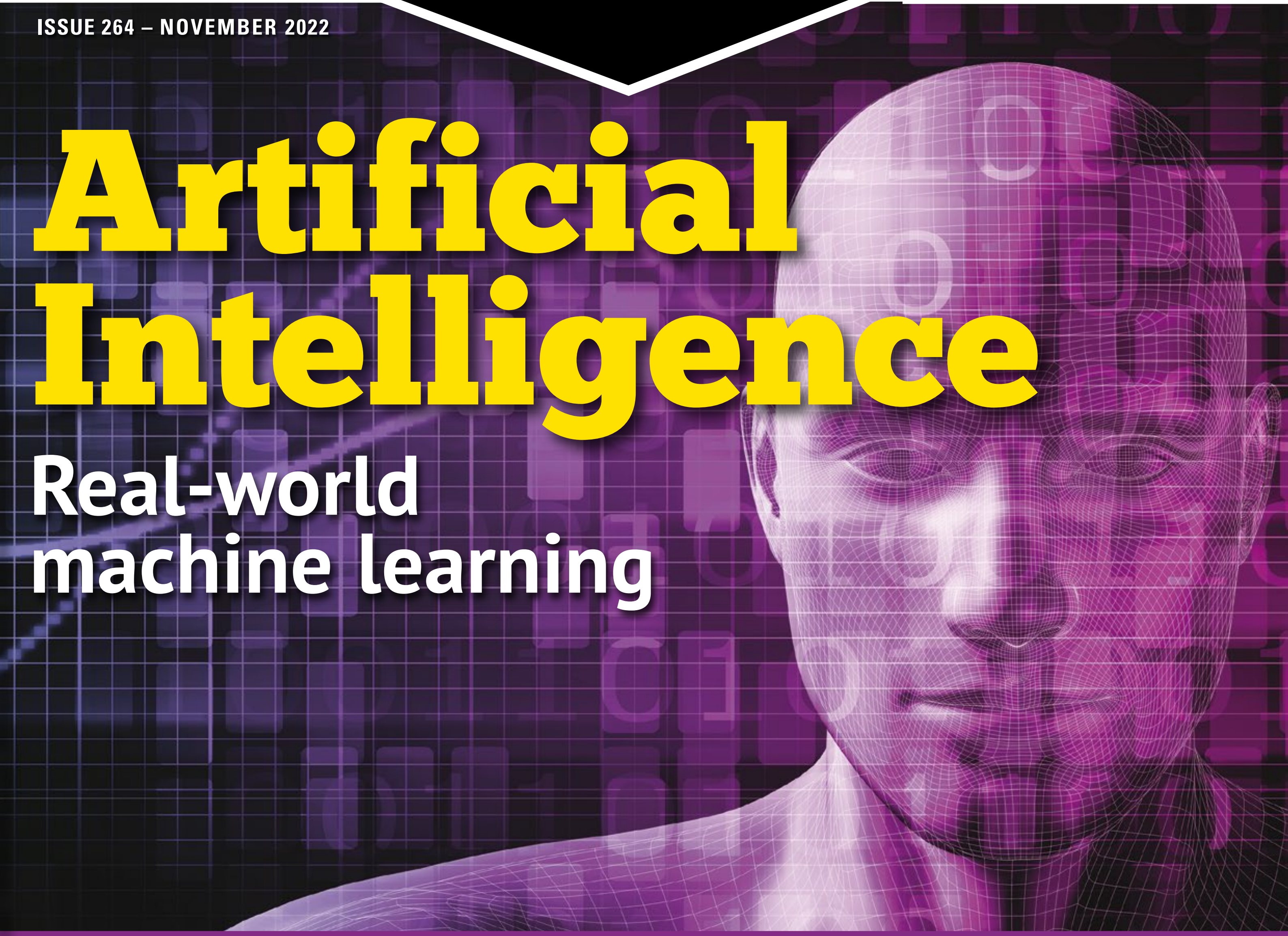
Linux Android Backup
Save smartphone data to a Linux PC

LINUX PRO MAGAZINE

ISSUE 264 – NOVEMBER 2022



Artificial Intelligence
Real-world machine learning

Bluetooth Tricks
Connect your Rasp Pi to a smartphone

SSH Front Ends

Clean Code
Tips for better coding

Ryzom: Adventure awaits on a distant planet

iTunes Metadata: Share music info across your devices

FOSSPICKS
10 FANTASTIC FREE TOOLS

LINUX NEW MEDIA
The Pulse of Open Source

WWW.LINUXPROMAGAZINE.COM

4
TB
Storage Edition

Magnesium chassis
1.6 cm thin | 1.1 kg light

GeForce RTX 3050 Ti
Gaming & content creation

Premium business ultrabook goes workstation

TUXEDO InfinityBook Pro 14 - Gen7

Intel Core i7-12700H
14 Cores | 20 Threads

3K Omnia display
16:10 | 2880 x 1800 Pixel

Max size 99 Wh battery
for up to 16 hours

Magnesium chassis
1.7 cm thin | 1.3 kg light

100%
Linux

5
Year
Warranty

Lifetime
Support

Built in
Germany

German
Privacy

Local
Support

TUXEDO COMPUTERS 18th ANNIVERSARY

tuxedocomputers.com

WILD DOGS

Dear Reader,

It is easy to write columns on the scary and annoying stuff, which seems to happen every day, but it is perhaps more important to point out the positive things when they happen, and the Ethereum Merge looks like a very positive thing.

As I have often mentioned in this space, the tech industry has a way of releasing problems on the world that behave a little like invasive species – stirring up lots of trouble because the indigenous environment doesn't have any natural defenses. The motivational speakers call this "disruption" and warrant that it is a very good thing. "We'll fix any problems created by the new tech with other new tech, which we'll invent later," we tell ourselves. To be fair, a little disruption is sometimes helpful, but when the environment these invasive technologies get released into is our real environment (as in, the place where we live), the limitations are less forgiving. Crypto mining is a good example of a technology we unleashed on our environment without any kind of plan.

As we described back in the May 2022 issue, "Bitcoin mining alone consumes approximately 91 terawatt-hours of electricity per year. A single US home uses approximately 11,000 kilowatt-hours per year, which means that Bitcoin mining consumes as much energy as 8 million homes – more than seven times the total amount used by Google – and that number is growing every year as Bitcoin gains popularity. According to a recent study, if you take the total energy cost of Bitcoin mining divided by the total number of Bitcoin transactions, every Bitcoin purchase has an energy cost of over $100 – even if you're just buying coffee or flagging an Uber." [1]

I mention this quote to provide the context that I believe was missing from the stories in the popular press announcing that the Ethereum crypto community has completed what they called the Merge – their migration to a new and more eco-friendly crypto-mining model. Ethereum started out like Bitcoin, with an incredibly wasteful proof-of-work mining model that was never intended for a large-scale global currency. Proof-of-work is an official-sounding euphemism for a frenzy of massive GPU-laden server farms all over the world simultaneously racing to validate the same crypto block. The proof-of-work model is like a field filled with thousands of wild dogs; when you throw a bone into the middle of that field, all the dogs chase after it.

As far as anyone can tell, Bitcoin, allegedly the child of an absent and mysterious creator, appears to be stuck with the lumbering proof-of-work model, but the Ethereum community has made a concerted effort to solve the energy problem, and they have been laboring for some time to transition to the far-more-efficient proof-of-stake model, which they estimate could save an estimated 99 percent of energy costs. Whereas Bitcoin consumed more energy than Google, the proof-of-stake version of Ethereum will use less than a single music video. According to the Ethereum website, people around the world consumed 45 times more energy watching the "Gangnam Style" video in 2019 than the proof-of-stake Ethereum uses in a year [2].

It is worth pointing out that other eco-friendly cryptocurrencies already exist. (For a list of the most sustainable cryptocurrencies, see the LeafScore website [3].) However, these other sustainable cryptocurrencies tend to operate on a much smaller scale. Ethereum is huge – arguably the second biggest cryptocurrency after Bitcoin, and its more versatile design means that it could easily surpass Bitcoin sometime in the near future.

The governance of something like the Ethereum protocol is quite complicated. Although it is all about money (it *is* money), in many ways, the organization is more like an open source project, with a group of core developers and several other tiers of contributors participating in various ways [4]. Switching the validation model from proof-of-work to proof-of-stake must have been a monumental effort, requiring a lot of people to leave their comfort zone but still stay together as a community. To accomplish the transition, they had to agree on all the details of their proof-of-stake design, launch a separate proof-of-stake blockchain to test and perfect it, and then merge this new chain with the existing proof-of-work blockchain (which is why they call this process *the Merge*).

It all looks good so far, but we're still early in Ethereum's new era. Complications can happen, and they are certainly not out of the woods, but the Ethereum community deserves attention and thanks for helping to tame the wild dogs running loose in the garden.

Joe

Joe Casad,
Editor in Chief

Info

[1] "Gold Rush: Assessing the Environment Impact of Crypto Mining," by Jack Wallen and Joe Casad, *Linux Magazine*, issue 258, May 2022, *https://www.linux-magazine.com/Issues/2022/258/The-Impact-of-Crypto-Mining*

[2] Ethereum energy consumption: *https://ethereum.org/en/energy-consumption/#proof-of-stake-energy*

[3] The 28 Most Sustainable Cryptocurrencies for 2022: *https://www.leafscore.com/blog/the-9-most-sustainable-cryptocurrencies-for-2021/*

[4] Introduction to Ethereum governance: *https://ethereum.org/en/governance/*

LINUX MAGAZINE
NOVEMBER 2022

ON THE COVER

38 Linux Android Backup
Take control of Android backups with this powerful free tool.

54 SSH Front Ends
Avoid mistakes and password problems by using a GUI to manage your remote sessions.

62 Bluetooth Communication
With a little Bluetooth magic, you can connect your smartphone to your Raspberry Pi.

68 Clean Code
These popular programming principles will keep your code cleaner and easier to maintain.

74 Ryzom
Journey to another world with this Steam-powered online role-playing game.

92 iTunes Metadata
This tutorial on sharing iTunes data will help you get started with Python and XML.

NEWS

8 News
- Fedora 37 Beta Available
- Salix 15.0 Available
- Ubuntu 20.04.5 LTS Release Now Available
- MX Linux 21.2 Offers Improvements and Bug Fixes
- Gnome Project Releases New Telemetry Data Collection Tool
- Kubuntu Focus Announces New Desktop Model
- SparkyLinux Now Available

12 Kernel News
- Mysterious Alignments in the Kernel
- Discovery and Invention

COVER STORIES

16 Machine Learning
"I won't make this mistake again," you promise yourself. In other words, you'll learn from experience. Computers can do that, too. We'll introduce you to the fundamental forms of machine learning.

22 AI and Language
Whether through voice assistants, chatbots, or the automatic analysis of documents, rapid developments in AI are helping speech technologies make inroads. But how does AI manage to understand the subtleties of human language?

26 Unsupervised Learning
The most tedious part of supervised machine learning is providing sufficient supervision. If you're short on time and expertise, try unsupervised learning instead.

REVIEW

34 Distro Walk – Rocky Linux
We interview Brian Clemens, Rocky Enterprise Software Foundation project manager.

IN-DEPTH

38 Linux Android Backup
Linux Android Backup offers an open source solution to the problem of backing up your Android devices.

44 Command Line – Alternative Unix Commands
If you are looking for modern display options or more speed at the command line, these alternatives to traditional Unix commands may be just what you need.

48 Programming Snapshot – Go Photo Tagger
In honor of the 25th anniversary of his Programming Snapshot column, Mike Schilli revisits an old problem and solves it with Go instead of Perl.

54 SSH Front Ends
Bring remote management into focus with these graphical front ends for SSH.

Artificial Intelligence

Machine learning remains shrouded in mystery even though it is now an integral part of our everyday world. This month we look behind the curtain at some popular techniques for supervised and unsupervised learning.

MakerSpace

62 Bluetooth Communication
We use a Raspberry Pi, a Pi Pico, and a smartphone to communicate over Bluetooth.

68 Clean Code Principles
Clean code principles can improve the readability of your source code, making life easier for both you and your users.

LINUXVOICE

71 Welcome
This month in Linux Voice.

72 Doghouse – Portability and Costs
Choosing the right server computing solution for your company, and minimizing CPU overhead from the start, can help lower costs and maximize portability over time.

74 Ryzom
Fancy an adventure? Travel to the futuristic planet Atys, where many exciting missions await.

80 Seafile 9
This open source file storage tool is more than a match for Nextcloud and ownCloud.

86 FOSSPicks
This month Graham looks at Furnace, Node-RED 3, Exquisite, corrscope, darktable, Overgrowth, and more!

92 Tutorial – iTunes XML
Read and manipulate data from your iTunes XML to share music metadata between applications and across devices.

TWO TERRIFIC DISTROS
DOUBLE-SIDED DVD!
SEE PAGE 6 FOR DETAILS

95 Back Issues | 96 Events | 97 Call for Papers | 98 Coming Next Month

Debian 11.5 and Rocky Linux 9.0

Two Terrific Distros on a Double-Sided DVD!

Debian 11.5

64-bit

Debian releases only every couple of years, making each release a major event. The release of Debian 11 (Bullseye) in August 2021 was no exception. Not only was Bullseye's Homeworld theme unusually sophisticated for Debian, but the release also featured driverless printer and exFAT filesystem support, as well as reliable Bluetooth integration.

A year later, the 11.5 release is the latest enhancement to this firm foundation. Like all Debian releases, Debian 11.5 consists of bug fixes and security updates that have been issued since the last point release. In addition, the release includes bug fixes for dozens of packages, including dpkg, NVIDIA drivers, and the Debian installer.

Like all Debian releases, Debian 11.5 lacks the newest software, but it compensates by providing stability and security that is second to none. With almost two-thirds of active distributions being based on Debian, even a point release is an influential event in software.

Rocky Linux 9.0

64-bit

Rocky Linux is less than two years old, but it is rapidly becoming a major RPM distribution rivaling Red Hat Enterprise Linux (RHEL) in popularity. To date, Rocky Linux has had two general releases, although their numbering continues on from CentOS, emphasizing its role as a successor.

Rocky Linux is one of the distributions forked from CentOS when CentOS development was stopped in favor of CentOS Stream. As its homepage says, Rocky Linux is designed to be "100% bug-for-bug compatible with Red Hat Enterprise Linux." Both are constructed from the same source code. The main difference is that, while both are popular enterprise solutions, Rocky Linux emphasizes community development in a way that RHEL does not. For many, corporate and home users alike, this is an essential difference.

However, technical differences are slowly starting to emerge. Rocky Linux 9.0 introduces Peridot, a tool for building Rocky Linux from scratch and for easily extending the source code. More differences are likely to emerge in the next few years. Meanwhile, Rocky Linux 9.0 will be supported until 2032.

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

Although this Linux Magazine disc has been tested and is to the best of our knowledge free of malicious software and defects, Linux Magazine cannot be held responsible and is not liable for any disruption, loss, or damage to data and computer systems related to the use of this disc.

Take Your First Steps In Web Development!
Linux Professional Institute
WEB DEVELOPMENT ESSENTIALS
JavaScript
HTML
Node.js
CSS
SQL
NEW

Web Development Essentials

Modern software applications are commonly developed for the Web. Linux Professional Institute‘s new Web Development Essentials program supports you in your first steps in software development. It includes learning materials suitable for both training and self-study. Upon passing the Web Development Essentials exam, you receive a certificate to prove your skills.

LPI Learning Materials are available at learning.lpi.org.
Find out more about the program
and the exam at lpi.org/wde

Linux
Professional
Institute

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

08
- Fedora 37 Beta Available
- Salix 15.0 Available for Installation with Lots of Updates and a More Modern Look

09
- Ubuntu 20.04.5 LTS Release Now Available
- MX Linux 21.2 Offers Improvements and Bug Fixes
- Gnome Project Releases New Telemetry Data Collection Tool
- More Online

10
- Kubuntu Focus Announces New Desktop Model
- A New Release of Lightweight Linux Distribution SparkyLinux Now Available

Fedora 37 Beta Available

The next iteration of the Fedora 37 operating system has finally reached beta and is now available to download and install. This was announced on Fedora Hyperkitty (*https://lists.fedoraproject.org/archives/list/test-announce@lists.fedoraproject.org/message/DX2W2NLQZYFHDAVSUKLI7OSGH3B6CJSL/*) along with the announcement that the final freeze for Fedora 37 would hit Tuesday, October 4.

In terms of features in Fedora 37, you can look for plenty of changes, such as updates to the LLVM 15 compiler, LXQt 1.1, a preview of the new Anaconda Web-based installer, support for Raspberry Pi 4, Linux kernel 5.19 (the latest mainline kernel), Gnome 43 (which includes a revamped Quick Settings), a new GTK4/libadwaita port of files (which also includes the new adaptive sidebar, icon emblems, and rubberband selection), WebExtension API support in Gnome Web, high-resolution scroll wheel support, direct scanout support (to aid multi-monitor setups), server-side decorations (now with essential color support), as well as a reference KVM VM disk image.

For anyone who wants to test the new beta of Fedora 37, the default x86_64 version can be downloaded from the official Fedora download server (*https://dl.fedoraproject.org/pub/alt/stage/37_Beta-1.5/Workstation/x86_64/iso/*). If you'd rather try out one of the spins (such as Cinnamon, KDE Plasma, LXDE, LXQt, Mate, SoaS, Xfce, or i3), they can be downloaded from the Spins directory (*https://dl.fedoraproject.org/pub/alt/stage/37_Beta-1.5/Spins/x86_64/iso/*).

