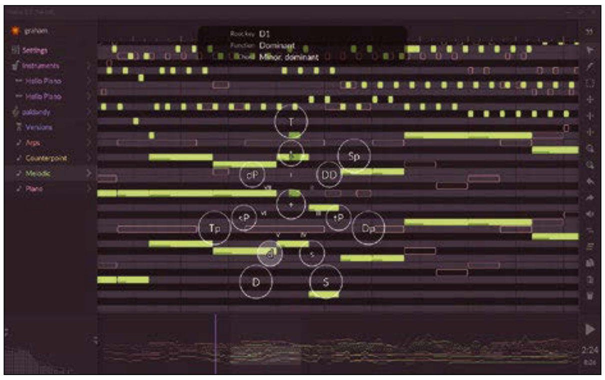
Project Website
https://eang.it/google-drive-integration-in-plasma/

Music sequencer

Helio Workstation 1.7

The complexity of music making software seems to have followed Moore's Law, as if the developers behind those applications can't just leave those extra transistors sitting idle. Ardour, BitWig, and even Rosegarden pile on the features, hoping to help moribund composers find their muse by clicking on extra buttons. Those extra functions and buttons can be incredibly useful and shouldn't be replaced. But, like the effectiveness of a simple melody, there's room for something that focuses more on the simple act of creating notes, much like a writer might use a distraction-free text editor to write words. And that's what Helio Workstation does so well. It's a beautifully designed application that puts note creation, and the relationship between those notes, at the heart of its functionality. And it does this in a fluid, dynamic way that helps you create music.

Helio Workstation is fundamentally a simple MIDI sequencer. The main view is a piano roll, with notes on the vertical axis and duration on the horizontal. Click the mouse to add a node, and press Space to start playback. But there's far more that is immediately obvious. Notes are added to a grouped and colored layer, and you can create new layers easily and use them for different music parts. Layers are transparent, which means you can easily create counter melodies or chords against notes in other layers. Double-click and you get a brilliant radial menu that lets you choose and hear different chords and notes transposed from the root note, and selecting one will add this to the current layer. There's even a built-in piano sound that is ideal for quick compositions. Best of all, version control is a fundamental part of the workflow. Much like a coding project managed with Git, you can check out branches (revisions) and make changes, with everything being tracked for easy reference. It's simply brilliant.


Helio Workstation has one of the best user interfaces I've seen, complete with OpenGL acceleration if enabled.

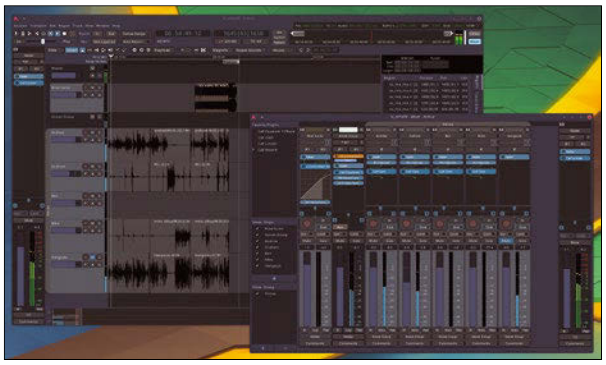
Project Website
https://helioworkstation.com/

Audio workstation

Ardour 5.9

In contrast to Helio Workstation (see above), Ardour is of course the behemoth of digital audio mastering and recording for Linux, Mac, and Windows. It's open source, but it's an amazing application that's worth spending money on. Ardour development is driven almost entirely by such payments, and even a minimal outlay will give you access to an entire generation of upgrades (e.g., every release of Ardour 5.x). This will in turn help the Ardour community. This release, version 5.9, takes us very close to the 6.0 update, and potentially lots of lovely new features, which is why supporting Ardour is so important. As with previous versions, however, this doesn't mean point releases like this are insignificant.

Ardour 5.9 represents the best of this major release cycle. It's full of small fixes, including sensible shortcuts for gain grouping, better color choices, and an updated theme. But it's also full of small additions. There's new MIDI controller support, more OSC commands, better VST plugin support, and parameter mapping. This version also works well without launching Jack. This is important because not everybody has a setup that requires the complexity of Jack, and it's often a hindrance to quickly launching Ardour for a quick recording or edit – something you want when inspiration strikes (or you're late for the podcast). We use Ardour to record, edit, and master our Linux Voice podcasts, and it's supremely capable, more so than many expensive DAW applications for other operating systems. In particular, you can side-chain an effect within a channel strip and actually see the results in the compression effect. This is brilliant and powerful, and although it's difficult to set up and initially understand, it gives you the ultimate control over your recordings.


Paul Davis, the principle developer behind Ardour (and Jack!), uses contributions to the project as his main source of income.

Project Website
https://ardour.org

Image mangler

Audio Shop

When it comes to explaining various aspects of audio, I've always found it easier to talk in terms of images rather than sound. This is obviously easier for people to visualize but it's also because many of the concepts that apply to images and image processing have analogous concepts and processes in the realm of audio. And it's not that they're both fundamentally about waves. Sample rate, for example, is the same as image resolution. The sample depth – 16- or 24-bit, for example – is analogous to the bit depth of an image, whether that's 8, 16, 24, or 32 bits. Low-resolution samples are the same as low-resolution images. Analog to digital converters often perform the same function, and the examples go on.

Robert Foss's Audio Shop scripts are based on this same idea, only they put that idea into practice. Using ffmpeg, ImageMagick, and SoX, Audio Shop lets you mangle image data with SoX's audio processing tools, so you can not only actually see what kind of effect they have on an image but also process images in weird and unusual ways. All the script is really doing is convincing ffmpeg to change the format of the image data into one that could potentially be audio, processing that data as audio, before then asking ffmpeg to save the data into a PNG image file. If each color channel in an RGB image uses 8 bits, this is processed as 8-bit unsigned audio data. There are 12 effects to choose from, including bass and pitch effects, phaser and flange effects, echo, and loudness. Each effect is taken from SoX and takes the same parameters, and the output is similar to what you may see if you've ever experienced synesthesia.

LINUX VOICE
Christmas 2013
PGP
Stop President Obama from reading your emails
SCRATCH
Teach your kids to program, then take early retirement
SQUID
Control the interwebs for your own evil ends
114 PAGES OF NEURAL ENHANCEMENT!
THE BEST FREE
fedora

Process images as if they were audio with Audio Shop.

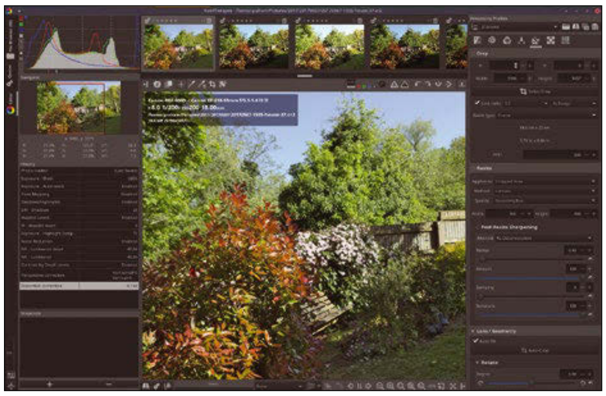
Project Website
https://github.com/robertfoss/audio_shop/

RAW photo processing

RawTherapee 5.1

Earlier, while covering Luminance, I described how Darktable is my favorite open source RAW photo-processing tool. This is true, but it's only by a small margin. That's because RawTherapee gets very close indeed and even has the better UI. RawTherapee has a much more effective light table view for previewing thumbnails, loading hundreds of images quickly, and letting you quickly scan and categorize your collection. Processing is also easier to understand, with the GUI elements easier to see and control. The visual cues, such as the icons for each filter and filter-type, look better and are easier to understand than the Darktable equivalents, and it also helps that every change is added to your photo history in RawTherapee, unlike Darktable. Darktable's advantage is my slight preference for its output, but that's purely subjective.