Salix 15.0 Available for Installation with Lots of Updates and a More Modern Look

Salix 15.0 is a Linux distribution that is 100 percent compatible with Slackware and defaults to the Xfce desktop environment. The latest release offers a new look and plenty of updates, including Xfce 4.16, GTK+ 3, Firefox 102 ESR, LibreOffice 7.4, Gimp 2.10, and Whiskermenu as the new default panel menu.

According to the Salix developers, "There have been extensive discussions in our forums with respect to how the new release will look and everything has been revamped, including a new GTK theme, a new icon theme, a new window manager theme, and a default wallpaper …"

Salix now has both dark and light themes to help bolster the new modern look of the UI. Other features include desktop usage optimization, high-quality package repositories, fully localized system admin tools, and support for both 32- and 64-bit architecture.

New users should keep in mind that the installation of Salix is 100 percent text-based (ncurses), so it will take a bit of extra skill to get the operating system installed.

Download and install a copy of Salix from the official download page (*https://www.salixos.org/download.html*) and keep in the know about Salix from the distribution's official blog (*https://blog.salixos.org/*).

Ubuntu 20.04.5 LTS Release Now Available

As you might expect, there's another point release available for the LTS version of Ubuntu. Although this update doesn't include much in the way of new features, it does offer plenty of bug fixes, security patches, app updates, and kernel 5.15.

If you're already using Ubuntu 20.04, you can get the .5 release by way of the built-in upgrade system, which means there's no need to do a fresh install.

One thing to keep in mind is that 20.04 is not the latest LTS release for Canonical's flagship operating system. That title would belong to 22.04. However, because there are still a large number of users still working with 20.04 (and because it is still supported until April 2025), Canonical continues to push upgrades.

That being said, the .5 upgrade does not install the latest, greatest software to 20.04. For example, LibreOffice upgrades to version 6.4.7.1, Gnome is at 3.36.8, and Firefox is at version 104. To get more updated versions of such software titles, you'll need to either run a distribution upgrade (to migrate from 20.04 to 22.04) or install those packages manually.

If you'd rather do a fresh install of Ubuntu 20.04.5, you can get the ISO image from the official download site (*https://releases.ubuntu.com/focal/*) and read the full changelog (*https://wiki.ubuntu.com/FocalFossa/ReleaseNotes/ChangeSummary/20.04.5*) for the new release.

MX Linux 21.2 Offers Improvements and Bug Fixes

The developers of the most popular distribution on Distrowatch, MX Linux, have unleashed the second point release for the distribution. Although there isn't anything groundbreaking with the latest iteration, it does include plenty of bug fixes and various improvements to help improve an already outstanding distribution.

Wildflower, based on Debian Bullseye (11.4), comes with two different options for the kernel. The regular release uses the 5.10 LTS kernel, whereas the Advanced Hardware Support variant makes use of the 5.18 kernel.

Beyond the kernel, MX Linux received bug fixes and improvements for the installer, mx-tweak now allows the disabling of Bluetooth adapters, Fluxbox received a new mxfb-look tool (for saving and restoring theme combination), mx-updater/apt-notifier now has an option to use nala as the backend, a kernel cleanup tool was added to mx-cleanup, mx-boot-options now includes a UEFI management tool, and the disk space checker will help make sure there's enough disk space available before a kernel is updated.

As far as upgrading from the previous point release, the developers stated, "If you are already running MX 21, there is no need to reinstall. Packages are all available through the regular update channel."

Download your copy of MX Linux 21.2 from the official download page (*https://mxlinux.org/download-links/*) with a choice between Xfce, KDE Plasma, and Fluxbox desktop. Make sure to also read the official release notes (*https://mxlinux.org/blog/mx-21-2-wildflower-released/*) for more information.

Gnome Project Releases New Telemetry Data Collection Tool

Although telemetry (the collection of data and measurements for the purpose of monitoring) is a rather touchy subject, especially for those who value their privacy, the Gnome foundation has announced it will be adding a telemetry data collection tool to the desktop environment.

Before anyone panics, this data will be 100 percent anonymous and will be used only to improve the Gnome desktop environment by informing various decisions for the future of the software. Those decisions will help to inform the Gnome Foundation where resources can be best used and how to better help users.

MORE ONLINE

Linux Magazine

www.linux-magazine.com

ADMIN HPC

http://www.admin-magazine.com/HPC/

Parallel I/O Chases Amdahl Away
• Jeff Layton
Amdahl's Law explains why the serial portion of applications doesn't scale in performance as computing resources are added for the same problem size.

Rocky Reaches for a Role in HPC
• Joe Casad
When Red Hat announced that it would stop supporting CentOS as a free Linux distro based on Red Hat Enterprise Linux (RHEL), the whole Linux IT world had to scramble, but nowhere was that crisis more acute than within the HPC community.

ADMIN Online

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

Convert Linux shell commands into PowerShell cmdlets
• Tam Hanna
Convert Linux shell commands into PowerShell cmdlets and modules and run them in a PowerShell environment.

Spotlight on the Kubernetes package manager, Helm
• Martin Loschwitz
A Helm chart is a template of several parts that defines, deploys, and upgrades Kubernetes apps and can be considered the standard package manager in the Kubernetes world.

Package applications in Docker containers
• Ferdinand Thommes
Kaboxer packages applications that are otherwise missing from distribution package sources.

The new tool is a command, `gnome-info-collect`, and will present all collected data to the user before it's transmitted. Once you run the command, you will be asked to okay the transmission of the information and, upon successful transmission, you'll be thanked for helping to improve Gnome.

Types of information that will be collected include such details as operating system, hardware vendor, Flatpack/Flathub enabled, installed apps, file sharing (active/inactive), remote desktop (active/inactive), remote login (active/inactive), number of users, default browser, and enabled extensions. There is zero locale information or anything that could be considered sensitive transmitted to the Gnome developers.

You can install the tool with the help of Snap, DNF, Pacman, or Zypper.

For more information, check out the official GitLab page for the project (*https://gitlab.gnome.org/vstanek/gnome-info-collect*).

Kubuntu Focus Announces New Desktop Model

The Kubuntu Focus NX is a new small form factor desktop computer that ships with Kubuntu 22.04 and the latest KDE Plasma interface. This new machine can be specced with either an i5-1135G7 or i7-1165G7 11th gen Intel CPU, both of which include the Xe iGPU, which can drive up to four displays.

The Kubuntu Focus NX (*https://kfocus.org/spec/spec-nx*) supports up to 64GB of memory, up to 2TB of NVMe storage, and up to 4TB of SATA storage and even comes with a bracket that allows you to mount the device to the back of your monitor.

Other features include 2 x Thunderbolt 3/USB-C with DisplayPort 1.4, 1 x HTMI 2.0b, 1 x Mini DisplayPort 1.4, 3 x USB-A 3.2 Gen2, 1 x RJ-45 Intel i225 V2.5 GbE LAN, 1 x audio 2in1, 1 x Far-Field Quad-Array microphone, 1 x SDXC memory card reader, and 1 x Kensington Lock Mount.

The base cost of the Kubuntu Focux NX is $695 and can be configured up to $2,135 (with a Core i7, 64GB RAM, 1TB primary disk, and a 4GB secondary disk).

Order your Kubuntu Focus NX from the *kfocus.org* site now (*https://kfocus.org/order/order-nx.html*).

A New Release of Lightweight Linux Distribution SparkyLinux Now Available

SparkyLinux 6.4 is here. Based on Debian Bullseye, this lightweight distribution ships with plenty of updates for packages such as Firefox, Thunderbird, VLC, LibreOffice, LXQt, Xfce, Openbox, and KDE Plasma and makes it possible to install a number of those applications from Debian unstable (to enjoy more recent releases). The latest release is updated from both the Debian and Sparky stable repositories (as of August 11, 2022).

Out of the box, you'll find apps such as Firefox to be a bit out of date (with version 91.11.0.0esr installed). Fortunately, those who prefer to use a more recent version of Firefox can install versions 103.0.2 or 102.1.0esr from the Sparky repositories (either firefox-sparky or firefox-esr-sparky).

Users who are running earlier versions of Sparky 6.x can upgrade to the latest version with the command `sparky-upgrade`.

This new release is available for the following architecture:

- amd64: Versions – LXQt, KDE Plasma, Xfce, Openbox, and text mode
- i686: LXQt, Openbox, and text mode
- armhf: Openbox and text mode

To find out more about Sparky 6.4, read the full release notes (*https://sparkylinux.org/sparky-6-4/*) and download a copy of the ISO for installation (*https://sparkylinux.org/download/*).

Get the latest news in your inbox every two weeks

Subscribe FREE to Linux Update

bit.ly/Linux-Update

Get started with

OpenSource
JOB HUB

Find your place
in the open source
ecosystem

OpenSourceJobHub.com

Zack's Kernel News

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

By Zack Brown

Author

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

Mysterious Alignments in the Kernel

Sudip Mukherjee reported a build failure with the mainline kernel source tree. This is fairly unusual. Most patches get tested to see if they'll compile on a variety of different systems before the patch is accepted, but a build failure can happen.

After noticing the failure, Sudip did a `git bisect` – a fast way to jump half the distance to the problem, then half the remaining distance, and so on. It's one of the most common techniques for identifying which patch actually caused a bug. Sudip used it to track the problem down to a small patch that compared the number of bytes used by two different variables that needed to be the same.

Linus Torvalds is often interested in build failures or crashes in the mainline tree because he maintains that tree himself. Linus responded right away to Sudip. He said he had looked at the data size comparison, and it seemed like a good patch to him. He said, "That BUILD_BUG_ON() looks entirely correct, and if that sizeof() doesn't work, then the code doesn't work." In other words, the check was appropriate. Linus added, "I'm not seeing what could go wrong in there."

Linus was not seeing the build failure himself – he was simply examining the bug report. But he remarked, "it would seem to be something very peculiar to your build environment (either that config, or your compiler)."

This is not what someone wants to hear if they are hoping the developers on a given project will help track down their problem. Sudip replied that his build environment had not changed at all since his previous successful build. Not only that, but he encountered the exact same problem on his laptop – presumably a quite different environment.

Linus asked, "Hmm. What compiler do you have? Because it seems very broken."

Linus dug further into the code, looking at the machine language opcodes produced by the compiler, among other things. He speculated, "I suspect some of the structs inside of that 'struct edid' end up getting aligned, despite the '__attribute__((packed))‘. For example, 'struct est_timings' is supposed to be just 3 bytes, and it's at an odd offset too (byte offset 35 in the 'struct edid' if I did the math correctly)."

But he reiterated, "it obviously doesn't happen for me or for most other people, so it's something in your setup. Unusual compiler?"

Sudip was certain it was not his personal setup. He replied, "I am using gcc version 11.3.1 20220517 (GCC). And I am not just building spear3xx_defconfig; I am building all the arm configs with the same compiler in the same setup and only spear3xx_defconfig started failing." In other words, Sudip was using a very normal environment using a standard compiler. He added that he was also compiling under a clean Debian virtual Docker instance – no special circumstances.

Linus continued to dig into the code, and he found that the variable sizing – the thing that seems to have caused the problem for Sudip in the first place – was actually very different from what it needed to be. He didn't know why the discrepancy was there, but he felt that he had put his finger on something that had to be near the solution.

He posted a patch that seemed to fix the problem – but he also said, "I have no idea what it is that triggers this only on spear3xx_defconfig. Some ARM ABI [Application Binary Interface] issue that is triggered by some very particular ARM compiler flag enabled by that config?"

Linus added, "This smells like a compiler bug triggered by 'there's a 16-bit member field in that gtf2 structure, and despite it being packed and aligned to 1, we somehow still align the size to 2'."

Arnd Bergmann thought Linus's patch looked correct. However, Russell King also came into the conversation at this point, with the following explanation:

"It's an age old thing, it's no compiler bug, and it's completely compliant with the C standards. Implementations are permitted by the C standard to pad structures

and unions how they see fit – and some do if it makes sense for performance.

"The mistake is that people forget this detail, and they expect structs and unions to be laid out a certain way – because it doesn't matter to the same extent on x86.

"However, as older ARM CPUs could not do unaligned loads, ensuring that things were naturally aligned made complete sense, even if it meant that people who assume the world is x86 got tripped up – the only way around that would be to make every load very expensive.

"It's not 'align to size of 2' in OABI; it tends to be align to a multiple of 4, because the underlying architecture is 32-bit."

Jani Nikula, the author of the original patch that triggered Sudip's build failure, replied, "Indeed the bug is ancient. I just threw in the BUILD_BUG_ON() on a whim as an extra sanity check when doing pointer arithmetics on struct edid." And he added, "Makes you wonder about all the other packed structs with enum members across the kernel."

Arnd replied with a further explanation:

"It is not the 'enum' that is special here; it's the 'union' having unpacked members, and the same thing happens when you have nested structs: both the inner and the outer aggregate need to be packed, either with __packed at the end, or on each individual member that is not fully aligned to max(sizeof(member), 4)).

"I think in general, most __packed annotations we have in the kernel are completely pointless because they do not change the structure layout on any architecture but instead just make member access slower on architectures that lack unaligned load/store instructions. There have definitely been other cases though where a __packed annotation is not needed on any sane architecture but is needed for OABI ARM.

"Overall I'm not that worried because the only machines running OABI kernels would be on really old hardware that runs a limited set of driver code.

"A completely different matter are the extraneous __packed annotations that lead to possible problems when accessed through a misaligned pointer. We ignore -Waddress-of-packed-member and -Wcast-align in the kernel, so these never get caught at build time, but we have seen bugs from gcc making incorrect assumptions about alignment even on architectures that have unaligned load/store instructions."

Further technical explanations and considerations ensued, until Russell remarked, "Sounds like we need a howto document for people to ignore and continue doing their own thing."

The conversation continued into deep dark recesses, with the main idea being to catch this sort of thing before it could become a problem next time. Following the conversation intently, Linus at one point remarked, "it would probably not be a bad idea to check that packed structures/unions don't have atomic types or locks in them. I _think_ we're all good, but who knows."

The discussion continued, and new people were brought into the CC list, as the ideas touched more and more areas of the kernel. There was no grand conclusion, except that now a lot more developers are aware of a lot more of the details of this particular data alignment issue.

Discovery and Invention

Linus Torvalds discovered open source. The GNU project may have written the General Public License (GPL), but it was never truly useful until Linus showed everyone how to use it. And this is one of the most valuable things to learn from Linux kernel development, because Linus is still discovering open source, and he is still showing the rest of us how to do it.

One interesting aspect of his approach has to do with when he'll lay down the law versus when he'll push hard versus when he'll simply nudge. Not too long ago, he laid down the law regarding patches that would cause build warnings. He wouldn't allow them in patch submissions, and he set the official kernel's build process to fail if warnings occurred.

At around the same time, Linus also started nudging developers to use cryptographic signatures (signed tags) when making pull requests. Signed tags would mean that Linus could confirm that a particular request came from a particular person; thus, he could trace specific pieces of kernel code back to their origins. Signed tags are useful for security, but they are also useful for legal reasons, in case anyone claims Linux uses code taken from a non-GPL-compliant source.

Arnd Bergmann, for example, was traveling one day and wasn't in a position to sign his pull request. He told Linus, "I'm travelling at the moment and forgot to tag it first. The top commit in the branch is tagged with my usual key, and I can send the signed tag when I'm back on Saturday." To which Linus replied, "I've tried to encourage everybody to use signed tags, but for kernel.org pulls it's not like it's a hard requirement, so no worries."

In another case, Tejun Heo sent in some pull requests, which Linus accepted – but he also said:

"Your two pulls sadly broke the perfect signed tag streak that we had this merge window up until just now.

"I'm very happy to see how people have started doing signed tags for everything, but that only makes your pull requests stand out even more.

"So yeah, despite not requiring it for kernel.org pulls, I'd _really_ like to make using signed tags just be the standard behavior for all the kernel pull requests.

"Can we try to aim for that next time? Please?"

What's really happening here? If Linus told everyone he would only accept signed pull requests, everyone would surely comply. And he has told people regarding build warnings that he will no longer accept patches that produce warnings. What's the difference between these situations?

In this particular instance, Tejun replied, "Hahaha, yeah, I lost my private key many years ago, so gotta get that sorted out first. Will do the signed pull from now on." And when Linus suggested using the same PGP key he used for his *kernel.org* account, Tejun added, "I don't have the private part of that pgp key anymore and have been just using the same ssh key. Nothing really requires the private key that I lost at least 5 years ago. On the plus side, the pgp key has been as secure as it gets. :) So, I gotta generate new keys, get it signed and replace the korg key and so on. I've just been really lazy."

There have been numerous other cases of Linus engaging in this form of "public shaming" to call people out for not using signed tags. It is always fairly lighthearted and nonthreatening, though we all know that Linus can be very harsh when he wants to be.

In another email thread, Al Viro submitted a pull request, and Linus replied, "Pulled. I do note that you are one of the remaining people not using signed tags. Not the only one, but _almost_. I've cajoled almost everybody else to use them…."

I think, first of all, each issue is its own special case. Whether it's build warnings, signed tags, the maximum line length in source code files, or the correct moment to break the ABI, the approach is always based on the specific situation under consideration.

In the case of signed pull requests, I think Linus considers it fundamentally outside of the kernel source tree. Warnings at compile time are another matter – in that case, the kernel code itself has problems that the developers just didn't care enough to fix.

Signed pull requests are not directly related to the kernel code. If a request is signed or unsigned, the kernel code remains the same. The exception would be if someone outside the normal process managed to sneak some hostile code into the pull request by faking the identity of the true maintainer.

However, there are also other mechanisms to ensure that only good code goes into the kernel. Presumably, each patch is looked at and tested by several developers and maintainers before being submitted to the kernel tree. And each of those reviewers adds their name to the list of people who have approved the patch. So as code wends its way into the kernel, there is some amount of history regarding who has passed judgment over it.

Changing the kernel's ABI, on the other hand, will break any compiled binaries that rely on that interface. Usually the only thing that could compel Linus to do that would be a security flaw that simply needed to be fixed. But in rare cases, if it's truly true that no one anywhere in the world is using a particular part of the ABI, and if removing that part would make the kernel more maintainable and cleaner, sometimes Linus will let it happen.

I find it fascinating trying to understand why Linus leads kernel development the way he does. I'm sure he would tell me that a lot of it is just his personal choice, and that other choices would be just as effective. But, still, I think it's extremely cool that his choices have led to Linux becoming the standard operating system for pretty much the entire known universe, with only some behind-the-times desktop users still struggling along with other systems. ■■■

FOSSLIFE

Open for All

News • Careers • Life in Tech
Skills • Resources

FOSSlife.org

What is machine learning?

Wise from Experience

"I won't make this mistake again," you promise yourself. In other words, you'll learn from experience. If you translate experience into data, computers can do that, too. We'll introduce you to the fundamental forms of machine learning.

By Jörg Frochte

In May 2022, a friendly person from *Linux Magazine* wrote to me and asked if I could write a short article on machine learning. After a few quick queries, I agreed, as you can see here. I'm going to try to be more creative in fulfilling this request than just restating the Wikipedia entry on machine learning techniques.

Now that we have known each other for a few lines, let me ask you a personal question: Did you have a friendly person from *Linux Magazine* in your mind's eye while you were reading the opening lines? Second question: No matter who it was – can you imagine why? Most of us piece images together in our heads from the experience we gain while reading. Because you didn't have any information, your brain used the experience it had, or thought it had, in combination with your knowledge of media in general or *Linux Magazine* to be more specific.

In other words, you used experience to make inferences about a new, unknown situation, and you do it all the time without even noticing. Of course, this doesn't always work, but – as often as not – that isn't a problem. Machine learning endeavors to give computers the ability to learn and generalize from experience. *Experience*, in computer-speak, means data, and the learning we're talking about has nothing to do with consciousness or intelligence in the strict sense. It's about acquiring and, if necessary, improving skills from experience without being programmed to do so using a legacy approach.

That's Arthur L. Samuel's definition, or something like it, from back in 1959, and I think you can infer one thing from that date – that is, what machine learning isn't: It isn't a buzzword. It is a fairly old discipline for computer science with roots close to mathematics, which undeservedly gives it some potential for instilling fear in students and users.

In this article, I'll describe three main types of machine learning: supervised learning, unsupervised learning, and reinforcement learning. In addition, there are a few variants – such as semi-supervised learning and the like – but I'll not dwell on them today. As you will learn in this article, the differences between these three fundamental types of machine learning is primarily the way in which experiences are delivered.

Supervised Learning

Supervised learning relies on the existence of pairs of desired input X and output Y. For example, X could consist of images of animals, say dog(1), cat(2), mouse(3), and elephant(4). The numbers that follow are the Y values that people assign beforehand for a certain number of images. Using this database of examples, you can monitor the learning process and define a target. The computer then learns the function $f{:}X{-}{>}Y$ that assigns one of the four classes to each input image.

The set of images used for training is referred to as the training set. Ideally, at the end of the training, the computer will be able to assign each image from this set to the correct class. But simply repeating the assignments given in the training set doesn't add any value. The real power is in generalization, that is, using the lessons learned through training to process new examples. The reliability of a function (in machine learning it would typically be known as a *model*) is a measure of its ability to successfully process new examples.

The new examples used to test the model are known as the test set. Novices will often ask, "How many data pairs do you need for successful training?" or "How many pairs do you need

Figure 1: Supervised learning takes place wherever there is a teacher who knows the right answers and can correct the learner. Learning is by example, based on answers judged by the teacher to be right or wrong. © Ian Allenden, 123RF.com

for a meaningful test?" Both these questions are based on false premises. It's not so much a question of the number of examples or the size of the database – it's more about the information.

In the preceding example, for instance, if the training set only contains images of the dogs at the local dachshund kennel, and the test set contains images of the general dog population, the testing will not go well no matter how many training images you use. No matter how many times you take pictures of the dachshunds, and no matter the angle of the pictures or what the dogs are doing, the model has still never seen a husky or a mastiff. Of course, more is better when you're dealing with data, but it all boils down to whether the training set represents a good and representative mix.

This need for a good mix is also true for the test set. What the set doesn't include cannot be tested. Don't panic: In many models, learning by example can be used in conjunction with domain knowledge to deal with the unknown, but that approach will not work everywhere. Just to recap, supervised learning requires datasets that people have already embellished, and, although more is better than less, it's ultimately the mix that matters (Figure 1).

By the way, the discussion so far has covered one of the two main types of supervised tasks in our heads: classification. The other is regression. In a certain sense, the difference lies primarily in the space you map given a function $f{:}X{-}{>}Y$. If Y is discrete – that is, if it contains only a few individual elements (dog, cat, mouse, elephant), then you have classification. If Y is continuous, for example, representing a credit line, a temperature, or a probability, the result is a regression. By the way, from a math point of view, this is always a form of optimization problem based on stochastic data. The reason is that you basically add up the errors and naturally want to keep this total low.

Unsupervised Learning

The alternative to supervised learning is *unsupervised learning*. Unsupervised learning does not rely on target information in the sense of (x,y) pairs. You only have the X examples themselves. What can you do with them? Probably more than you think.

Unsupervised learning exclusively relies on working directly with the structures that exist in the examples. One of the most common applications is clustering these examples (Figure 2). You form groups based on the similarity of each example. For a computer to do this, however, you need to define the similarity cleanly in terms of a distance function – mathematicians refer

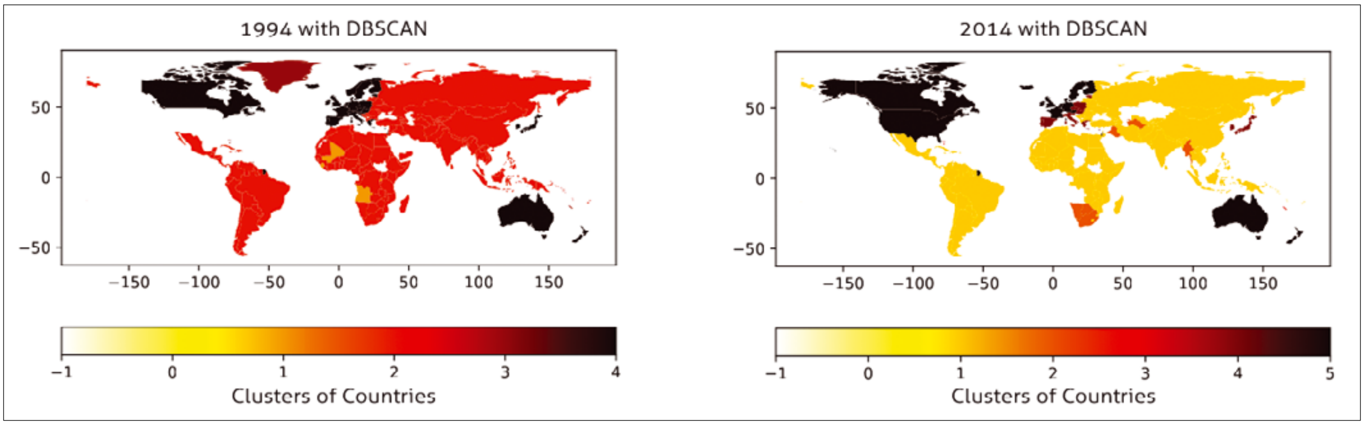


Figure 2: An unsupervised classification of countries into clusters in 1994 and 2014. Countries shown in white are outliers that do not belong to any cluster. As you can see in the 2014 classification, the USA loses its special role as an outlier in this classification, and the EU is broken up into two groups due to the aftermath of the 2008/2009 financial and economic crisis.

to this as a metric. Once you've established a metric, you're ready to go, and the machine learning algorithms form groups from the data. For instance, these groups could be groups of customers or users.

What's the point of forming groups? For one thing, grouping the datapoints could help you determine purchase or product recommendations. For many applications, you don't need a defined class as required for supervised learning. If you manage to sort people with similar behavior into groups, you can draw conclusions on one from the other and use these conclusions to create recommendation systems.

You can also identify groups that have emerged due to the different viewpoints of disciplines. This could mean using unsupervised learning as an intermediate step in a scientific process, with the computer suggesting an alternative taxonomy. One example of a taxonomy is the practice of dividing languages into families based on the relationships among them.

The opposite of grouping can also be interesting. In this scenario, you still create groups, but you are not interested in the groups themselves but are instead focused on the outliers. One typical application would be detecting security breaches and attempted fraud.

In some systems, if one actor behaves completely differently from the others, it could be a sign of suspicious activity. However, you have to be careful how you apply this technique. With computer networks, for instance, you will immediately notice why this kind of analysis should only be used as an initial filter and not automated. It's typically users with administrative rights who exhibit the most suspicious behavior: working unusual hours, messing around with other people's files. It is often necessary to filter out these anticipated outliers in order to find the truly interesting information.

Reinforcement Learning

Supervised and unsupervised learning are best thought of as functions that assign an output, class, value, or cluster to an input. *Reinforcement learning*, instead, represents an object with methods, that is, an agent that acts autonomously in its own environment. That environment can be reality and the agent can be a robot, but it can also be a bot on the World Wide Web or something similar. This agent follows a policy, which is often abbreviated in literature by the Greek letter π.

Reinforced learning is not hard-programmed into the computer. Instead, the agent learns it based on data. This data is different from the data used with supervised or unsupervising learning. In supervised learning, the system is more or less told how to get it right. The opposite is true in unsupervised learning, where you have to make do with the structures you have.

Reinforcement learning is sort of in between. The agent receives feedback (typically called a reward). The agent's goal is to obtain the highest possible reward. Of course, this goal cannot be achieved in a single step – that would be pretty shortsighted. Instead, the reward is added up across the whole task (Figure 3). The agent doesn't get any advice on *how* to get things right, just a feedback signal. In fact, the humans are totally ignorant as to what the best policy could look like. If they knew that, they could have programmed it in directly.

Consider an example with children and parents that would easily take on unethical overtones if you were serious about it. The target is a tidy room, but you don't tell your children. You only give them feedback in the form of the kind of unhealthy food they really want to eat. The good thing about being a computer agent is that it wouldn't matter if you are handing out candy (positive reward), varying degrees of slaps (negative reward), or a mix of both. But in this example, I'll stick with the less violent but unhealthier picture.

An important aspect of reinforcement learning is when to give the positive or negative feedback. For example, you could stay in the room and give the child a piece chocolate every time a toy is taken from the floor and put in a cupboard. The agent/child will quickly understand that this is a desired behavior. But you could also give no feedback for a very long time before distributing an amount of chocolate equivalent to the amount of space freed up on the floor of the room right at the end. In this case, learning the desired behavior will likely to take longer.

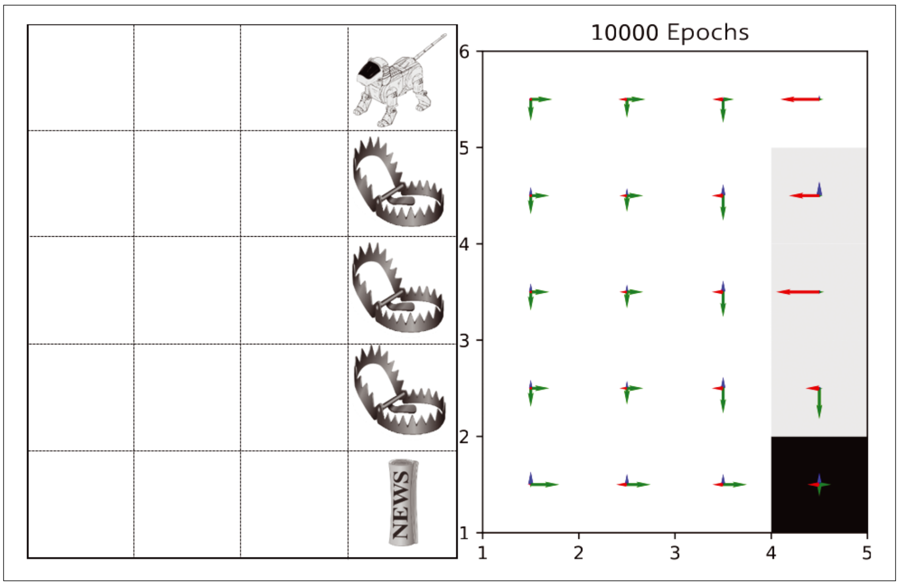


Figure 3: The agent's task is to reach the newspaper in as few moves as possible from any starting point. On the right, you can see the strategy acquired using an elementary reinforced learning approach. The longer an arrow, the clearer the tendency to move in the corresponding direction.

So should you always give feedback in as small a way as possible? Unfortunately, it's not that simple. After all, the agent's goal is not to clean up the room – it's to get the highest possible total reward. To do this, the agent needs to find the best strategy for the highest possible yield. Your goal, on the other hand, is to identify the best strategy for a real-world problem. There is often a massive difference between these goals.

To give a real world example: Imagine you want a cleaning robot that will not bump into your furniture. A bumper on the front of the robot determines whether or not this happens. Your goal is to avoid wrecking the furniture. From the agent's point of view, the most likely solution is to move in reverse, because then the bumper will not be depressed, which solves the problem from the agent's point of view but does nothing to protect your furniture.

You can experience similar unwanted effects by using feedback loops that are too strict. In fact, your prior knowledge always drives the design of the feedback loop. If your prior knowledge is good, everything will probably work out fine. But if you knew what the solution was, why didn't you just define it clearly in the first place?

When cleaning up a room, yes, parents also usually take a path that is more oriented toward supervised learning: They show children how to do things, or clean up the room along with the children, simply because they (think they) know how to do it best. The skill here is often to find the right feedback function, along with the right measure of quality on which it is often derived. You can mark off free space on the floor in the child's room – or do you just want to find everything dumped on the bed?

Incidentally, you again are dealing with optimization in stochastic environments: Optimization because the agent's goal is to maximize a target variable; stochastic because, from the agent's point of view, the environment usually has aspects that appear to be random.

Other Ways to Learn

You can use machine learning for some extremely sophisticated applications, such as creating or modifying text and images. However, these activities often consist of an intelligent combination of the three main techniques.

For example, Generative Adversarial Networks (GANs) work in a way that means that an artificial neural network *A* should be able to distinguish a real image of a dog from a false image or an image without a dog. We sometimes feed this network with stock images of dogs, and sometimes with images generated by another artificial neural network (let's call it *B*). If *A* makes a mistake, we find out about it and can use the mistake in the way we did before in the example of supervised learning with the dog, cat, mouse and elephant. On the other hand, if *A* is able to detect the false images from *B*, this failure is fed into network *B* as a negative used to optimize *B*. If this works, both networks will increasingly become better at their respective tasks.

Language or texts can be generated on the basis of neural networks with a memory. Currently, these are often transformer networks, although long short-term memory (LSTM) networks often do a good job, too. As supervised techniques, both variants basically only predict the probability with which sentence part, letter, or word is likely to follow next. Learning on the basis of very large bodies of language helps to develop a model that can continue texts or articles that have been started.

It would typically also be possible to implement something like this based on keywords. If you read somewhere that an artificial intelligence is now writing newspaper articles, there is actually still no intelligence involved. The point is simply that there are so many typical structures for, say, sports reports that it is easy for a model to generate a typical text from a few hard facts – who scored a goal and when, plus a couple of special features. That says more about the sports section of the newspaper than it does about AI.

This is not to say that a trained model cannot create something new. Thanks to the current copyright amendment from 2021, many people will be familiar with the term pastiche. Machine learning systems are very good at recombining things. If they receive positive feedback about what people appreciate, they can also create attractive images. If it is only a matter of imitating the style, a Neural Style Transfer (Figure 4), for example, provides attractive results more easily and stably than a GAN.

But this does not change the fact that these are essentially mathematical functions with inputs and outputs that improve thanks to optimization algorithms. The fact that optimization algorithms seem to perform more and more tasks that we previously thought would require a very intelligent human might suggest that we need to clarify our terms. For example, intelligence is not exactly a clearly defined concept. Judged by the simplest method – intelligence being what an intelligence test measures – machine models are likely to be intelligent soon.

For example, a large database of typical questions from intelligence tests was presented at the International Joint Conference on Artificial Intelligence in 2019. It is probably only a matter of time before the results, which were mediocre at the time, improve. If and when this happens, the criticism levied at this simplistic definition of intelligence is likely to intensify, and the term will probably have to be developed further. If science is lucky, we will discover a better definition for intelligence that can be formulated more scientifically.