RawTherapee has seen huge updates recently; 5.0 was released at the beginning of the year and now 5.1 packs in even more features. Version 5.0 massively increased performance and added Wavelets and Retinex tools for adding extra detail to an image – especially useful with close crops and zooms. The big emphasis for the 5.1 release is the inclusion of Pentax Pixel Shift support. This is a range of cameras that moves the sensor by a single pixel while taking a small set of photos, allowing for less noise and better color accuracy, but the set of photos need to be specifically blended for the increase in quality to be realized. There's now a command-line tool, too. This makes running an image process much quicker and easier to automate as the entire GUI application doesn't need to be loaded. As always, plenty of new formats and cameras are supported. It's an amazing application and both the 5.0 update from January and this significant upgrade are great. Even if you're already set on your photo tool of choice, you need to give RawTherapee a try.



If you're new to RAW processing, RawTherapee is slightly easier to use than the equally powerful Darktable.

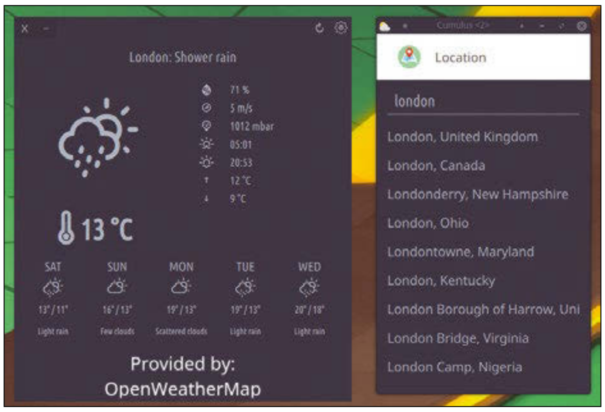
Project Website
http://www.rawtherapee.com/

Weather applet

Cumulus 3.0.2

In the UK, the weather is particularly dynamic, sometimes going through several seasons in a single day. This may explain why we're all obsessed with reading weather reports and watching weather broadcasts. You always need to know whether the weather will change or stay the same. Linux and its various desktops aren't short of tools that give some kind of weather forecast. Because these tools are often simple to write, they were some of the first applications you could download for your desktop, and they're a prerequisite for panel widgets. But it's always nice to see a new tool that's well designed, like Cumulus.

Using either OpenWeatherMap or Yahoo for its data, Cumulus will take a location and render a five-day forecast simply into a different window. The icons and style are excellent, with monochrome images used for the different weather types plus a descriptive line on what the weather is going to do, and the whole window package scales perfectly, whether you leave it as a small square on the screen or a larger window on the background. Additional details include the current weather, temperature, humidity, pressure, wind speed, and sunrise/sunset times, which are all genuinely useful. The tray icon can be enabled from the settings, which will show you the current temperature as a simple panel applet. As you might expect, you can also change the units used and the colors. It's simple, but you don't need any more controls.



A weather application is essential if you're planning a camping trip in England.

One unusual feature is that it can run from an installer that will install the executable binary locally for a single user. This may help if you're running on a machine without root access or want to limit the liability that something may go wrong.

Project Website
https://github.com/vadrian89/cumulus-qt

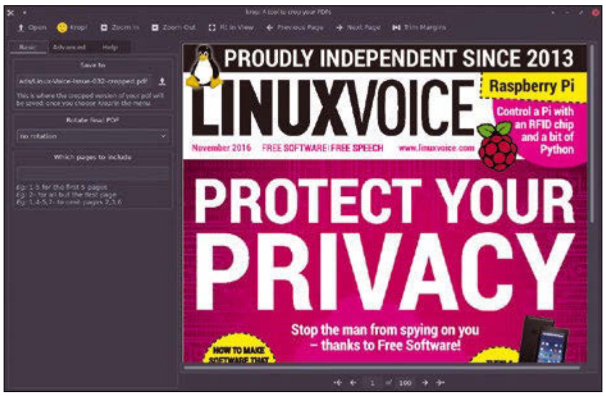
PDF Viewer

Krop 0.4.12

PDF has become the dominant format for lots of offline reading content, as well as some online content unfortunately. It's the only format that works on a screen as well as on a hard copy when needed. But if you do a lot of PDF reading, one aspect becomes very annoying – you always have to readjust, zoom, and recenter your display to best fit whatever document you're looking at. Even then, you often have to manually scroll around a page to see a full line of text or down to the very bottom to see enough of the text. And, your settings are then never saved with the PDF, which means you need to perform the same actions next time.

Krop is a small ingenious little app that can really help in these situations as well as in more complex examples, such as when a single page of a PDF contains what would be on multiple physical pages. With Krop, you load up a PDF and use the selection tool to highlight an area of a single page. It could be the text without a margin, or only the top half of a document. If it is just the margins you want to remove, a button will perform the same step with a single click. With areas selected, a click on the smiley Krop! icon will generate a new PDF file that only contains the highlighted area. Even for complex documents with many

> ...a click on the smiley Krop! icon will generate a new PDF file that only contains the highlighted area.



Krop uses Qt4 but still works well on Qt5. (Thanks to einonm for this find!)

pages – such as an issue of Linux Voice – this only takes a few moments. You can then read or archive the new more usable version. Krop is also capable of more advanced operation. You can create multiple regions, for instance, and each will be cropped into their own files. And you can select all pages, odd/even pages, or just single pages. It works brilliantly.

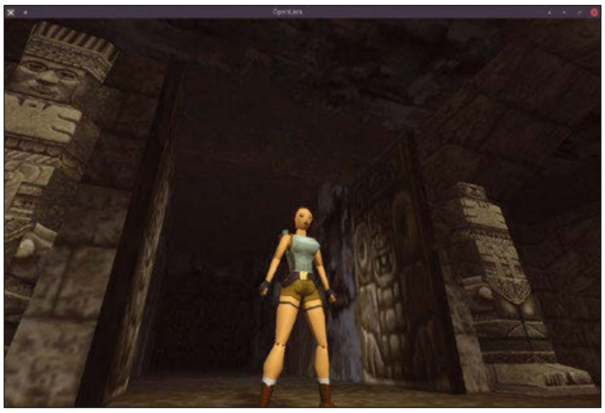
Project Website
http://arminstraub.com/software/krop

Tomb Raider engine

OpenLara

When I first heard about OpenLara, I initially thought it was going to be a recreation of a wonderful and underrated cricket game for the PlayStation 1, Brian Lara Cricket (which was better than its original Amiga forebear thanks to the use of 3D graphics). Even if you didn't like cricket, Brian Lara's action and strategy gameplay was compelling enough to make anyone a fan. OpenLara isn't a reference to Brian Lara. Instead, it's a reference to the much more famous Lara – Lara Croft, from the game series Tomb Raider. The original Tomb Raider games were fantastic. They combined a powerful game engine, capable of letting you explore caves, cities, and the London Underground, while the player solved complex problems and interacted with the environment. The game engine, combined with wonderful scripting and storytelling, was responsible for making those early games so successful. Everyone who's played them remembers the Tyrannosaurus Rex appearing out of the Lost Valley in the first game.

OpenLara is an open source reimplementation of that classic game engine. OpenLara itself has taken inspiration from another open source project, the OpenTomb project, which developed a Tomb Raider-compatible game engine from scratch, which is remarkable as neither of these projects use code from the original studios, Eidos or Core. This was partly to sate the appetite of a Tomb Raider community desperate for the source code, and partly as an academic exercise. OpenLara's party trick is that it can use WebGL to run the game in a modern web browser, which is a great reminder of how far technology has come, but you can also run it directly from your desktop. It's worth playing with not just because it's an interesting project and the source code could help games and level designers, but because it's already advanced and capable enough to make the demo level enjoyable as a free game.


Make Tomb Raider great again by recreating the original game engine.

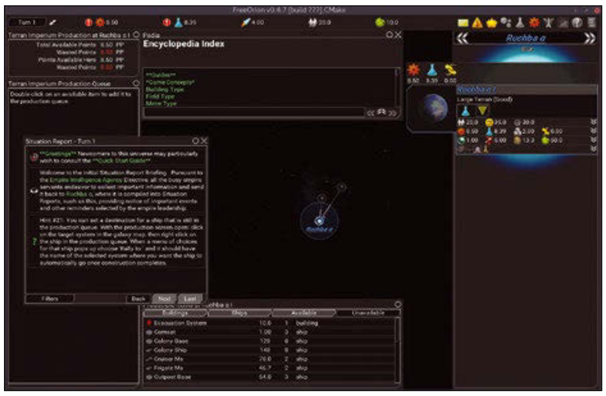
Project Website
https://github.com/XProger/OpenLara

Strategy game

FreeOrion 0.4.7

Linux may not have Elite Dangerous, but we do have several turn-based, space empire, domination type games that potentially have more depth, and FreeOrion could be the best. It's based on a game called Master of Orion, originally released in the early 1990s by Microprose for the PC. Although it's based on similar ideas, FreeOrion is not a clone or remake; it's just similar in gameplay and visual style. It has also been in development for a long time, which means the complexity and design is a little more advanced than other open source titles, even if the game is still incomplete. Small things, like the rotating planets or the parallax star fields, make you feel like you're playing a game that the developers have put their passion into, which is reason enough to give it a go.

If you're a new player, getting into the game can be difficult. You're definitely going to have a better time if you've played this type of game before, because there's a lot of complexity behind every element and decision. Both the in-game help and the wiki will help if you put in the effort. Essentially, you deploy your initial race and build your empire from the home system. You have control over all of this in the beginning when you create a game, balancing things like system times and other attributes against risks. You can develop industry or technology, using research to improve technologies. In this way, you can do things like unlock defensive or offensive capabilities. You start with different fleets for attacking, colonization, and exploration into adjacent unknown systems, all the time using a credit system built on turns to control the speed of the game, along with the hunt for resources and opportunities for colonization. You build up your power and plan things like an invasion against an alien fleet or valuable system, or simply defend your colony against a monster attack.



FreeOrion is great fun as a single player playing against the AI, but nothing beats playing against real(ish) humans.

Project Website
http://www.freeorion.org/index.php/Main

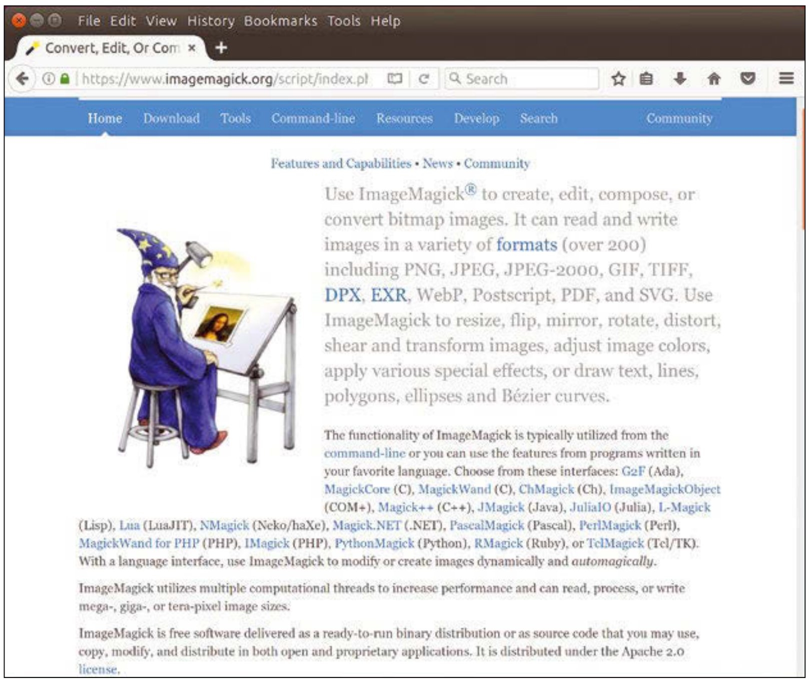
Protect Your Privacy with ImageMagick

Ben Everard wants to keep his location open, but only vaguely.

BY BEN EVERARD

The ImageMagick tool lets you do almost everything you would want to with an image. You can tweak, poke, prod, and otherwise meddle with pixels (Figure 1). In this tutorial, we'll look at how to use this tool to form a command that can help protect our privacy online. This is also the perfect excuse to see how we can use the Linux terminal to build up a complex, powerful command from many smaller commands.

When you take pictures on a digital camera, a whole lot of information is stored along with the actual image. The exact information varies from camera to camera but might include things like the camera settings and location at which the picture was taken. This information is all useful until you start sharing the images online, at which point it can be used to trace exactly where you've been. Many tools will just remove the location data, but we want to be a little more clever. We want to keep the general area the picture was taken but remove the precision. We want a command that embeds the location to the nearest town. Rather than put this information in hard-to-read metadata, we want the info displayed in the image itself as a caption at the bottom. This can be done as shown in Listing 1.



File Edit View History Bookmarks Tools Help
Convert, Edit, Or Com ×
https://www.imagemagick.org/script/index.pl

Home Download Tools Command-line Resources Develop Search Community

Features and Capabilities • News • Community

Use ImageMagick® to create, edit, compose, or convert bitmap images. It can read and write images in a variety of formats (over 200) including PNG, JPEG, JPEG-2000, GIF, TIFF, DPX, EXR, WebP, Postscript, PDF, and SVG. Use ImageMagick to resize, flip, mirror, rotate, distort, shear and transform images, adjust image colors, apply various special effects, or draw text, lines, polygons, ellipses and Bézier curves.

The functionality of ImageMagick is typically utilized from the command-line or you can use the features from programs written in your favorite language. Choose from these interfaces: G2F (Ada), MagickCore (C), MagickWand (C), ChMagick (Ch), ImageMagickObject (COM+), Magick++ (C++), JMagick (Java), JuliaIO (Julia), L-Magick (Lisp), Lua (LuaJIT), NMagick (Neko/haXe), Magick.NET (.NET), PascalMagick (Pascal), PerlMagick (Perl), MagickWand for PHP (PHP), IMagick (PHP), PythonMagick (Python), RMagick (Ruby), or TclMagick (Tcl/TK). With a language interface, use ImageMagick to modify or create images dynamically and *automagically*.