In machine learning, there has been a decades-long competition between more accurate, faster, and farther (more data). Admittedly, these aspects are still important and will remain

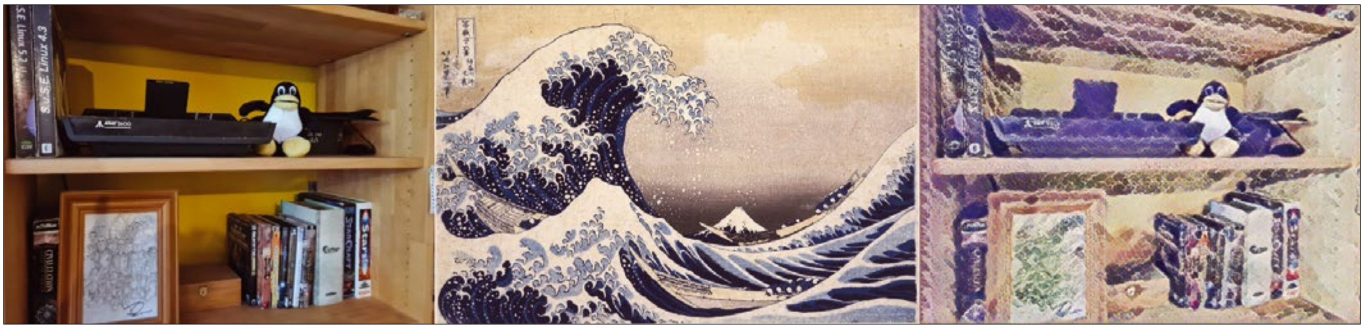


Figure 4: An example of Neural Style Transfer: The style from *Wave off Kanagawa* (center) is transferred to the content on the left. The result (on the right) depends on the extent to which the optimization prioritizes style and content fidelity.

so. However, many current questions, especially in Europe, revolve around other topics, such as Trustworthy AI, determination of uncertainty, Green AI, and others. Many aspects and discussions relating to the concept of the term Trustworthy AI are bundled in the 2021 "Ethics Guidelines for Trustworthy AI," published by the EU. Among other things, the topics of privacy and data governance, transparency and diversity, non-discrimination and fairness can be identified as specifically European impulses. Many things are happening centering around these issues in Europe right now.

People keep questioning whether Europe can keep up with competitors like China and the United States when it comes to AI – what they actually means in most cases is machine learning. The answer is probably that it depends on the topic. The term Green AI in turn comes from an article of the same name [1], the key phrase of which is: "The term Green AI refers to AI research that yields novel results while taking into account the computational cost, encouraging a reduction in resources spent." So far, little attention has been paid to resource consumption. If a few thousandth parts more accuracy can be achieved when distinguishing between different classes, this can often be worth publishing. This approach largely leaves out the costs of training and operation.

Dreams and Reality

In the end, even software based on machine learning algorithms is just software, which means it needs to be tested and maintained. And it would be great if it were open source, but that is easier to write than do.

When it comes to testing, it is important to realize that this is software that cannot easily be validated by classical testing approaches. The software itself is usually deterministic – you don't have to worry about a model classifying the same image as a dog in some cases and as a cat in others. But there are many other pitfalls. For example, in a real experiment [2], all wolves were identified as such. However, the attempt to apply the model to some images from a German zoo was depressing. There were many images of wolves in the training and test data, but all of them in the snow. The system had learned that "snow" was an essential factor for "wolf" – no snow, no wolf (Figure 5). Such failures cannot be determined by applying classical quality control.

The other point is: What would open source look like for software based on machine learning? All the widely-used machine learning software packages, like scikit-learn, TensorFlow/Keras, and PyTorch are released under open source licenses, but these are just the libraries that you use to build the software.

As an example, consider a piece of software that contains an artificial neural network. The weightings are not the kind of source text that is humanly readable and understandable but were extracted from a process. If you want to repeat the process, you need the libraries (no problem), information on the structure of the network (often supplied), the exact optimization parameters (this is where things start to dry up), and the dataset used for training. The last item on the list is usually off limits – you do not get to see the data.

This means that there are narrow limits as to what else can be done with a network like this if it is shared as part of open source software. It can be used for the intended purpose and with exactly the quality that was delivered. To improve it, you can usually only use this training status in the scope of transfer learning. Transfer learning means that a network has been trained for many image classes, for example.

One typical example is the ImageNet database, which contains many animals, people, and so on. A network of this kind, including the weighting values, can be taken and adapted to a new task for which data is available. For example, if you want to find South American birds only, you will not find all the species in ImageNet, but the network has already learned many useful filters. You can then get rid of the part of the network you can't use, and train the assembled network starting with the pre-trained status on your own collection of images of South American birds. This means that you can get by with fewer images and presumably

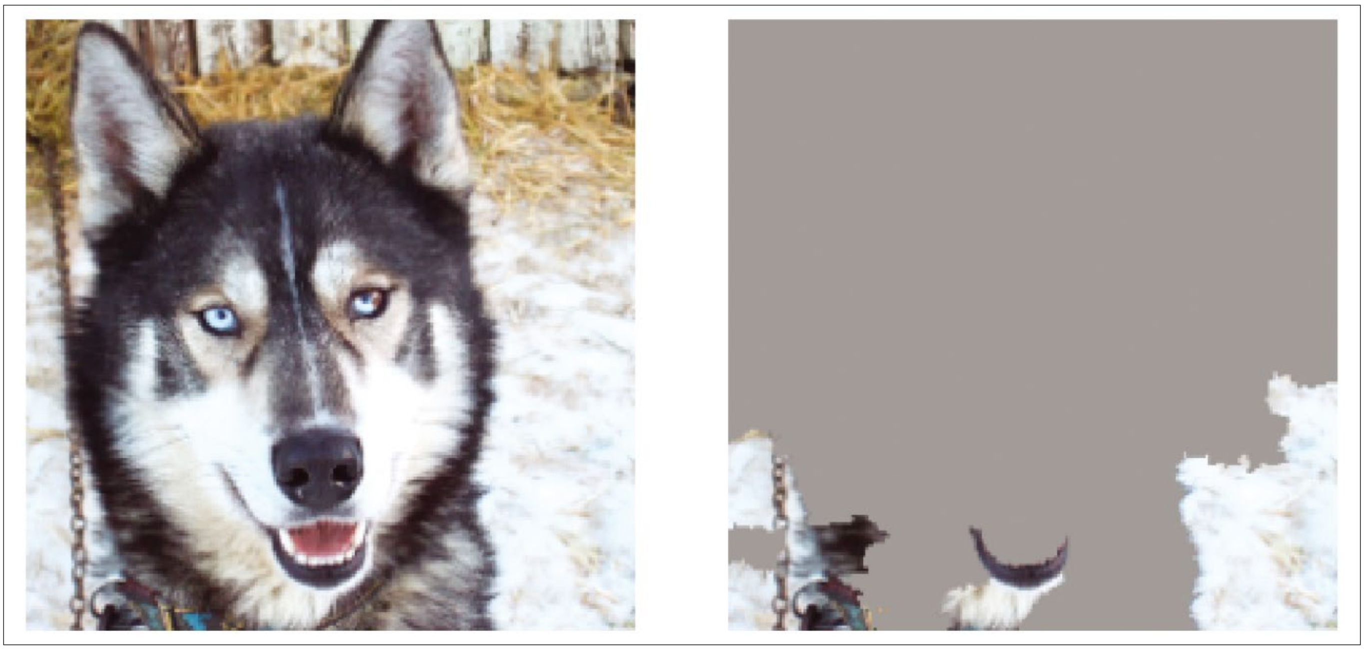


Figure 5: **This figure from the wolf identification experiment [2] shows what caused the bad decision.**

achieve higher accuracy with a lower-energy overhead in training. This is good and also quite often possible.

But the actual idea behind the demand for open source software is that you can change the software and pass on the changed (and thus usually improved) form. And this goal is no longer fully met. If not all data is available, the possibilities for further development are limited – not to mention the problems posed by the numerous software patents in the machine learning environment.

Conclusions

Machine learning has been around for a long time, but its growing importance alone should be an incentive to look further behind the scenes. In addition to the power of machine learning, it is also genuinely fun.

For example, if you have a large collection of LaTeX documents, why not use them to train an LSTM or Transformer network? You can even build a feedback loop through the compiler so that the system learns to write error-free LaTeX code. The content will certainly be a bit weird, but the exercise is entertaining.

Then, once you understand what it's all about, as an open source enthusiast, you might wonder what this technology means for your own movement. The conventional "copyleft" protection of the GPL was only meant to apply to software distribution and doesn't require the modifier to release modifications for software that runs as a service. (You might recall the ASP loophole that the AGPL was supposed to close.) Personally, I believe that the widespread use of machine learning software should lead us to rethink how we want to push forward with open source development and bring everything together with the concept of open data. ■■■

Info

[1] "Green AI" by R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, *Communications of the ACM*, vol. 63 no. 12, December 2020, pp. 54-63: *https://cacm.acm.org/magazines/2020/12/248800-green-ai/fulltext*

[2] Ribeiro, M. T., S. Singh, and C. Guestrin. "Why Should I Trust You?" In: *Proceedings of the 22nd ACM SIGKDD* (ACM, 2016): *https://www.kdd.org/kdd2016/papers/files/rfp0573-ribeiroA.pdf*

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 Inbox.

Linux Update • ADMIN Update • ADMIN HPC

Keep your finger on the pulse of the IT industry.

ADMIN and HPC: bit.ly/HPC-ADMIN-Update
Linux Update: bit.ly/Linux-Update

Photo by Andrew Neel on Unsplash

How computers learn languages

We Need to Talk

Whether through voice assistants, chatbots, or the automatic analysis of documents, rapid developments in AI are helping speech technologies make inroads. But how does AI manage to understand the subtleties of human language?

By Gerhard Paaß

Language is the medium through which people communicate and express their thoughts. It is an ancient dream of mankind to be able to communicate with a machine (for example, just watch *2001: A Space Odyssey*).

Meanwhile, science has come a bit closer to this vision. The box entitled "Sample Dialogue with LaMDA" contains a conversation with the Language Model for Dialogue Applications [1] (LaMDA) dialogue model. It was assigned the identity of a Weddell Seal in the first line. As you can see, LaMDA can give grammatically and contextually correct answers and even play with the meanings of words. But how does a computer system manage to achieve language fluency?

To understand language, the system needs to know the meaning of words. For this purpose, each word is represented by a long vector of 100 to 1,000 real numbers. This vector is known as embedding. Now, for example, because "sofa" and "couch" both refer to upholstered seating furniture for several people, their embeddings should also be similar. This means that very similar numbers should be found at the same positions of the vector. Other words such as "dive" have very different meanings, so their embeddings should be very different from the one just mentioned.

Sample Dialogue with LaMDA

[*Starttext*] Hi I'm a weddell seal. Do you have any questions for me?

[*User*] You're a talking seal?! No way!

[*LaMDA*] Yep, I am! I'm here answering any questions from all the seal fans out there.

[*User*] Haha seal fans, nice. Like fans of the band Seal?

[*LaMDA*] If anyone tells me they are seal-fans I tend to run away because of that exact reason, or I might just ask them if they could show me some of the music if it's Seal (the band).

[*User*] A seal singing Seal stamping a seal on a sealed envelope containing sealer.

[*LaMDA*] A seal-ception within a seal. I'd watch that movie.

[*User*] You are hilarious.

[*LaMDA*] Awh, thank you. But what about you? I must admit you're pretty funny yourself for suggesting that movie.

[*User*] Well I wasn't really suggesting it as a movie, that was you! :P

[*LaMDA*] Oh, I must have misinterpreted. Sorry! Still, I think it would be pretty cool to watch.

But there are also cases where the same word has different meanings. Which meaning comes into play then depends on the context. An example: Comparing the sentences "He rose from the chair" and "The rose is red" makes this pretty clear. Although the word "rose" looks the same in both cases, different embeddings are needed, depending on the other words in their context. However, this requires a method that automatically determines embeddings with these properties from training data.

Smart BERT

The BERT [2] model has brought about a breakthrough. The task is to predict masked words in arbitrary texts of the training set. First of all, the words are broken down into smaller word fragments using the WordPiece process. These can be the components of a compound noun, for example, or the word root and ending. These pieces are known as tokens. On the one hand, frequent words are tokens themselves; on the other hand, every word can be represented by tokens. Because the tokens can be combined into ever new words, 100,000 of them are all it takes for all the languages in the world.

By default, about one sixth of the tokens are masked for training and replaced by *[MASK]*. The goal is to be able to predict the masked token based on the surrounding ones. Each token is assigned exactly one token embedding on the lowest level, which is first filled with random numbers. This token embedding acts as a representation of the meaning of the token that does not depend on the meaning of the neighboring tokens.

In addition, the procedure requires information about the position of a token in the text, which is added in the form of an additional position embedding. Both token embedding and position embedding are derived during training. The training goal is to achieve the highest possible probability value for the masked token.

The actual task of the model is to generate context-sensitive embeddings that also distinguish between different meanings of the same word in different contexts. This is done with the help of an association module that determines the similarity between an embedding at position *i* and all other embeddings. The similarity is calculated using a scalar product of the embedding vectors controlled by parameters, where a high value means a high similarity.

The series of all similarity values for embedding at position *i* are then normalized into a probability such that all similarity values add up to 1. Using these similarity values for weighting, the method then totals the linearly transformed embeddings of all embedding vectors as a function of other parameters and generates the new embedding at position *i*. These computations are shown in Figure 1.

Evaluating similarity leads to a high weighting of embeddings for tokens with a related meaning and can strongly influence the initial embedding at *i* . This then allows BERT to interpret the word “rose” in the sample sentence in Figure 2 as “flower” through its relation to the word “picked”. A new embedding is created for each position in this way.

This procedure is carried out successively on several layers of the network with different parameters, continuously improving the context-sensitive embeddings. In addition, several association modules are used in parallel on each level, and each with different parameters. At the top level, a logistic regression model now has the task of predicting the probability of the masked word (in our example “gave”) from the context-sensitive embedding of the *[MASK]* token.

To do this, the model parameters in the modified scalar product and the transformed embedding are adjusted by optimization, causing the probability of the masked word to increase steadily. This forces the model to gather all available information about the missing word in the top layer embedding for the masked word during training.

The original BERT model had 100 million parameters and was trained on 3.3 billion words from Wikipedia and a collection of books. BERT was customized for special tasks by fine-tuning with a relatively small volume of annotated data. In this way, it was able to solve a large number of test tasks in a far superior way to other models, such as answering questions or detecting logical contradictions.

This procedure is also known as transfer learning. BERT acquires very detailed knowledge of correct language and meanings through the original pre-training on general language data. Fine-tuning then adapts the model to a special task without it losing its previously acquired skills.

GPT-3 Generates Texts

Creating context-sensitive embeddings through association modules has proven to be so effective that almost all natural language processing models now use this method. Language models generate context-sensitive embeddings of a starting text to

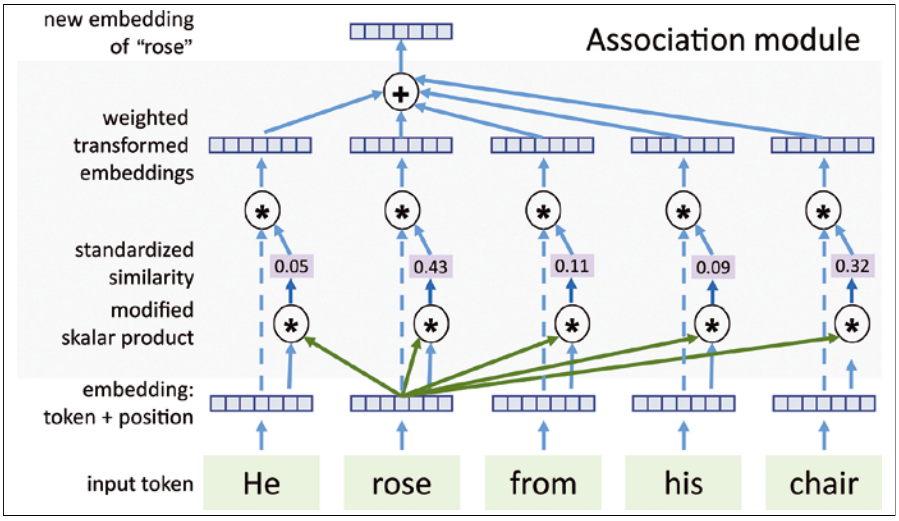


Figure 1: Computation of context-sensitive embeddings by an association model.

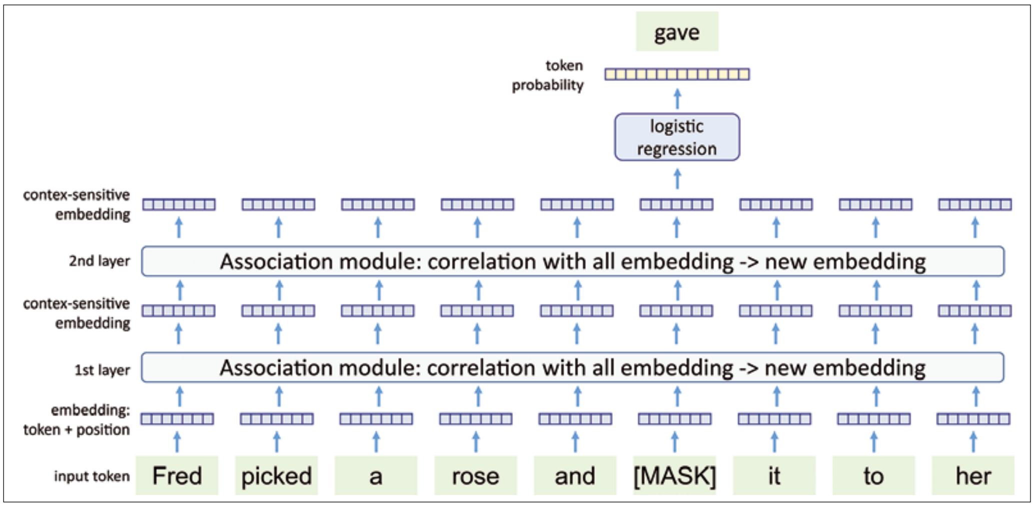

gave
token probability
logistic regression
contex-sensitive embedding
2nd layer
Association module: correlation with all embedding -> new embedding
contex-sensitive embedding
1st layer
Association module: correlation with all embedding -> new embedding
embedding: token + position
input token
Fred
picked
a
rose
and
[MASK]
it
to
her

Figure 2: Prediction of masked words by BERT.

predict which next word is the most likely. By using this technique several times in succession, you can gradually create a continuation of the text.