ImageMagick utilizes multiple computational threads to increase performance and can read, process, or write mega-, giga-, or tera-pixel image sizes.

ImageMagick is free software delivered as a ready-to-run binary distribution or as source code that you may use, copy, modify, and distribute in both open and proprietary applications. It is distributed under the Apache 2.0 license.

Figure 1: If you ever need to interact with images from the command line, chances are that there's an ImageMagick tool for the job. Refer to ImageMagick.org to find out.

Obviously, this command is a little complex, but don't worry. It's made up of lots of smaller bits that are all easy to understand. Let's break it down and look at it bit by bit. The two basic building blocks of this command are the pipe (`|`) and command substitution (`$()`). The pipe sends the output of one command to another. In some cases, we use this to send text from one command to the next; in other cases, we're sending images. Command substitution runs the command inside the brackets and inserts the text that the command outputs in the place of the dollar sign.

Let's first look at the command substitutions.

```
$(identify -format '%w' IMG_20170504_191133.jpg)
```

This uses the `identify` command (part of ImageMagick, so you might need to install this package via your package manager) to extract the width of the image. To test this command substitution, run it without the first dollar sign or opening and closing brackets. Obviously, you'll need to change the image name to an image you have.

The second command substitution is a little more complex (Listing 2).

This itself contains a command substitution:

```
$(exiftool IMG_20170504_191133.jpg ↩
  -n | grep 'GPS Pos' | cut -d ':' -f2 | cut ↩
  -f2,3 -d" " | tr " " ",")
```

`exiftool` gets the metadata out of the image (we could use `identify`, but `exiftool` is better for formatting the GPS data in the way we need). The metadata is then piped to `grep`, which discards all of it other than the data that matches "GPS Pos" – the latitude and longitude. A pair of `cut` commands are used to first get everything after the colon, and then get the second and third fields (when separated by a space). This gives us the latitude and longitude with a space between them. But, as you'll see in a minute, we need them separated by a comma, so the final part of this subcommand is `tr`, which just replaces every space (and there is only one) with a comma.

Now, let's go back to the outer command substitution (Listing 3).

As you can see, we've put `1.0000,1.0000` where the previous command placed the latitude and longitude of the picture. The first part of this pipe uses the Google Maps API to get the closest address for this latitude and longitude in JSON format. By default, you're limited to 2,500 daily requests from the API, so if you have a lot of images, you may need to spread them out over several days.

We then use the `jq` command (which you may need to install via your package manager) to get the data we want out of this. The format for `jq` is a language in itself, but what we do is get the first item in the results list, and then loop through the `address_components` list inside this item. If the `address_components` item has a type of country or town, output the `long_name`.

This gives us the town and country on different lines (Figure 2). We would like them comma-separated, so use `tr` to first delete the quote marks, and then replace the newlines with commas. The final `sed` removes the last character, which is a comma.

Let's now look at the full command with this data put in:

```
convert -size 680x100 -background ➘
  '#00000080' -fill white caption:➘
  "Bristol,United Kingdom" miff:- | ➘
  composite -gravity south -geometry ➘
  +0+3 - IMG_20170504_191133.jpg ➘
  annoimg.jpg
```

What was once a matted knot of commands is now reduced to two. Both are commands from ImageMagick: `convert` and `composite`. The first makes an image that's the same width as the image we're editing (from the first command substitution) and a hundred pixels high that has a black slightly transparent background and the text of the town where the picture was taken.

The command saves this new image in MIFF format (ImageMagick's own format) and sends this data to stdout where it's picked up by the pipe and sent to the `next` command. `composite`, as you might expect, takes two images and combines them. The `-gravity` and `-geometry` options tell `composite` to put the text horizontally at the bottom of the other image. The hyphen tells the command to take input from stdout, and the final two options are the original image and the newly modified image to save the output as.

This one massive command does all the modifying. The only thing left is to strip the metadata out of the image, which can be done with:

```
convert annoimage.jpg -strip strippedimg.jpg
```

Here ends our whistle-stop tour of ImageMagick and the Bash tools for composing commands. We haven't had time to look into all the options, but you should now have an idea of how you can put together several simple tools to build complex commands in a way that's almost endlessly flexible. Here, we've built a command that we've hand-coded for the specific image that we want to edit. As an extension, you could put this into a script that took an option for the image and performed this action on it or a command that iterated through a collection of images updating each one as it went. ■■■

Listing 1: Embed Location

```
convert -size $(identify-format '%w'
   IMG_20170504_191133.jpg)x100 -background '#00000080'
-fill white caption:"$(curl "http://maps.googleapis.com/maps/api/
   geocode/json?latlng=$(exiftool IMG_20170504_191133.jpg -n |
grep 'GPS Pos' | cut -d ':' -f2 | cut -f2,3 -d" " | tr " " ",")" -s |
   jq '.results[0] .address_components[] |
select( .types[0] | contains("country") or  contains("town"))
   .long_name' | tr -d '"' | tr  '\n'  ',' | sed 's/.$//')" miff:- |
composite -gravity south-geometry +0+3 -  IMG_20170504_191133.jpg
   annoimg.jpg
```

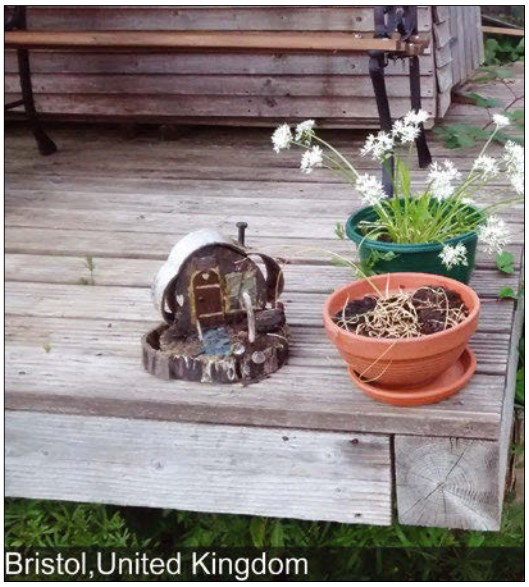
Listing 2: Second Command Substitution

```
$(curl "http://maps.googleapis.com/maps/api/geocode/
   json?latlng=$(exiftool IMG_20170504_191133.jpg -n |
grep 'GPS Pos' | cut -d ':' -f2 | cut -f2,3 -d" " | tr " " ",")" -s |
   jq '.results[0] .address_components[] |
select( .types[0] | contains("country") or  contains("town"))
   .long_name' | tr -d '"' | tr  '\n'  ',' | sed 's/.$//')
```

Listing 3: Outer Command Substitution

```
$(curl "http://maps.googleapis.com/maps/api/geocode/
   json?latlng=1.0000,1.0000" -s | jq '.results[0] .address_components[] |
select( .types[0] | contains("country") or  contains("town"))
   .long_name' | tr -d '"' | tr '\n' ',' | sed 's/.$//')
```

Figure 2: The caption in the finished image means we'll never forget where we took the picture, and we can be confident that we can share it online without telling everyone where we live.



Markdown: One Format to Rule Them All

Create attractive and structured documents from the comfort of your text editor – and convert them to a huge array of formats.

BY MIKE SAUNDERS

It should come as no surprise that we Linux Voicers are big fans of open standards. For complicated documents or spreadsheets, Open Document Format (ODF, as used by LibreOffice) is the way to go. But, it's also a rather complicated beast, and for shorter or simpler texts that you may want to process using other tools, it's arguably overkill. So, what other options do you have for text-heavy content?

Well, there's HTML – which is somewhat standardized but gets a bit fiddly to write with all the tags. If you're working on a scientific paper, then LaTeX is a great choice, but it has a pretty steep learning curve. And, of course, there's always plain ASCII text, but that has its limitations as well – namely, there's no way to add formatting.

Wouldn't it be great if you could add some symbols and other bits to plain text, so that it's perfectly readable in an editor like Vim, Emacs, or Nano, but could also be processed to add formatting? Enter Markdown [1]. The name is a play on markup, as in a "markup language" like HTML, but Markdown is very different. It was originally created in 2004, and today there are various implementations and supersets (with no official standard).

Regardless of the implementation, the core features of Markdown are the same – and it's used in many places, including GitHub (look for files with `.md` extensions). With Markdown, you can create structured plain text files in any editor and process them to prettier HTML (or other formats) when necessary. In that sense, it's the perfect middle ground between raw ASCII text and word processor formats. In this article, we'll show how to write, edit, and process Markdown docs, all from the comfort of your favorite text editor. (See the "Dedicated Markdown Editors" box for more information.)

Dedicated Markdown Editors

Although any plain text editor can be used for Markdown, some custom editors are worth checking out as well. Remarkable [2] is a polished editor that lets you edit syntax-highlighted Markdown text in a pane on the left-hand side, while on the right, you can see a constantly updating preview of the HTML equivalent. Then there's Dillinger [3], which you can test immediately in your browser, along with StackEdit [4], which does a similar job.

If you use Vim, you can get some Markdown integration via PlasticBoy's Markdown Vim Mode [5], whereas Emacs fans have their own package thanks to Jason Blevins [6].

Getting Started

The first thing you'll need to do is install Markdown itself, which converts plain text files into prettier HTML. Search for it in your distro's package manager (most distros have it available). If you're on an Ubuntu-based distro, you can install it from the command line like so:

```
sudo apt install markdown
```

If you can't find it anywhere in your distro, you can get it from the project's website [1]. Extract the `.zip` file, `cd` into the resulting directory, and use `./Markdown.pl` in place of "markdown" for the rest of the commands in this tutorial.

Now, let's try some Markdown! Using your favorite text editor, create a file called `test.md` with the following content:

```
My _first_ file in **Markdown**!
```

Just by looking at that, can you guess what the formatting is? The underscores surrounding "first" suggest underlining, right? The double asterisks around "Markdown" look like they're trying to make the word stand out, yes? If you follow some plain-text mailing lists, you may have seen people using these pseudo-formatting options before – especially on the Linux kernel mailing list, where Linus Torvalds uses them a lot.

Anyway, let's convert this plain text into HTML. At the command line:

```
markdown test.md > test.html
```

This simply uses the Markdown tool to process the contents of our `test.md` file and redirect the output to a `test.html` file. Now open that HTML file in a web browser, and voilà – you'll see the results, like in Figure 1, which also shows the HTML source that is generated.

Now, you'll notice right away that the HTML formatting is slightly different to what you may have expected from the Markdown version. Specifically, the underscores haven't made the word "first" underlined, but rather in italics. This is because Markdown converts the underscores to `<em>` tags – for emphasis – which the browser then chooses to interpret as italics.

So, you already know how to create italics and bold with plain text. But what about combining them together? Try this:

```
Check out this **_word_**
```

Run the previous command to generate the HTML, and view it again in your browser – this time "word" is in both bold and italics. It's possible to use Markdown formatting across multiple words, like so:

```
_This sentence is in italics._
**And this one is bold!**
```

When you look at the HTML version of this, you'll notice that both sentences have been combined into the same line. Markdown is very strict about paragraph formatting; to make it clear that you want the lines to be in separate paragraphs, put a blank line between them:

```
_This sentence is in italics._

**And this one is bold!**
```

If you look at the HTML version of this, you'll see that each line is surrounded by its own `<p>` tags now.

Heads Up

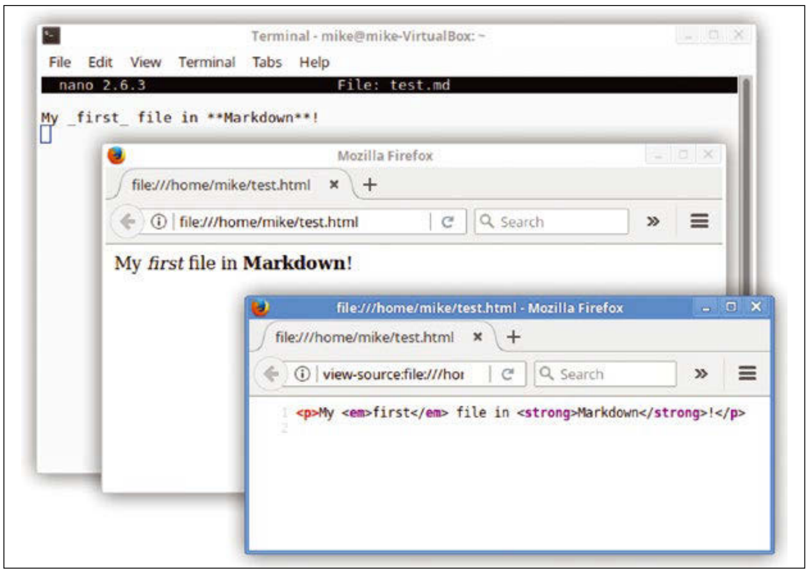
Now that we have some basic formatting sorted out, let's turn to structure. In Markdown, you can create headings of different sizes by prefixing words with hash marks (`#`). Try this, for example:

```
# A big heading

A normal paragraph.
```

Note that you don't strictly need the space between the hash mark and the content, but when you're writing Markdown content, always bear in mind usability. Even if your ultimate goal is to make an HTML version of your document, the whole point of Markdown is that it should still be easy to read in plain text.

For smaller headings, add more hash marks. There are six heading sizes in total.

```
# Title of doc

Introductory paragraph.

## Section header

In this section we will...

### Subsection

By the way, you should...
```


Terminal - mike@mike-VirtualBox: ~

```
nano 2.6.3                    File: test.md

My _first_ file in **Markdown**!
```

Mozilla Firefox — file:///home/mike/test.html

My *first* file in **Markdown**!

file:///home/mike/test.html - Mozilla Firefox — view-source:file:///ho

```
<p>My <em>first</em> file in <strong>Markdown</strong>!</p>
```

Figure 1: Our first Markdown document, showing how it's rendered in Firefox (and the HTML source).

Convert this to HTML, and you'll see the results like in Figure 2. Always keep in mind that the headings aren't just used for presentation purposes; they add structure to a document as well. As I'll show later in the tutorial, you can convert Markdown documents into many other formats, where structure is often especially important.