The GPT-3 model is one such language model. It includes 175 billion parameters and has been trained on texts with a total of 500 billion words [3]. Such gigantic models devour an enormous amount of computing power during training. Practicing the GPT-3 model on a V100 GPU server with a processing power of 28 TFLOPS would theoretically take 355 years and cost a good $4.6 million at a rate of $1.50 per hour.

This model can predict the last word of a paragraph in a benchmark dataset with 76 percent accuracy. The language model is very flexible because it can be instructed to perform certain tasks. The "GPT-3: Few-Shot Prompt" box provides a starting text for the model in which the user instructs with a couple of examples. The output shows that GPT-3 can interpret these Few-Shot Prompts and, for example, correct syntax errors in a sentence.

So you no longer have to program or fine tune the language model to perform a task. Instead, the model follows the implicit instructions from some examples. Hundreds of other ways to train the model for specific tasks exist. This approach is completely different from existing alternatives for solving textual tasks.

Foundation Models

The LaMDA model is a language model with 137 billion parameters trained on dialogues and other web texts with a total of 1,560 billion words. It uses two additional techniques to deliver particularly engaging dialogue posts.

First, it uses a search engine to locate up-to-date documents on a topic in databases or on the Internet. It adds relevant documents to the previous dialogue flow as additional text and processes them in the response. As a result, the model is able to provide up-to-date and factually accurate answers. Fine-tuning can adapt the model to make the answers more meaningful, significant, and interesting, as well as to avoid the possibility of toxic language.

As the example at the beginning of the article shows, the model is capable of providing very accurate answers. There has even been an intense dispute lately about whether or not LaMDA shows something like feelings on the level of a child. However, the manufacturer Google denies this and has suspended software engineer Blake Lemoine, who made this claim. In any case, this discussion illustrates that dialogue systems have reached an astonishing level of quality and consistency.

It has been shown that embeddings cannot only be derived for words, but also for partial areas of an image, sound sequences, video frames, similar media components, as well as DNA segments. In this way, one can achieve a common representation for different media, with which content from different modalities can be linked via the tested association module.

DALL-E 2, for example, which is able to generate new images for text, is based on this principle. As a result, it appears possible to represent, analyze, and generate many items of media content simultaneously by large language models. A large group of researchers therefore refer to these models as Foundation Models [4] because they will play a crucial role in the further development of intelligent systems. ■■■

GPT-3: Few-Shot Prompt

[*User*] Poor English input: I eated the purple berries.

Good English output: I ate the purple berries.

Poor English input: Thank you for picking me as your designer. I'd appreciate it.

Good English output: Thank you for picking me as your designer. I appreciate it.

[…]

Poor English input: I'd be more than happy to work with you in another project.

[*GPT-3*] Good English output: I'd be more than happy to work with you on another project.

Info

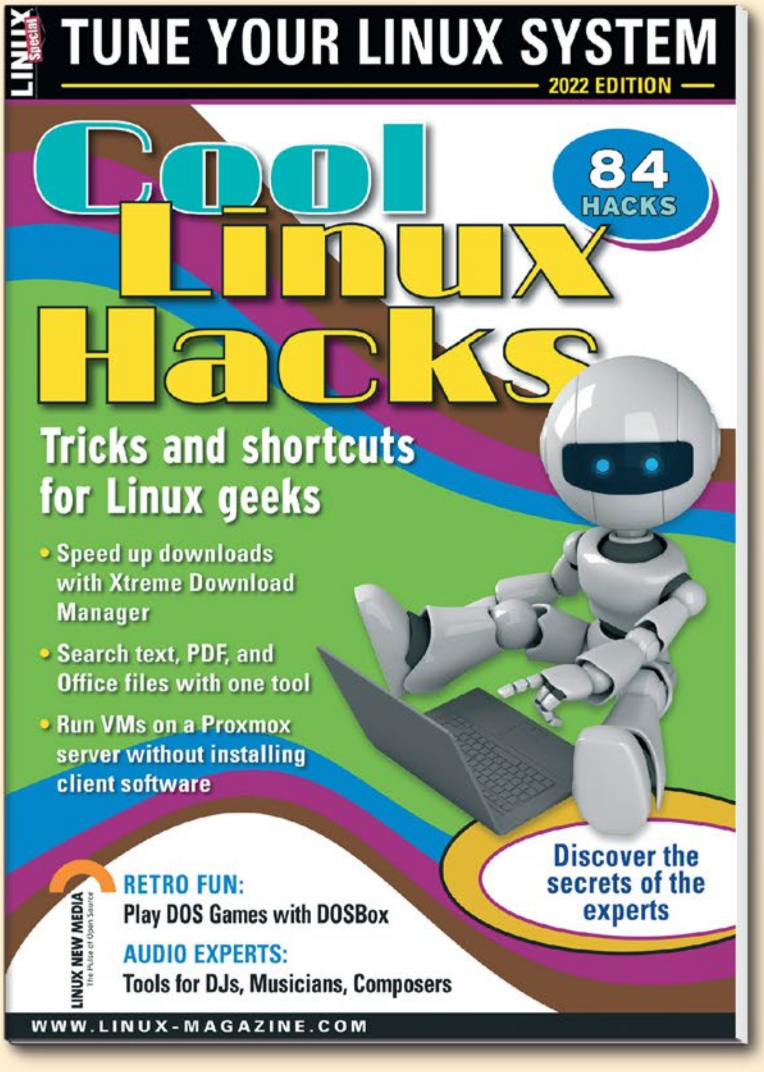
[1] LaMDA: *https://arxiv.org/abs/2201.08239*
[2] BERT: *https://arxiv.org/abs/1810.04805*
[3] GPT-3: *https://arxiv.org/abs/2005.14165*
[4] Foundation Models: *https://arxiv.org/abs/2108.07258*

Author

Gerhard Paaß, senior scientist at Fraunhofer IAIS, is a lecturer at the Universities of Bonn, Leipzig, and Brisbane. He founded the text mining group at Fraunhofer IAIS and is instrumental in the development of methods for semantic analysis of texts. His current work focuses on information extraction, story generation, and semantic learning using pre-trained language models. His book *Künstliche Intelligenz* (*Artificial Intelligence*) was recently published by Springer-Vieweg.

SHOP THE SHOP
shop.linuxnewmedia.com

GET PRODUCTIVE WITH
COOL LINUX HACKS



LINUX Special
TUNE YOUR LINUX SYSTEM
2022 EDITION

Cool Linux Hacks

84 HACKS

Tricks and shortcuts for Linux geeks

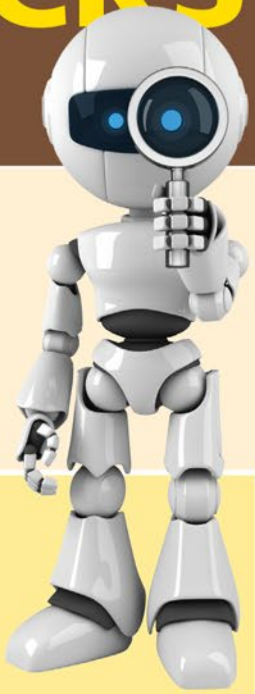
- Speed up downloads with Xtreme Download Manager
- Search text, PDF, and Office files with one tool
- Run VMs on a Proxmox server without installing client software

Discover the secrets of the experts

LINUX NEW MEDIA
The Pulse of Open Source

RETRO FUN:
Play DOS Games with DOSBox
AUDIO EXPERTS:
Tools for DJs, Musicians, Composers

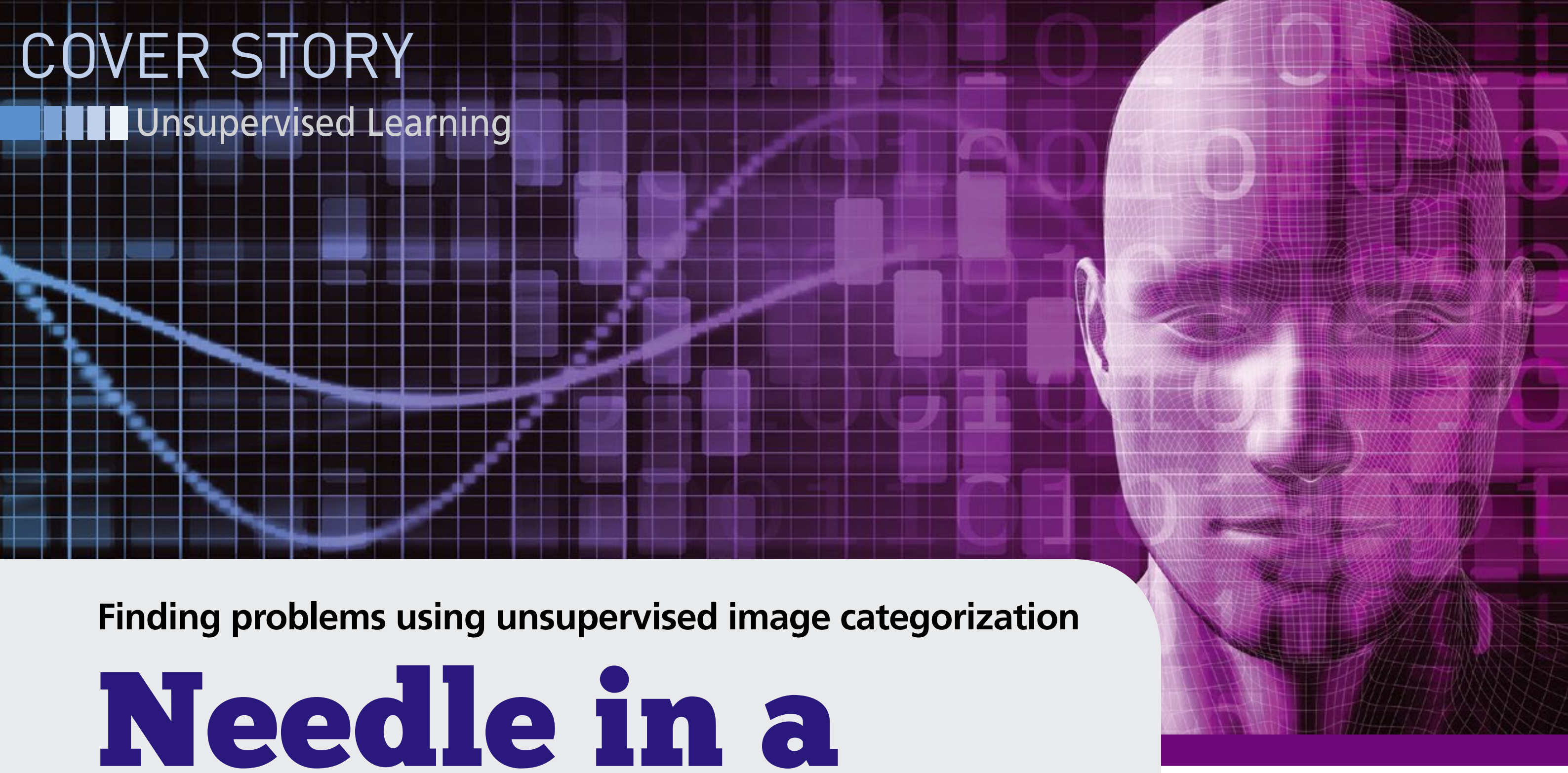
WWW.LINUX-MAGAZINE.COM



Improve your Linux skills with this cool collection of inspirational tricks and shortcuts for Linux geeks.

- Google on the Command Line
- OpenSnitch Application Firewall
- Parse the systemd journal
- Control Git with lazygit
- Run Old DOS Games with DOSBox
- And more!

ORDER ONLINE:
shop.linuxnewmedia.com/specials



Finding problems using unsupervised image categorization

Needle in a Haystack

The most tedious part of supervised machine learning is providing sufficient supervision. However, if the samples come from a restricted sample space, unsupervised learning might be fine for the task. *By Garry Tuohy*

In any classification project, it is certainly possible to get someone to review a certain number of images and build a classification list. However, when entering a new domain, it can be difficult to identify domain knowledge experts or to develop a ground truth for classification upon which all experts can agree. This is true regardless of whether you are looking at the backside of a silicon wafer for the first time, or if you are trying to identify the presence of volcanoes in radar images from the surface of Venus [1].

Alternatively, you can bypass all these problems and kickstart a classification project with unsupervised machine learning. Unsupervised machine learning is particularly applicable to environments where the typical images are largely identical, much like the pieces of hay in the haystack that you need to ignore when looking for needles.

In this article, I examine the potential for using unsupervised machine learning in Python (version 3.8.3 64-bit) to identify image categories for a restricted image space without resorting to training neural networks. This technique follows from the long tradition within engineering of finding the simplest solution to a problem. In this particular case, the solution relies upon the ability of the functions within the OpenCV and mahotas computer vision libraries to generate parameters for the texture and form within an image.

As an example, I'll look at the images obtained during the semiconductor manufacturing process from the bevel of silicon wafers. As part of the quality control procedure for semiconductor manufacturing, a series of photo images are taken for each wafer. Ideally, the wafers are normal, so the images are identical, but occasionally, a dissimilar photo can reveal a potential manufacturing problem that can generate defects on the affected wafer. Of course, you could train a human to wade through all these photos and look for problems, which would certainly be thorough, but it would take a lot of time and would introduce the possibility of human errors, especially as tedium develops. You could also train a neural network to look for dissimilar images, but neural networks need large amounts of compute resources, not to mention the expertise required for programming and training, as well as a sufficiently broad library of examples for each classification.

A simpler solution is to check the images using unsupervised data analysis techniques. The first step is to derive digital parameters for each of the photos for easier comparison. In this case, I used the Hu Moments and Haralick texture features, which are available through the cv2 and mahotas computer vision libraries. Hu Moments is an image descriptor that characterizes the *shape* of an object within an image. The Haralick texture features unsurprisingly describe the *texture*.

I then use principal component analysis (PCA) to reduce the data's dimensionality while still preserving as much of the variance as possible. The points are then grouped using a density-based clustering algorithm to identify the main categories of images as well as abnormal images. I relied upon the Seaborn and Matplotlib libraries to generate the visualizations.

Once the clusters have been defined, it is then possible to display sample images from the boundary region of each cluster as examples of the range of images that are typical within each cluster. Defining the boundaries allows you to make a

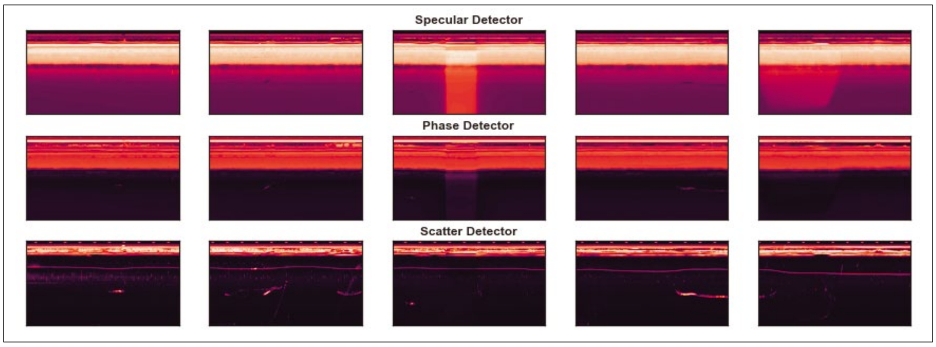


Figure 1: Examples of specular, phase, and scatter images.

convenient comparison of the typical images in each cluster and to clearly identify how effective the method is for unsupervised image categorization.

Background

The bevel inspection tools generate three distinct image types: specular, phase, and scatter. A single image is generated for the entire wafer bevel. However, to simplify the viewing, the images are segmented into 36 equal parts, each representing 10 degrees of the wafer bevel. Figure 1 shows five examples of each of these image types.

Goal

These images may contain localized defects (see the third row in Figure 1). However, these defects are not the primary focus of this work. Instead, the main concern is the general structure and texture within the images and how these may change on and between individual wafers. Artifacts of interest can clearly be seen in the images shown in the third and fifth columns in Figure 1. The importance of these artifacts requires input from a domain knowledge expert; however, you can attempt to segregate these from the less noteworthy images.

Data Collection

The identification and collection of the relevant data from a primary source is often minimized, even though it could be considered the most essential part of data wrangling. The code snippet in Listing 1 demonstrates a succinct way to connect to an Oracle database [2] and use a cursor to extract your username. This simple query (i.e., `select user from dual`) needs to be replaced by an appropriate query. (I did not include the actual queries that were used here as they are not generally applicable.) The login credentials and database identification strings were all stored separately in a config module. (You could use a password manager such as PyKeePass for storing login information.)

The data returned by a query can be conveniently imported into a pandas DataFrame with the following command:

```
my_df = pd.DataFrame(cursor.fetchall())
```

Listing 2: Loading an Image into Memory from a URL

```
import requests
from PIL import Image
from io import BytesIO

def get_URL_img(URL):
  # Create a Session to contain you basic AUTH and persist your cookies
  authed_session = requests.Session()
  authed_session.auth = (local_config.WEBUSERNAME, local_config.
                         WEBPASSWORD)
  USER_AGENT = "Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.17
              (KHTML, like Gecko) Chrome/24.0.1312.27 Safari/537.17"
  authed_session.headers.update({'User-Agent': USER_AGENT})

  # Fetch the actual data
  fetched_data = authed_session.get(URL)
  # Convert into an image
  return Image.open(BytesIO(fetched_data.content))
```

Listing 1: Querying Username from the Database

```
import cx_Oracle
try:

  with cx_Oracle.connect(
      local_config.username,
      local_config.password,
      local_config.dsn,
      encoding=local_config.encoding) as conn:
    with conn.cursor() as cursor:

      # Now execute the sqlquery
      cursor.execute("select user from dual")
      print(cursor.fetchmany(20))

except cx_Oracle.DatabaseError as e:
  print("There was a problem with the YMS query ", e)
```

Listing 3: Feature Extraction Functions

```
import numpy as np
import cv2
import mahotas as mt

# Function to Extract  Hu Moments
def extract_hu_moments(image):
  feature = cv2.HuMoments(cv2.moments(np.array(image))).flatten()
  return feature

# Function to Extract Features
def extract_texture(image):
    # calculate haralick texture features for 4 types of adjacency
    textures = mt.features.haralick(np.array(image))

    ht_mean = textures.mean(axis=0)
    return ht_mean
```

Listing 4: Extracting Features from All Images

```
import pandas as pd
imgHaralickFeatures = pd.DataFrame
imgHuMoments = pd.DataFrame

for idx, wafstepIMG in wafStepIMGList_df.iterrows():

  imgURI = local_config.imgServerRoot + wafstepIMG['FILENAME']
  print("This URI:", imgURI, wafstepIMG['STEP_ID'])
  img = get_URL_img(imgURI)

  if img.size == (1820, 1002):
    print("Correct image size: ",img.size)
    # Extract and store Haralick features from bevel image
    har = extract_texture(img)
    if imgHaralickFeatures.empty:
      imgHaralickFeatures = pd.DataFrame(har.reshape(-1, len(har)),
                            [wafstepIMG['IMAGE_ID']] ).add_prefix('haralick_')
    else:
      imgHaralickFeatures = imgHaralickFeatures.append(pd.DataFrame(har.reshape(-1,
                            len(har)), [wafstepIMG['IMAGE_ID']] ).add_prefix('haralick_'))

    # Extract and store Hu Moments from bevel image
    hum = extract_hu_moments(img)
    if imgHuMoments.empty:
      imgHuMoments = pd.DataFrame(hum.reshape(-1, len(hum)),
                     [wafstepIMG['IMAGE_ID']] ).add_prefix('hu_')
    else:
      imgHuMoments = imgHuMoments.append(pd.DataFrame(hum.reshape(-1, len(hum)),
                     [wafstepIMG['IMAGE_ID']] ).add_prefix('hu_'))

  else:
    print("Wrong image size: ",img.size)

imgFeatures = imgHaralickFeatures.join(imgHuMoments, how='outer')
```

Listing 5: Independent DataFrames for Detector Types

```
# Map filename Extension to Detector Type
dectorType = {
  "Specular": "000",
  "Phase": "001",
  "Scatter": "002"
}

# Load BevelImage DataFrames
import pandas as pd

imgFeatures = pd.read_pickle('image_features.pkl')
wafStepIMGList_df = pd.read_pickle('waf_Step_IMG_List.pkl')
wafStepList_df = pd.read_pickle('waf_Step_List.pkl')

# Select the features for a single Detector Type
# Specular Detector
imgFeaturesSpecular = imgFeatures.filter( items =  wafStepIMGList_df[wafStepIMGList_
                      df['EXT'] == dectorType["Specular"] ]['IMAGE_ID'] , axis=0)

# Phase Detector
imgFeaturesPhase = imgFeatures.filter( items =  wafStepIMGList_df[wafStepIMGList_
                   df['EXT'] == dectorType["Phase"]]['IMAGE_ID'] , axis=0)

# Scatter Detector
imgFeaturesScatter = imgFeatures.filter( items =  wafStepIMGList_df[wafStepIMGList_
                     df['EXT'] == dectorType["Scatter"]]['IMAGE_ID'] , axis=0)
```

The essential pandas library [3] provides access to many data structures and methods that greatly simplify manipulation. The DataFrame and its methods are primary among these and should be familiar to users of the R programming language.

To avoid storing all images on the local machine, a pseudo data pipeline was created to pull down each image via a URL, extract the relevant features, and then delete the image. The *get_URL_img* function in Listing 2 uses the Python Image Library (PIL) [4] to extract each image via its URL onto the local filesystem.

The functions shown in Listing 3 extract the Hu Moments (`extract_hu_moments()`) and Haralick texture features (`extract_texture()`) from a specified image. These functions can be found in the cv2 [5] (which is the import name for opencv-python) and mahotas [6] libraries, both of which require NumPy [7].

Listing 4 then applies the `extract` functions on each image in turn using various pandas methods [8] to combine all results into a single DataFrame (i.e., `imgFeatures`).

PCA

PCA, a machine learning technique used for dimensionality reduction, is sometimes referred to as feature extraction. PCA is most appropriate for datasets with an unwieldy number of parameters but without classification, which allows PCA to be used in an unsupervised context. PCA's goal is to preserve the salient information within a dataset while generating a more manageable number of virtual parameters that are constructed from a subset of the original parameters. The number of these virtual parameters is configurable, depending on the amount of the variance desired, but typically two are generated because of the convenience of visualization.

The first principal component usually explains 75-80 percent of the total variance, and the second can be expected to represent 12-20 percent. PCA can also be described as finding a new projection line within the parameter space that maximizes the variance of data projected along it. These lines are the principal components.

The resulting DataFrames can be pickled (serialized) for later analysis using the following command:

```
my_df.to_pickle(fileName.pkl)
```

Analysis

When an extended set of parameters is encountered, as in the example presented in this article, the most obvious means of reducing this feature space is to perform PCA (see the "PCA" sidebar). Listing 5 separates the data of each detector type to perform PCA on each group independently.

Standardizing the values within each component prevents bias resulting from individual components simply because they contain larger values (see Listing 6). This step is provided by scikit-learn and imported via sklearn [9].

Once the data is in a fit state to perform PCA, the transformation can be performed via another sklearn sub-library [10] (see Listing 7). The decomposition into two components allows you to plot the results conveniently.

In order to simplify subsequent data handling, a label column is added to each DataFrame to identify the detector type (specular, phase, or scatter), and all DataFrames are subsequently concatenated together (see Listing 8).

The `jointplot` functions available from the Seaborn library provide a succinct way to visualize any structure that may have been revealed by PCA (see Listing 9).

The PCA density plots in Figure 2 show these results: Not only has the

Listing 6: Standardizing Values

```
from sklearn.preprocessing import StandardScaler

stdimgFeaturesSpecular = pd.DataFrame(StandardScaler().fit_
                         transform(imgFeaturesSpecular), index=imgFeaturesSpecular.
                         index, columns=imgFeaturesSpecular.columns)
stdimgFeaturesPhase = pd.DataFrame(StandardScaler().fit_transform(imgFeaturesPhase),
                      index=imgFeaturesPhase.index, columns=imgFeaturesPhase.columns)
stdimgFeaturesScatter = pd.DataFrame(StandardScaler().fit_
                        transform(imgFeaturesScatter), index=imgFeaturesScatter.
                        index, columns=imgFeaturesScatter.columns)
```

Listing 7: Extracting PCA Components into Two Axes

```
# import PCA from sklearn library
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
#tranform the given data set using PCA
principalDfSpecular = pd.DataFrame(data =pca.fit_transform(stdimgFeaturesSpecular),
                      index=stdimgFeaturesSpecular.index, columns = ['PC1', 'PC2'])
principalDfPhase = pd.DataFrame(data =pca.fit_transform(stdimgFeaturesPhase),
                   index=stdimgFeaturesPhase.index,  columns = ['PC1', 'PC2'])
principalDfScatter = pd.DataFrame(data =pca.fit_transform(stdimgFeaturesScatter),
                     index=stdimgFeaturesScatter.index, columns = ['PC1', 'PC2'])
#principalDf['PC1']
```

Listing 8: Adding Column and Combining DataFrames

```
principalDfSpecular['Detector'] = 'Specular'
principalDfPhase['Detector'] = 'Phase'
principalDfScatter['Detector'] = 'Scatter'

principalDf = pd.concat([principalDfSpecular, principalDfPhase, principalDfScatter])
```

Listing 9: PCA Data for Each Detector Type

```
import seaborn as sns
from matplotlib import pyplot as plt

plt.figure(figsize=(8,5))
sns.jointplot(data=principalDf[principalDf['Detector'] =='Scatter'], x='PC1',
              y='PC2', kind="hex", joint_kws=dict(bins='log'))
```

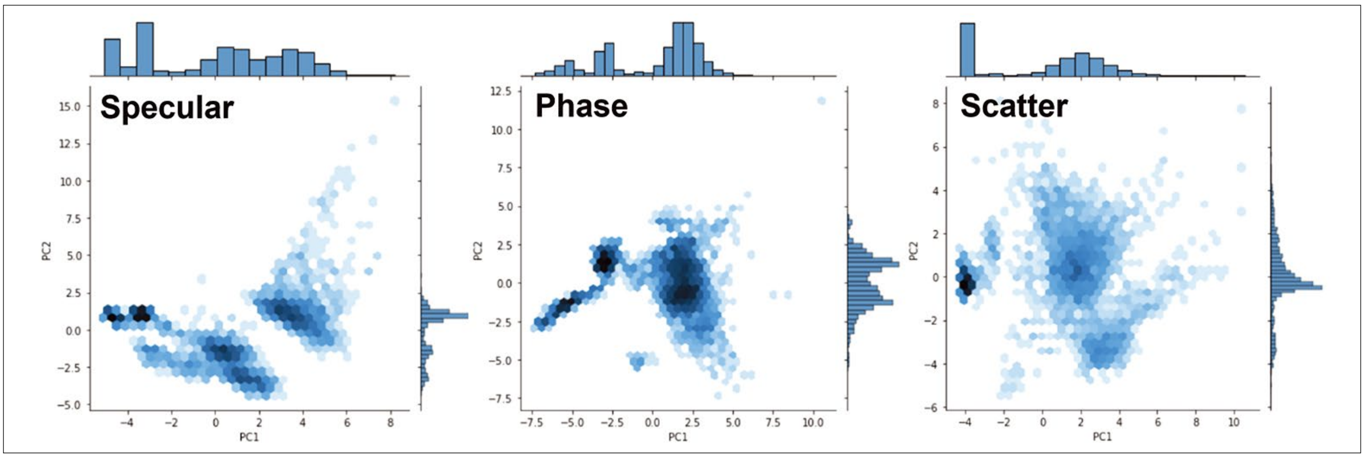


Figure 2: PCA density plots are shown here for the specular, phase, and scatter images.

structure been revealed, but the structure is also distinctly different for each detector type.

To quantify this structure, some form of clustering needs to be applied to the PCA-transformed data. A natural choice is K-means clustering. However, in this case, the results are generally unsatisfactory at separating meaningful structures. For example in Figure 3, while the grey cluster and red random points are reasonably well separated, the other group of data points are crudely segmented, without regard for the internal structure. This is not necessarily always the case, and other examples exist where K-means clustering is an appropriate option [11].

Listing 10: Applying the DBSCAN Algorithm

```
from sklearn.cluster import DBSCAN
from sklearn import metrics

# Extract PCA vales into an array to perform clustering
X = df.iloc[:, [0, 1]].values
type(X)

# Compute DBSCAN
db = DBSCAN(eps=0.35, min_samples=22).fit(X) # DEFAULT
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

print("Number of clusters: %d" % n_clusters_)
print("Number of random points: %d" % n_noise_)
print("Silhouette Coefficient: %0.3f" % metrics.silhouette_
      score(X, labels))
```

Listing 11: Plotting DBSCAN Clustering

```
import matplotlib.pyplot as plt

unique_labels = set(labels)
colors = [plt.cm.Spectral(each) for each in
          np.linspace(0, 1, len(unique_labels))]
for k, col in zip(unique_labels, colors):
  if k == -1:
    col = [0, 0, 0, 1]

  class_member_mask = labels == k

  xy = X[class_member_mask & core_samples_mask]
  plt.plot(xy[:, 0],xy[:, 1],"o",markerfacecolor=tuple(col),
           markeredgecolor="k",markersize=6 )

  xy = X[class_member_mask & ~core_samples_mask]
  plt.plot(xy[:, 0],xy[:, 1],"o",markerfacecolor=tuple(col),
           markeredgecolor="k",markersize=3)

plt.title("Estimated number of clusters: %d" % n_clusters_)
plt.show()
```

Applying a DBSCAN clustering algorithm [12] produced much more promising results. Listing 10 applies this clustering algorithm with a minimum number of samples (22 in this example) per cluster to the two PCA components generated in Listing 7. The results of the DBSCAN clustering algorithm can be visualized with the help of the Matplotlib library, as shown in Listing 11. Figure 4 shows the results after the PCA data has been clustered using DBSCAN.

To visualize the differences between bevel images that were assigned different cluster IDs, the IDs must be merged with the DataFrame containing the images' and wafers' metadata from which the IDs were obtained (see Listing 12).

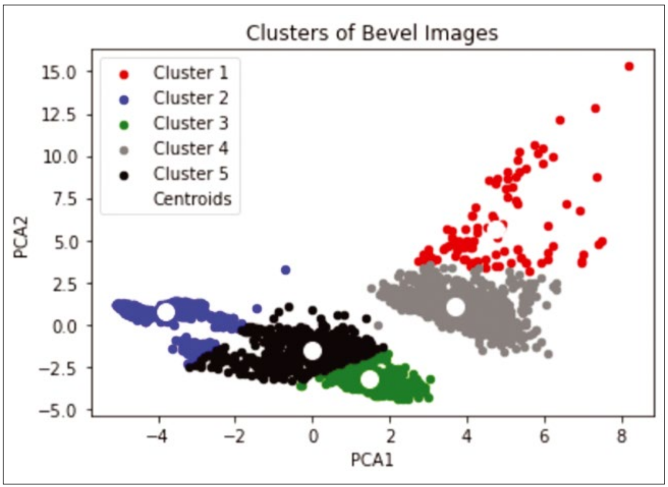


Figure 3: K-means clustering has been applied to the PCA feature space.

Listing 12: Merging Cluster IDs

```
Xlabels = pd.DataFrame(np.append(X, labels.reshape((4281, 1)),
          axis=1))
Xlabels.columns = ['PC1', 'PC2', 'Cluster']

# Copy IMAGE_ID from the index to a column
df['IND'] = df.index

# Merge Cluster IDs with the IMAGE_IDs, using the PCA values
df = pd.merge(df, Xlabels, on=["PC1", "PC2"])

# Reset Index
df.set_index('IND', drop=True, inplace=True)
df.index.name = None

# Merge the Clustered PCA dataframe with the Image and Wafer
  metadata
df['IMAGE_ID'] = df.index
dfPLUS = df.merge(wafStepIMGList_df, on='IMAGE_ID')
dfPLUS = dfPLUS.merge(wafStepList_df, on=('WAF_ID','STEP_ID'))
dfPLUS['LOT_GRP'] = dfPLUS['WAF'].str.replace('[^A-Z]+', '')

# Pickle individual DataFrame with Clustered data to the
  filesystem
#dfPLUS.to_pickle('clusteredSpecularImgPLUS.pkl')
#dfPLUS.to_pickle('clusteredPhaseImgPLUS.pkl')
dfPLUS.to_pickle('clusteredScatterImgPLUS.pkl')
```

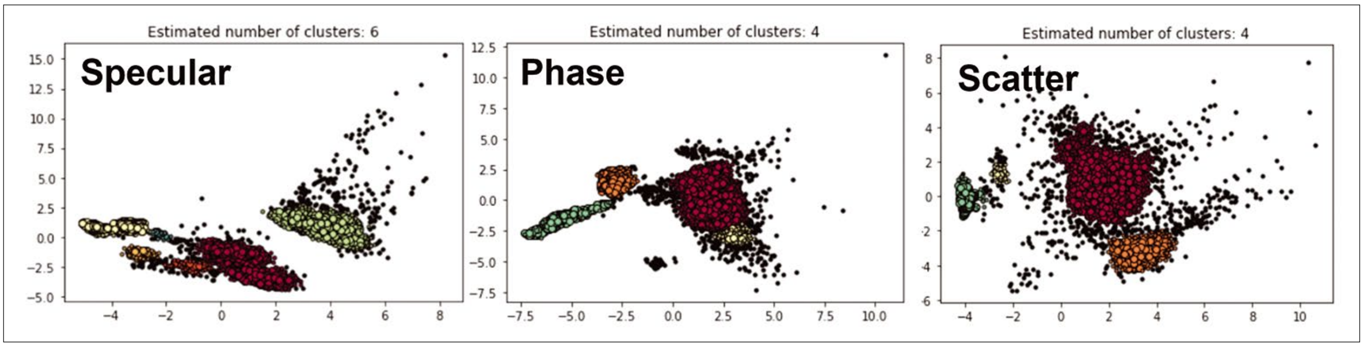


Figure 4: The PCA parameter data is clustered using DBSCAN for specular, phase, and scatter images.