Creating bulleted lists is easy – just prefix the items with asterisk (*) marks like so:

```
* One thing
* Another
* And one more
```

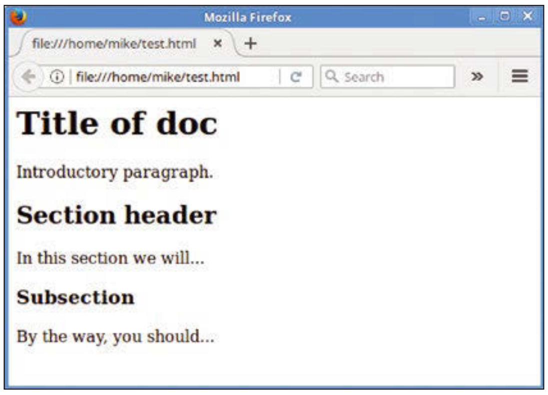
What about numbered lists? These are simple too:

```
1. One thing
2. Another
3. And one more
```

Note that you can put lists inside other lists; to do this, adjust the indentation by adding an extra space for the sublist. Here's an example:

```
* Outer list
 * Inner list
 * Inner list
* Outer list
```

For yet another level of indentation, use two spaces at the start instead of one. You can, of course, use the other formatting options we've covered beforehand, like bold and italic.


Mozilla Firefox — file:///home/mike/test.html

Title of doc

Introductory paragraph.

Section header

In this section we will...

Subsection

By the way, you should...

Figure 2: Structure is important in Markdown documents – there are different headers to denote sections and subsections.

If you're writing a document and want to quote something from another source (or just make a particular paragraph really stand out, you can use block quotes. These are marked by greater-than (>) signs and work like this:

```
Normal paragraph

> Blockquoted para.
This is also blockquoted

Normal paragraph
```

Note how the line beginning with "This is also" is included in the block quote – because it follows immediately after the line beginning with >. An empty line is required to end the block quote and return to the normal paragraph style.

If you're citing something in a block quote, then you may want to provide a link to the website as well. These look a bit tricky in Markdown, but with a bit of practice you'll soon get used to them. Links are made of two parts, like in a regular HTML document: the text that's displayed for the link and the actual address to which it points.

Say we want to create a link with the text "Linux Magazine" that points to the website at *http://www.linux-magazine.com*. We put the text inside square brackets, and the address inside round brackets:

```
[Linux Magazine](http://www.linux-magazine.com)
```

This might look a bit ugly as raw text, but it gets the job done and when you generate an HTML version, you just see the link text (so just *Linux Magazine*, which will be clickable).

Suppose you're writing a long document that contains multiple links to a specific web page. Later, you need to change all of those links to point to something else. You could do a find-and-replace job in your editor for this purpose, but Markdown offers a more elegant solution in the form of reference links. These let you define a link at the end of the document, with a name, and then you can reference that name throughout the text. Here's an example:

```
You should go to [the LM website!][our-site]

...

[Visit our website][our-site] for more info.

[our-site]: http://www.linux-magazine.com
```

In this document, both "the LM website" and "Visit our website" links point to the same place, as defined by the "our-site" reference at the end. So, if we wanted to change that link to something else, we only have to edit one line and not go through the entire document. Pretty handy, right? Note that the reference link itself doesn't appear in the actual HTML that's generated.

A Picture's Worth a Thousand Words

What about images? Obviously, these are not easy to integrate into plain-text documents (unless you want to do very funky things with ASCII art), so Markdown's solution involves an exclamation point (!), alt text, and the address of the image, like so:

```
# NetBSD rules!

It really does. Just check out its logo:

![NetBSD logo](http://www.netbsd.org/images/↩
NetBSD-smaller.png)
```

Here, the alt text is provided in square brackets – this is the tooltip text that appears in the HTML version, when you hover over the image. Using alt text on the web is good practice, as it provides extra information for search engines and visually impaired users. It's good to have useful text in Markdown as well, keeping in mind that the plain-text version should be just as useful (or close enough) as the HTML equivalent.

We then have the address of the image in rounded brackets, and the results can be seen in Figure 3.

If you want to include code in your documents and have it rendered in a monospace font in the HTML version, you can surround the code with three back tick (`) characters like so:

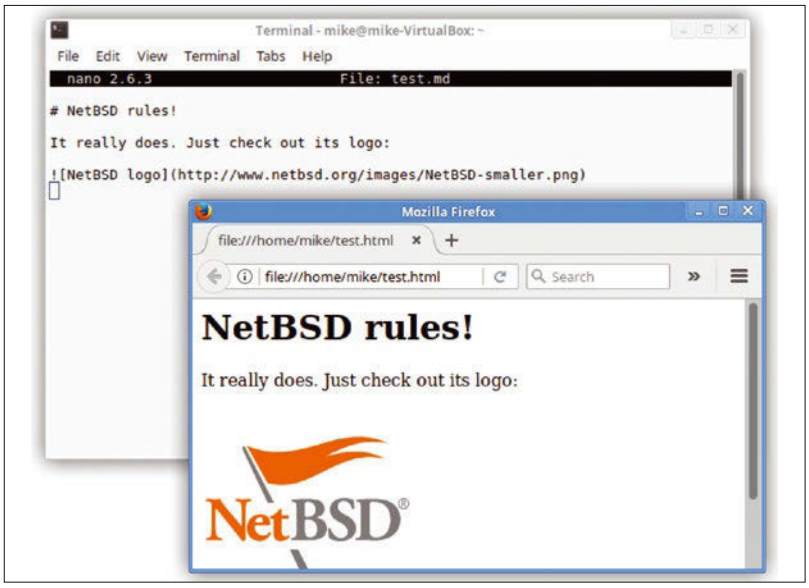


Figure 3: Want to include an image in the converted document? That's easy, too.

```
nano 2.6.3                    File: test.md

# NetBSD rules!

It really does. Just check out its logo:

![NetBSD logo](http://www.netbsd.org/images/NetBSD-smaller.png)
```

Terminal - mike@mike-VirtualBox: ~
File Edit View Terminal Tabs Help
Mozilla Firefox
file:///home/mike/test.html
Search
NetBSD rules!
It really does. Just check out its logo:
NetBSD

~~~
Some Python code:

```x = 1
print(x)```
~~~

As your documents get longer, it's a good idea to break them up using horizontal lines. To add these in Markdown, and get `<hr>` tags in the resulting HTML, use three or more dashes:

```
This is the end of a section.

---------

Now, onto something else...
```

Three dashes suffice, but I like to use more so that they're wider in the Markdown text and emphasize that they're splitting up the document.

Finally, in some circumstances, you may want to use Markdown symbols on their own, without Markdown poking its nose in and trying to convert them into something else. To do this, use backslashes before the symbol – for instance, here's how to generate the word awesome, including the asterisks, without Markdown making it bold in the HTML version:

```
\*\*awesome\*\*
```

You'll probably never need this feature, but it's worth bearing in mind, especially if you ever end up writing a Markdown document about... writing Markdown documents!

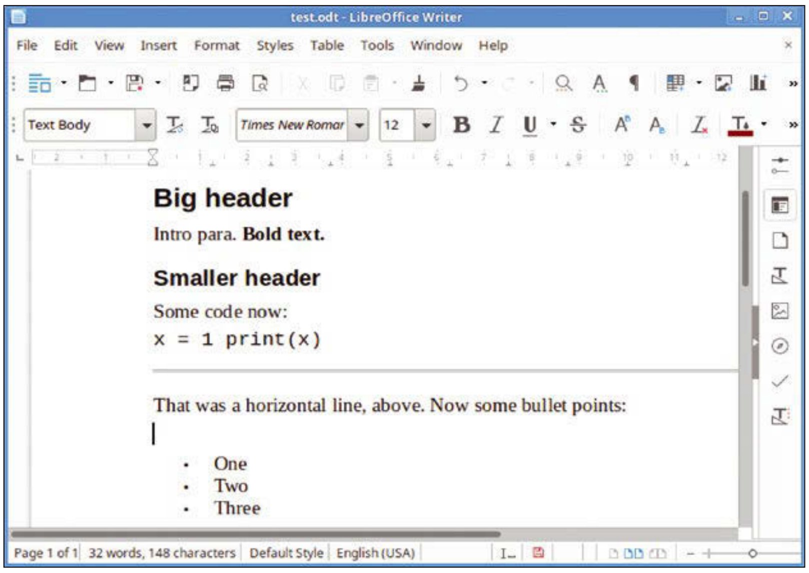
Advanced Usage

So far, we've been converting our work to HTML, because that's the only output format supported by the main Markdown tool. But there's another app we can use to convert to a much wider range of formats called Pandoc [7]. This is an immensely versatile program – available in the package repositories of most Linux distros – and once you have it installed, you can see how many formats it supports by running:

```
pandoc --help
```

That shows quite a bit of output, but you can scroll up to see the input and output formats. For exchanging documents with word processor users, the two most useful formats are `.docx` (Microsoft Word) and `.odt` (OpenDocument text, as used by LibreOffice – the better choice in terms of fully open standards).

So, create a Markdown document containing various bits of formatting from the previous examples in the tutorial. Add some headers, emphasis, paragraphs, bulleted lists, and horizontal lines. Save as `test.md` as usual, and then run this command:


test.odt - LibreOffice Writer
File Edit View Insert Format Styles Table Tools Window Help
Text Body
Times New Roman
12
Page 1 of 1 | 32 words, 148 characters | Default Style | English (USA)

Big header

Intro para. **Bold text.**

Smaller header

Some code now:

`x = 1 print(x)`

That was a horizontal line, above. Now some bullet points:

- One
- Two
- Three

Figure 4: Thanks to Pandoc, you can convert your Markdown text into a huge range of formats, including OpenDocument.

```
pandoc test.md -t odt -o test.odt
```

This tells Pandoc to use the `.odt` format and send the results to a file called `test.odt`. Open that file in LibreOffice, and you should see all the formatting from your Markdown document, implemented neatly in the word processor (Figure 4). Helpfully, the conversion uses OpenDocument paragraph styles for easier modification of the text – it's not just built around hard-coded sizes and formatting.

You may find Markdown so comfortable and useful that you want to use it all the time. If you live inside Emacs or Vim, you may want to make it your default format for long text documents – in which case, you'll want to convert other formats to Markdown before editing. Pandoc can do this as well. To test that, create a new LibreOffice Writer document called `test2.odt` and put some content inside it. Mix up the formatting a bit (paragraph styles, bullet points, emphasis, etc.) and save it. Then, at the command line, run:

```
pandoc test2.odt -t markdown -o test2.md
```

Now look at the contents of `test2.md`, and you should see some familiar Markdown code. (Note that you may see some slight differences, such as dashes being used for bullet points instead of asterisks, but it should largely be the same.) Of course, LibreOffice Writer can create documents way too advanced for replication in Markdown, so you may lose some formatting and need to tidy things up, but generally it works well.

As mentioned previously, Markdown is being adopted by more and more sites for quick publishing jobs, so here you've learned some valuable skills. And, best of all, you can get more done without leaving your trusty text editor! ■■■

Info

[1] Markdown: *https://daringfireball.net/projects/markdown/*

[2] Remarkable: *https://remarkableapp.github.io*

[3] Dillinger: *http://dillinger.io*

[4] StackEdit: *https://stackedit.io*

[5] Markdown Vim Mode: *https://github.com/plasticboy/vim-markdown*

[6] Markdown Emacs package: *http://jblevins.org/projects/markdown-mode/*

[7] Pandoc: *http://pandoc.org*

FEATURED EVENTS

Users, developers, and vendors meet at Linux events around the world. We at *Linux Magazine* are proud to sponsor the Featured Events shown here.

For other events near you, check our extensive events calendar online at *http://linux-magazine.com/events.*

If you know of another Linux event you would like us to add to our calendar, please send a message with all the details to *events@linux-magazine.com*.

SUSECON 2017

Date: September 25–29, 2017

Location: Prague, Czech Republic

Website: *http://www.susecon.com*

SUSECON 2017 features exceptional technical content presented by SUSE employees, customers, partners, and community enthusiasts. SUSECON focuses on helping you find ways to build and define your future to provide a flexible and efficient infrastructure. The dynamic and entertaining keynote speeches inspire, inform, and invigorate.

JAX London

Date: October 9–12, 2017

Location: London, England

Website: *https://jaxlondon.com/*

JAX London is a four-day conference for cutting-edge software engineers and enterprise-level professionals. JAX brings together leading innovators in the fields of Java, microservices, continuous delivery, and DevOps. Tracks include Emerging Technologies, Java Core & Languages, Agile & Communication, Big Data & Machine Learning, and more!

Linux Kernel Summit

Date: October 24–27, 2017

Location: Prague, Czech Republic

Website: *http://events.linuxfoundation.org/events/linux-kernel-summit*

The annual Linux Kernel Summit brings together core kernel developers to discuss the state of the existing kernel and plan the next development cycle. New in 2017 are four days of sessions and workshops opened to a larger group of developers, along with the half-day, invitation-only Maintainer Summit.

EVENTS

InterDrone	September 6–8	Las Vegas, Nevada	http://www.interdrone.com/
WiSTEM 2017	September 11–12	San Francisco, California	http://www.womeninstemconference.com/
Storage Developer Conference (SDC)	September 11–14	Santa Clara, California	http://www.snia.org/events/storage-developer
Kieler Open Source and Linux Conf.	September 15–16	Kiel, Germany	http://kilux.de/
14th Annual 2017 HPC for Wall Street, Cloud, AI & Data Centers	September 18	New York, New York	http://www.flaggmgmt.com/hpc/
OSBConf	September 25-26	Cologne, Germany	http://osbconf.org/
SUSECON 2017	September 25–29	Prague, Czech Republic	http://www.susecon.com/
data2day	September 26–28	Heidelberg, Germany	https://www.data2day.de/
CopyCamp 2017	September 27–28	Warschau, Poland	https://copycamp.pl/en/
Internet Security Days	September 28–29	Brühl, Germany	https://isd.eco.de/
JAX London	October 9–12	London, England	https://jaxlondon.com/
it-sa	October 10–12	Nürnberg, German	https://www.it-sa.de/
Heise Cloud Conference	October 17	Cologne, Germany	https://www.heise-events.de/cloudkonf
All Systems Go!	October 21–22	Berlin, Germany	https://all-systems-go.io/
All Things Open	October 23–24	Raleigh, North Carolina	https://allthingsopen.org/
Open Source Summit Europe	October 23–25	Prague, Czech Republic	http://events.linuxfoundation.org/events/open-source-summit-europe
WebTech Conference	October 23–27	Munich, Germany	https://webtechcon.de/
heise devSec	October 24–26	Heidelberg, Germany	https://www.heise-devsec.de/
EclipseCon Europe	October 24–26	Ludwigsburg, Germany	https://www.eclipsecon.org/europe2017/
Linux Kernel Summit	October 24–27	Prague, Czech Republic	http://events.linuxfoundation.org/

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.*

AUTHORS

Erik Bärwaldt	18, 56
Swapnil Bhartiya	8, 16
Zack Brown	12
Bruce Byfield	28, 32, 64
Joe Casad	3
Mark Crutch	69
Ben Everard	69, 78, 90
Dr. Rolf Freitag	38
Andrew Gregory	71
Karsten Günther	42
Jon "maddog" Hall	76
Charly Kühnast	36
Martin Loschwitz	22
Vincent Mealing	69
Graham Morrison	84
Simon Phipps	70
Mike Saunders	72, 92
Mike Schilli	60
Valentine Sinitsyn	80
Ferdinand Thommes	48, 52

CONTACT INFO

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Managing Editor
Rita L Sooby, rsooby@linux-magazine.com

Localization & Translation
Ian Travis

News Editor
Swapnil Bhartiya

Copy Editors
Amber Ankerholz, Amy Pettle

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen, Lori White

Cover Image
© bluebay and William Mahnken, 123RF.com

Advertising – North America
Ann Jesse, ajesse@linuxnewmedia.com
phone +1 785 841 8834

Advertising – Europe
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 99 34 11 48

Publisher
Brian Osborn, bosborn@linuxnewmedia.com

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
616 Kentucky St.
Lawrence, KS 66044 USA

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)
Fax: 1-785-856-3084

For all other countries:
Email: subs@linux-magazine.com
Phone: +49 89 99 34 11 67

www.linuxpromagazine.com – North America

www.linux-magazine.com – Worldwide

Printed in Germany

Distributed by COMAG Specialist, Tavistock Road, West Drayton, Middlesex, UB7 7QE, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 616 Kentucky St., Lawrence, KS, 66044, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 616 Kentucky St., Lawrence, KS 66044, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany.

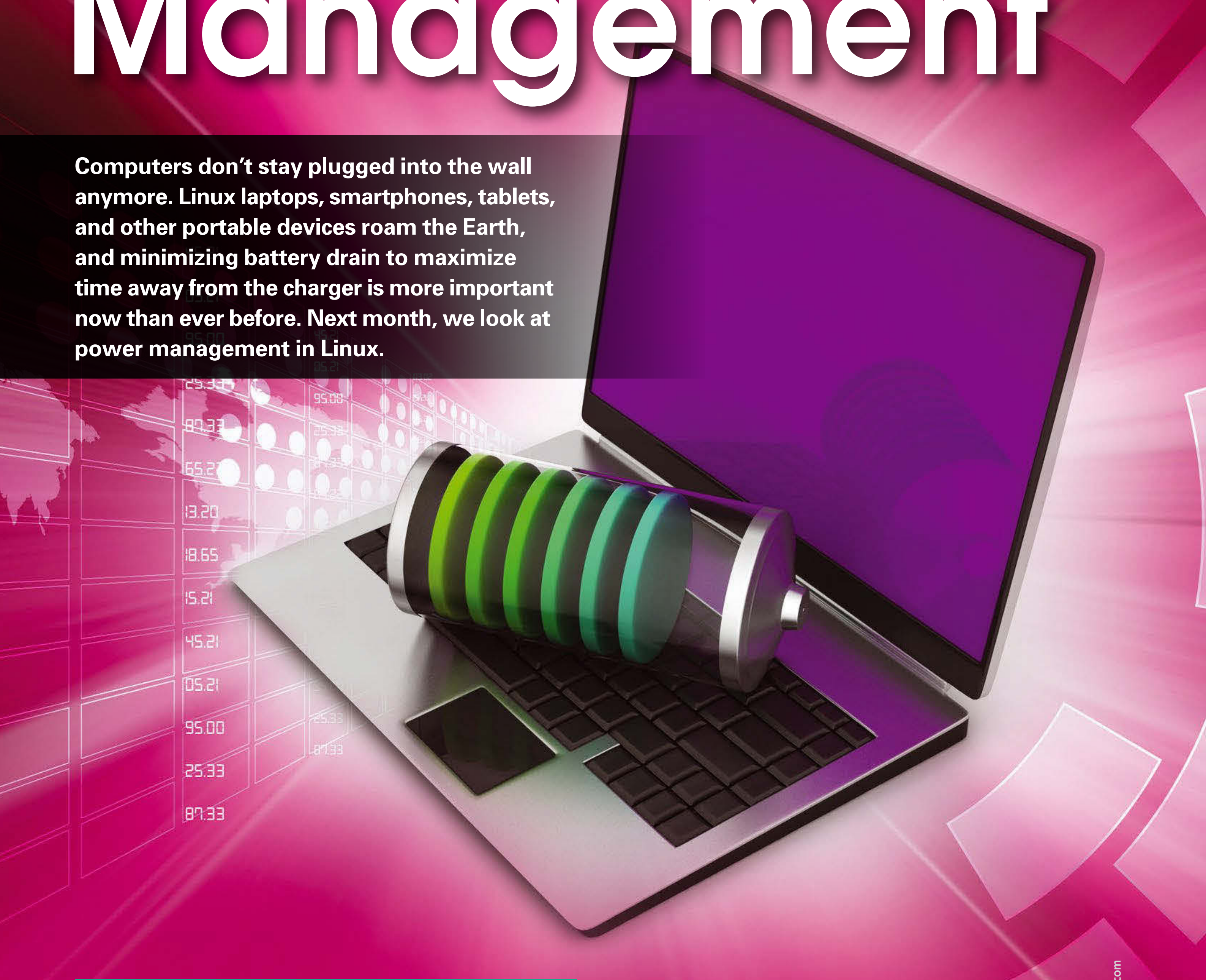
Issue 202 / September 2017

Power Management

Approximate

UK / Europe	Aug 05
USA / Canada	Sep 01
Australia	Oct 02

On Sale Date



Computers don't stay plugged into the wall anymore. Linux laptops, smartphones, tablets, and other portable devices roam the Earth, and minimizing battery drain to maximize time away from the charger is more important now than ever before. Next month, we look at power management in Linux.

jax® LONDON

Join 1000+ Software Innovators!

VERY EARLY BIRD
Save up to £200 by July 27

Use **LMcode10** to save an extra 10%!

The Conference for Java and Software Innovation

October 9 – 12, 2017 | London

Big & Fast Data

DevOps, Container & Cloud

Continuous Delivery

Agile & Communication

Java Core & Languages

Microservices

Software Architecture & Design

www.jaxlondon.com

SUPERMICRO®

MicroBlade™

Highest-Density • High-Performance • High-Efficiency • Cost-Effective
Enterprise, Data Center, Web Applications, HPC and Cloud Computing Solutions

- Best density and power efficiency with up to 56 UP or 28 DP server nodes
- M.2 NVMe SSD and HW SAS3 RAID support
- Ethernet switches supporting 1G, 2.5G and 10G downlinks with 96% fewer cables
- Redundant 2200W Titanium Level (96% efficiency) digital power supplies
- Battery Backup Power (BBP) modules with integrated power supply and cooling
- Supports Intel® Xeon® Processors E5-2600 v4/v3, E5-1500 v5, D-1500, E3-1200 v5/v4/v3 and Atom®

intel XEON inside™

Intel Inside®. Powerful Productivity Outside.

SUPERMICRO

Learn more at www.supermicro.com