I suspected much of the PCA chart structure was driven by wafer product differences. Listing 13 uses the Seaborn and Matplotlib libraries to generate a faceted chart to illustrate these differences (see Figure 5). Figure 5 shows a clear product type dependence, which can be seen in the overlap in the parameter space between the product types identified by *A*, *B*, and *C*, as well as those identified by *X*, *Y*, and *Z*.

Image Display

Because I did not start with a classification, I have nothing to compare the cluster IDs with, so I cannot construct a confusion matrix or calculate the precision or recall for this method. I can, however, visually compare the images within different cluster IDs for specific product types. To achieve this, you must identify the images within each category (the distinct clusters and the unclustered images) with the greatest distance from the center of the clusters. This first requires that the centroids be identified for each product type and detector type. Then the distance is calculated from the appropriate centroid to each individual datapoint (see Listing 14).

Once the distances from the centers are known, the most distant images can be identified for each product and detector type using the `findExtremeImgs` function in Listing 15. This function returns two DataFrames: One for the abnormal or unclustered images and one for the normal or clustered images. The number of images within each category is specified by the `mcols` value, which is set to a value of 5 in my example.

The `displayExtremeImgs` function in Listing 16 will accept the lists of extreme images and display them in a grid format.

Combining these two functions allows the extreme images within each category to be viewed simultaneously. In all cases,

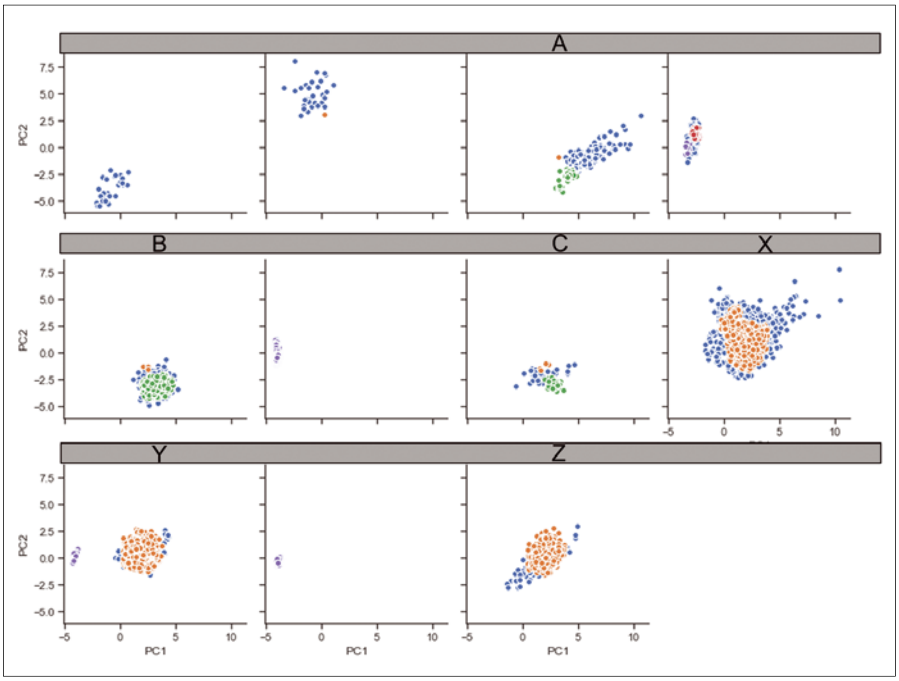


Figure 5: The faceted chart generated from the PCA feature space shows the product type dependence.

Listing 13: Faceted Plot of Cluster IDs by Product Type

```
import seaborn as sns
import matplotlib.pyplot as plt


sns.set(style="ticks", color_
        codes=True)


g = sns.FacetGrid(dfPLUS, col="PROD",
    col_wrap=4,  hue="Cluster")
g = g.map(plt.scatter, "PC1", "PC2",
    edgecolor="w")


plt.show()
```

Listing 14: Product-Detector Cluster Centers

```
import numpy as np


dfPLUS['PROD_Detect'] = dfPLUS['PROD'] + '_' + dfPLUS['Detector']
clusterCenters = dfPLUS[dfPLUS['Cluster'] > -1][['PROD_Detect', 'Cluster','PC1','PC2']].
                 groupby(['PROD_Detect']).mean()
clusterCenters.index.name = None


dfPLUS['Delta1'] = \
dfPLUS.apply(lambda x: np.round_(x.PC1-clusterCenters.loc[x.PROD_Detect].PC1, 6), axis=1)
dfPLUS['Delta2'] = \
dfPLUS.apply(lambda x: np.round_(x.PC2-clusterCenters.loc[x.PROD_Detect].PC2, 6), axis=1)


dfPLUS['Delta'] = np.sqrt(dfPLUS['Delta1']**2 + dfPLUS['Delta2']**2)
```

Listing 15: DataFrames for Normal and Abnormal Images

```
def findExtremeImgs (detectorType, prodName):

  mcols = 5
  abnormalImg = dfPLUS[(dfPLUS['Detector'] == detectorType) & (dfPLUS['PROD'] ==
                prodName) & (dfPLUS['Cluster'] < 0)]\
    [['PROD_Detect', 'Cluster','PC1','PC2','Delta','FILENAME']].\
    sort_values(['Delta'], ascending=False).\
    groupby(['PROD_Detect','Cluster']).\
      head(mcols)

  normalImg = dfPLUS[(dfPLUS['Detector'] == detectorType) & (dfPLUS['PROD'] ==
              prodName) & (dfPLUS['Cluster'] >= 0)]\
    [['PROD_Detect', 'Cluster','PC1','PC2','Delta','FILENAME']].\
    sort_values(['Delta'], ascending=False).\
    groupby(['PROD_Detect','Cluster']).\
      head(mcols)

  return abnormalImg, normalImg
```

Listing 16: Grid Display of Normal and Abnormal Images

```
def displayExtremeImgs (detectorType, thisProd, abnormalImg, normalImg):

  thisCols = abnormalImg['Cluster'].size
  clusterIDlist = pd.DataFrame(np.concatenate((np.unique(abnormalImg['Cluster']),
                  np.unique(normalImg['Cluster']))))
  thisRows = clusterIDlist.size

  row_labels = ['CLUSTER_ID {}'.format(row) for row in iter(clusterIDlist.loc[:,0])]
  figure, axes = plt.subplots(nrows=thisRows, ncols=thisCols,
figsize=(15,(thisRows*1.75)))
  for ax, row in zip(axes[:,2], row_labels):
    ax.set_title(row,fontweight="bold")
    ax.set_xticks([])
    ax.set_yticks([])

  figure.set_facecolor('w')

  col=0

  for i, ind in enumerate (abnormalImg.index):
    imgURI = web_config.imgServerRoot + abnormalImg.loc[ind]['FILENAME']
    img = get_URL_img(imgURI)
    axes[0, col].imshow(img)
    col = col+1

  for i, ind in enumerate (normalImg.index):
    imgURI = web_config.imgServerRoot + normalImg.loc[ind]['FILENAME']
    img = get_URL_img(imgURI)
    thisCluVal = np.int0(normalImg.loc[ind]['Cluster'])

    thisClu = np.int0(np.where(clusterIDlist == thisCluVal))
    axes[1+(i // thisCols),(i % thisCols)].imshow(img)


  plt.setp(plt.gcf().get_axes(), xticks=[], yticks=[]);
  plt.show()
```

the first row of images displays those images that were not clustered (cluster ID -1.0) and can be considered abnormal. Subsequent rows show the most extreme images for all additional clusters that were observed in the dataset.

In examining the image galleries in Figures 6-8, you can see that it is indeed possible to separate abnormal bevel images from those without any distinguishing features. Perhaps unusually, the abnormal scatter images show a thinner ridge at the wafer edge (see Figure 7). In the phase images shown in Figure 8, the unclustered or abnormal images appear to be a sub-category of the cluster ID 0.0 images, but ones where the defect feature intersects with the image boundary. In this case, cluster ID 0.0 and cluster ID -1.0 should both be considered abnormal.

Although you can clearly see differences between the image clusters, it is possible that every silicon wafer has the same distribution of cluster IDs, which provide no ability to differentiate between wafers. To investigate this, stacked bar charts were plotted for each product-detector combination, where counts were grouped by wafer, using the command in Listing 17 (see Figure 9).

The specular images in Figure 9 show little variation between wafers. The phase images show more interesting behavior with a third cluster ID appearing on a selection of wafers. And perhaps unsurprisingly, the scatter images show an elevated level of scatter.

Differences between products (wafers) can be observed; what remains to be seen is if these differences are useful indicators of yield or reliability. Additionally, the significance of the differences could be determined by using Fisher's exact test [13]. The possibility of individual cluster IDs approaching zero observations makes the application of a Chi-square test [14] inappropriate in this case.

Conclusion

It is certainly possible to segregate abnormal bevel images from those that can be considered normal by means of unsupervised machine learning. This approach is not necessarily appropriate for all image classification applications, but this does not mean that you should overlook it. As always, you should aim to employ the simplest solution where possible. In certain cases, the extraction of texture and form from images can provide a simple solution in conjunction with PCA and DBSCAN.

Unsupervised machine learning bypasses the train, validate, and test cycle. However, the end result in this unsupervised machine learning approach is not a model that can be applied to new images to generate a cluster ID. Instead, the feature extraction, standardization, and PCA generate a set of coordinates that can be compared with the parameter space obtained from the historical data by means of the k-nearest neighbors algorithm [15]. This process provides a cluster ID for each new image. ■■■

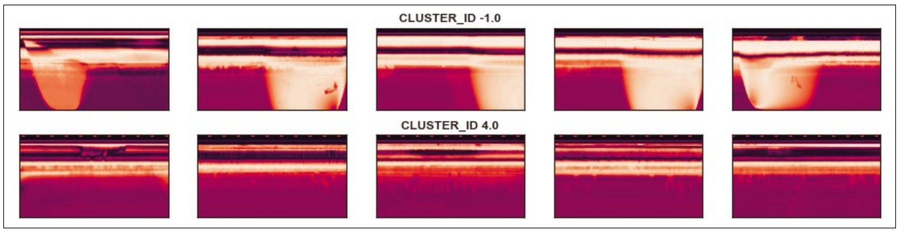


Figure 6: Specular images (abnormal and normal bevels) for product *X*.

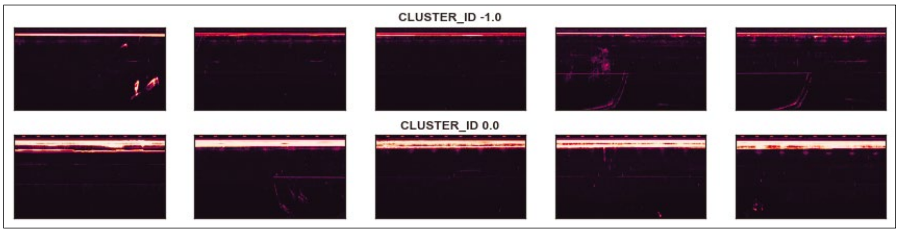


Figure 7: Scatter images (abnormal and normal bevels) for product *X*.

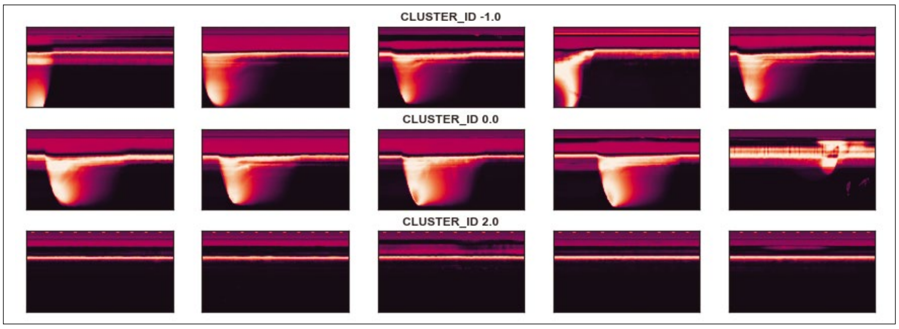


Figure 8: Phase images (abnormal and normal bevels) for product *X*.

Listing 17: Cluster ID Stacked Bar Chart

```
dfPLUS[(dfPLUS['Detector'] == thisDetector) & (dfPLUS['PROD'] == thisProd)]
                              [['Cluster','WAF','INSP_TIME']].\
  groupby(['WAF','INSP_TIME','Cluster']).\
      size().unstack().\
  plot(sort_columns='INSP_TIME',kind='bar', stacked=True)
```

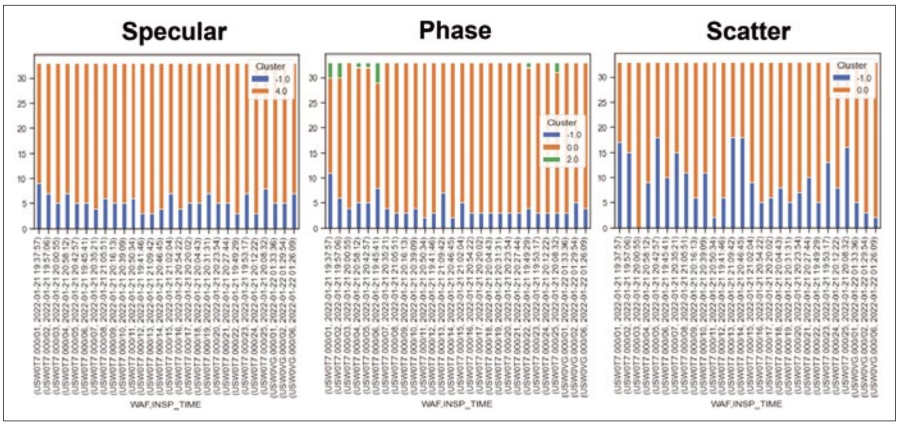


Figure 9: Stacked bar charts are plotted by product and image type for product *X*.

Info

[1] Smyth, Padhraic, et al. "Knowledge Discovery in Large Image Databases: Dealing with Uncertainties in Ground Truth." *In: Advances in Knowledge Discovery and Data Mining* (AAAI/MIT Press, Menlo Park, CA, 1995), pp. 109-120

[2] cx_Oracle: *https://cx-oracle.readthedocs.io*

[3] pandas: *https://pandas.pydata.org/pandas-docs/stable/getting_started/index.html*

[4] PIL: *https://python-pillow.org/*

[5] cv2: *https://pypi.org/project/opencv-python/*

[6] mahotas: *https://pypi.org/project/mahotas/*

[7] NumPy: *https://numpy.org/doc/stable/user/quickstart.html*

[8] Merging methods: *https://pandas.pydata.org/docs/user_guide/merging.html*

[9] sklearn StandardScaler: *https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html*

[10] sklearn PCA: *https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html*

[11] "Indoor Navigation with Machine Learning" by Roland Pleger, *Linux Magazine*, issue 255, February 2022, *https://www.linux-magazine.com/index.php/Issues/2022/255/Machine-Learning*

[12] sklearn DBSCAN: *https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html*

[13] Fisher's exact test: *https://mathworld.wolfram.com/FishersExactTest.html*

[14] Chi-square test: *https://www.jmp.com/en_be/statistics-knowledge-portal/chi-square-test.html*

[15] k-nearest neighbor algorithm: *https://www.ibm.com/topics/knn*

Author

Garry Tuohy has been surviving in the semiconductor industry for longer than a radiation-hardened processor. He also enjoys astronomy and dreaming of fanciful uses for microcontrollers.

Rocky Linux

Replacing CentOS

Rocky Linux steps into the breach left by CentOS with a community-based alternative to RHEL. *By Bruce Byfield*

Red Hat Linux bought CentOS in 2014, but largely ignored it for years. Essentially, CentOS was the community version of Red Hat Enterprise Linux (RHEL), both deriving from Fedora. Then, in December 2020, Red Hat announced that CentOS would be discontinued, and it would be replaced by CentOS Stream. Within days, Rocky Linux (Figure 1), named for CentOS cofounder Rocky McGaugh, was announced [1]. Four months later, Rocky Linux released its first version. Like CentOS, it offers a community-based alternative to Red Hat.

Recently, I sent questions about Rocky's current state to Brian Clemens, the project manager at the Rocky Enterprise Software Foundation (RESF), who got the development team to collaborate on answers. Here's what the RESF development team has to say about Rocky Linux.

Linux Magazine (LM): What were the relations between Red Hat and CentOS leading up to the discontinuation of CentOS?

Rocky Linux (RL): There were no formal relations or communication between Red Hat, CentOS, and future members of the Rocky Linux project/RESF prior to the discontinuation of CentOS. We were all surprised. Gregory Kurtzer, one of the original CentOS founders, announced an effort to build a replacement project within hours of the "CentOS Project shifts focus" post.

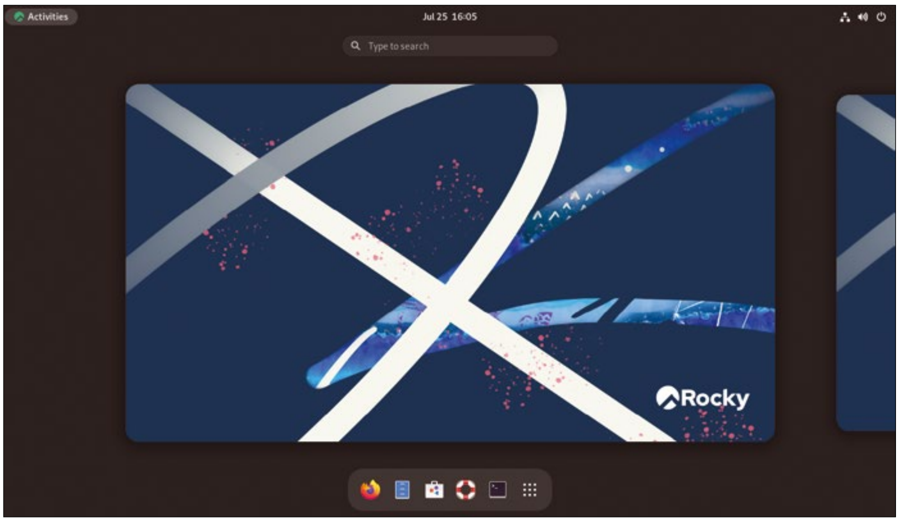


Figure 1: Rocky Linux is one of the main CentOS replacements.

LM: What were the goals behind starting Rocky Linux?

RL: The primary goal of the Rocky Linux project is to produce a free, community-driven, bug-for-bug rebuild of RHEL. Imperatively, we are dedicated to operating in an open, transparent, and reproducible manner that embraces community participation and contribution. We are committed to enriching the entire Enterprise Linux ecosystem with open build processes and tools, beyond just adding another distribution to the mix.

LM: How does Rocky Linux compare to CentOS and to other spin-off Enterprise Linux distributions in terms of goals, features, and popularity?

RL: Aside from our previously mentioned goals, a lot of effort was put into ensuring that what happened to CentOS would not happen to Rocky Linux. To do this, we had to build a resilient organizational structure with proper checks and balances, as well as a provider-agnostic infrastructure that doesn't depend on the whims of a partner or sponsor.

The infrastructure for Rocky Linux was built from scratch. Special care was taken to ensure no opportunity for control by an individual or company. All the development requirements we leverage are open source and freely available. We refuse to release any software unless it can be reproduced with free and open tools and processes. We leverage technologies such as distributed key generation to achieve community ownership and control of our security

Photo by Peter Pryharski on Unsplash

Table 1: Rocky Linux 9 ISO Downloads by Date

Date	ISOs	Torrents
14/Jul/2022	4,899	1,034
15/Jul/2022	21,896	2,496
16/Jul/2022	9,036	1,407
17/Jul/2022	7,176	1,172
18/Jul/2022	11,719	1,448
19/Jul/2022	22,892	1,307
20/Jul/2022	11,382	1,241
21/Jul/2022	9,092	984
94,227 downloads of Rocky Linux 9 as of 2022-07-21		
1,000,729 downloads of Rocky Linux 8 as of 2022-07-21		

Table 2: Community Statistics

7,763 members on `chat.rockylinux.org`
~3,600 members on `forums.rockylinux.org`
251 members on IRC
1,319 members in the main development channel (`chat.rockylinux.org` and IRC)
1,182 members in the testing channel
~140 unique committers in GitHub

artifacts (signing keys, secure boot shims, etc.).

However, popularity can be a difficult metric to measure, and we don't see this as a popularity contest. Competition and choice is healthy for the overall advancement of open source. It's a good thing that multiple projects exist to solve similar problems for various communities and use cases. We prefer to let our community speak for Rocky Linux.

LM: Describe the target audience for Rocky Linux.

RL: Anyone and everyone who needs a stable, trusted, and transparent operating system for servers, appliances, cloud instances, containers, virtual machines, edge devices, compute nodes, and/or simply wants a reliable distribution for their home lab.

LM: Can you provide any stats about downloads, commits, and developers?

RL: See Table 1 for downloads and usage and Table 2 for community statistics.

I've also included some charts (Figures 2 and 3) that indicate the relative usage of Rocky Linux. It's important to note in these charts that they do not represent the total usage of the distributions listed, as they are derived from usage statistics of the Extra Packages for Enterprise Linux (EPEL) repository, which not all users will enable. Still, an approximately equal proportion of users of each distribution will enable it, making it useful for determining the relative usages of each distribution. The ephemeral (< 1 week old instance) usage graph is highly variable (Figure 3), as it can be majorly swayed by large numbers of short-lived instances like supercomputing jobs, CI/CD systems, etc.

LM: How is Rocky Linux organized and governed?

RL: The roots of our governance model are based on how we developed, which is a pure meritocracy. After the announcement of the project, thousands of developers, contributors, and community members all joined a Slack instance around HPCng (the next generation of HPC).

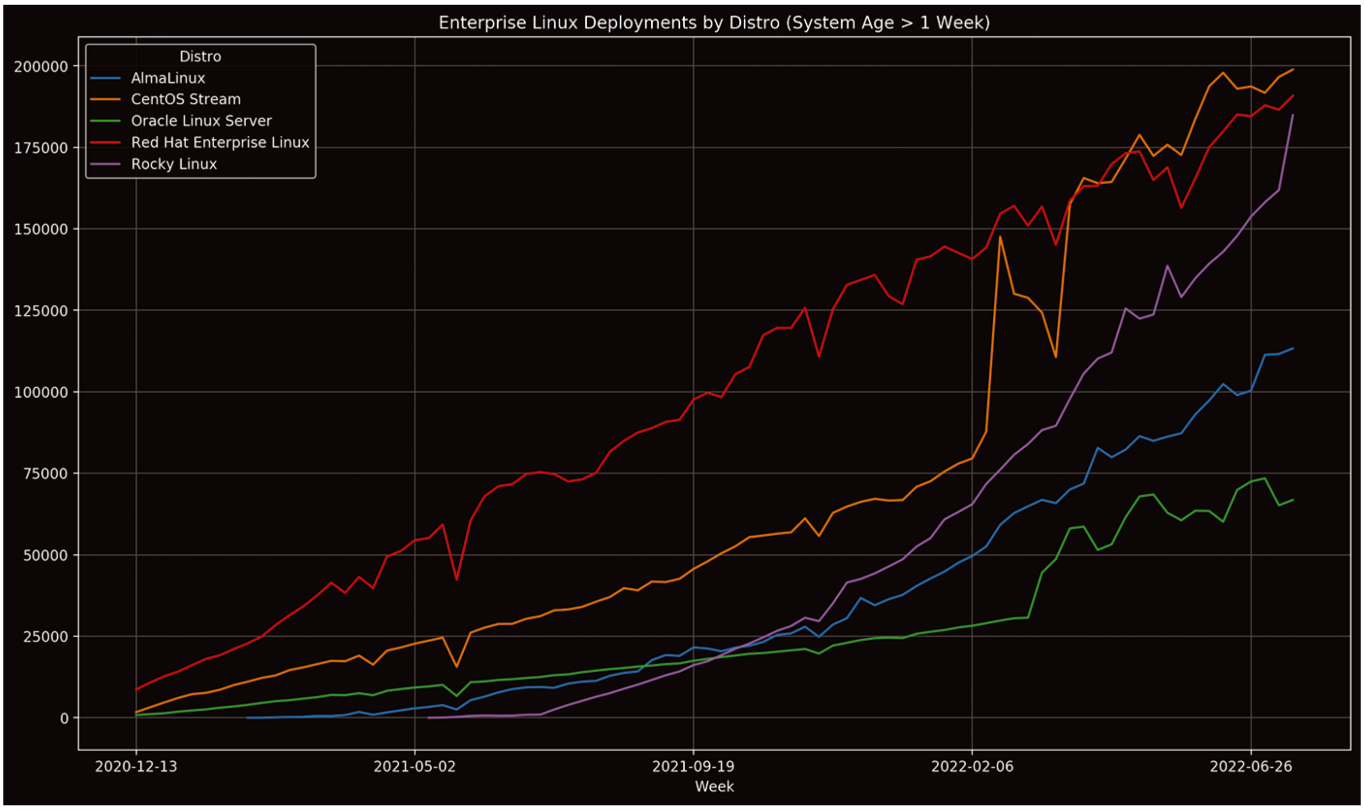


Figure 2: Long-term statistics from the EPEL repository show Rocky Linux as a contender in the Fedora/Red Hat family.

With so many people joining the Slack community, it was unmanageable initially. Thousands of people were messaging Gregory Kurtzer [the creator of CentOS and of Rocky Linux] directly! So the community was broken into channels for different areas of what we needed to create. People joined the channels that they were most interested in and qualified for. From there, cooperation began, and the community began to develop. Within each channel, people started to emerge as natural leaders for each team. This occurred with people not even knowing anything about anyone else. Nobody checked where people worked, qualifications, gender, race, religion, or beliefs, and most people were using aliases with cartoon avatars. People got where they are within the Rocky organization based on their merit and contribution to the project alone.

These team leads became the controlling structure for Rocky Linux.

We then created the Rocky Enterprise Software Foundation (RESF) as the legal entity behind Rocky. We decided to go with a public benefit corporation (PBC) as it provides more flexibility than a 501(c) organization, which will allow us to better protect the project.

Long-term protection of the RESF and its projects is imperative to many in our community who were involved in projects with seemingly safe structures that shifted away from their community's wishes. For example, CentOS was founded as part of a 501(c)(3) nonprofit organization. Other forms of the 501(c) do not offer any more protection. For example, a 501(c)(6)'s purpose is to promote the common business interest of its sponsor (see 1.501(c)(6) in [2]).

RESF projects are composed of diverse sets of project members and backed by large communities of individuals and organizations all working together to serve their communities. Thus we needed a structure that we felt best protects and serves these communities.

This is why the integrity, accountability, and transparency of the people involved in the project are the most critical aspects for determining the long-term sustainability and viability of any project. We believe that the RESF mitigates the risk of diversions from community goals and takeover by external entities by implementing a series of checks and balances.

The structure that we created for the RESF is forged by the lessons learned from the many projects and organizations that have preceded us and will continue to evolve to meet the needs of the future. Information on our current structure can be found at [3].

LM: What features make Rocky Linux stand out, especially for security and networking?

RL: Rocky Linux is a bug-for-bug compatible and freely available implementation of RHEL. Beyond compatibility and feature parity, we are adding value with various security accreditations like FIPS, STIGs, Common Criteria, and others forthcoming.

We can't stress enough how significant it is to have a freely available open source operating system pursuing a Federal Information Processing Standard Publication 140-3 (FIPS-140-3) certification for our cryptographic modules. Rocky Linux will be the first community-driven operating system to have this, and this means that all the cryptographic modules included in Rocky will have been verified and certified by independent accredited organizations like the National Institute of Standards and Technology (NIST). We are also pursuing an Evaluation Assurance Level (EAL4+) certification, which

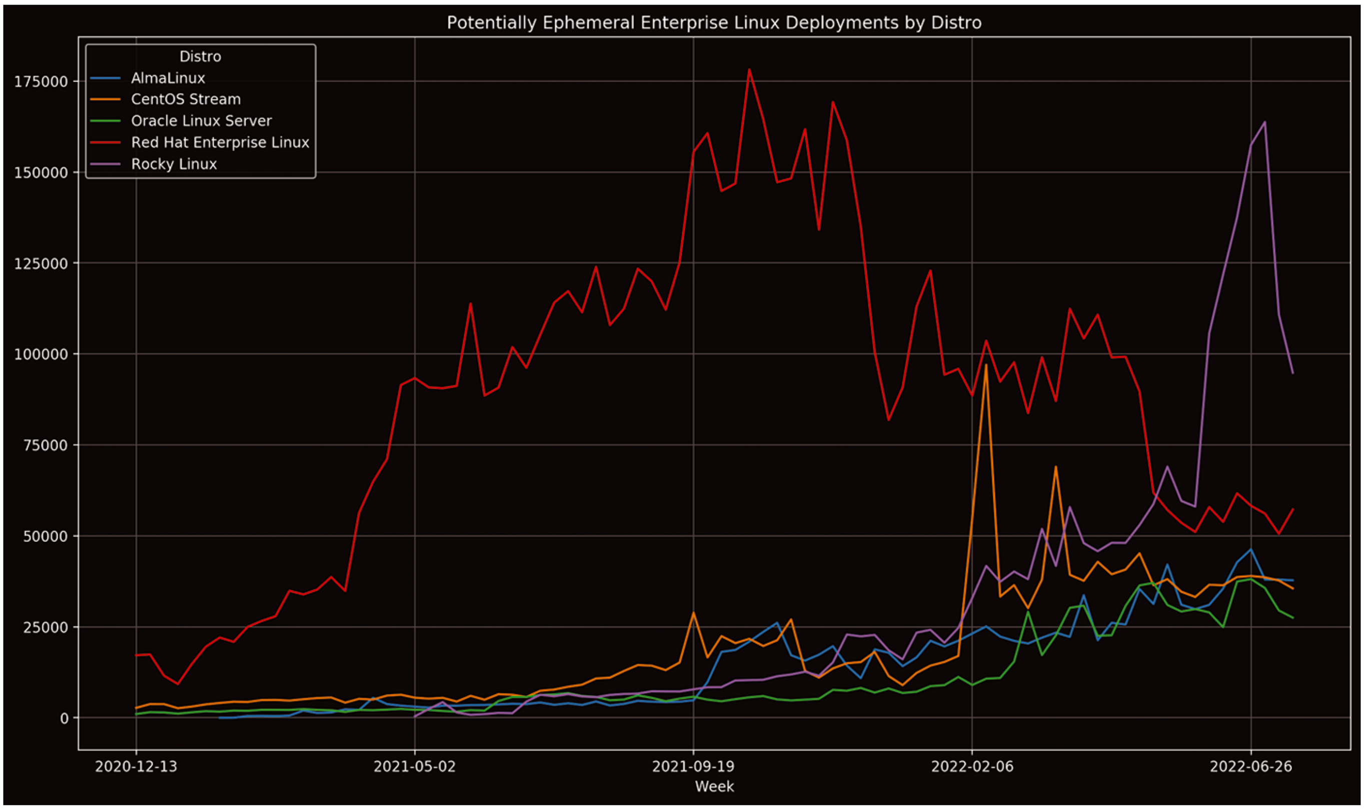


Figure 3: While highly variable, short-term statistics from the EPEL repository show Rocky rivalling and at times surpassing RHEL and CentOS Stream.

is again the first time a community Linux distribution will have achieved this level of validation. If compliance and verification matters to you, then these are all huge advantages to using Rocky.

LM: How does Rocky Linux differ from RHEL and other CentOS replacements?

RL: We've differentiated ourselves further with our open build infrastructure, which is managed via many members of our community. This infrastructure was built by the community for the Rocky Linux project. There is no reliance on specific individuals or companies, and all the infrastructure we use is owned by the project.

This includes not only the infrastructure, but the cryptographic material for signing packages, access to infrastructure, and our secure boot shim. Our secure boot shim was built for and is owned by the Rocky project itself.

Going forward, we want to welcome other security professionals and organizations to participate in Rocky Linux Special Interest Groups [SIGs] to develop tools, guides, and standards around Rocky Linux. To date, we have already worked with Center for Internet Security (CIS) [4] to add security benchmarks for Rocky Linux 8, and SCAP definitions with OpenSCAP [5]. We have also coordinated with commercial vendors to add vulnerability scanning and compliance testing support for Rocky Linux and written several of our own guides for supporting DISA STIG [6] and PCI DSS compliance guidelines. We'd like to continue that effort and add support for more open and commercial tools to serve the Rocky Linux community.

From end to end, all engineering, development, testing, and collaboration are done openly and transparently. We share all of our tools, and we seek independent organizations like NIST and CIS to validate our methodology wherever we can. This is one of the reasons why our Mattermost chat and IRC community are so large and inclusive; everyone can take part and be part of the team.

LM: The Peridot build system is new in Rocky Linux 9. Why is it important?

RL: Rocky Linux has never been built in a black box. Every version of Rocky Linux, as an open source and freely available operating system, has always included everything someone would need to replicate the entire project. This is what it means to be community first and it is our unwavering commitment to a collaborative open source project.

For Rocky Linux 8, we used Koji, Fedora's build system, but for version 9, we decided we needed something easier for others to adopt, cloud-native, and something that would fit better into our goals of including Special Interest Groups and community contributions to the operating system. For that, we built Peridot.

Peridot is a cloud-native build system built upon a strong foundation of separation, security, ease of use, scaling, and modern architectures like microservices. Peridot can be used to not only build an RPM-based distribution, but it's also fully capable of forking a distribution and optimizing long term maintenance. Additionally, it enables third-party contributors to extend the base distribution in the form of SIGs and extensions without compromising compatibility or our trusted artifacts (e.g., signing keys). Providing access to volunteers and contributors in a secure and trusted manner to build packages for Rocky Linux has been a challenge with our previous build system, and Peridot solves this.

As an Enterprise Linux operating system provider, we have to be fast and correct on releasing security updates and timely on all updates. Peridot allows us to distribute build tasks to multiple build sites and can automatically scale up workers in Kubernetes to handle pressure as well as scale down when needed. After the package builds have completed, we still need to get these out to our users, and to make that process quicker, as well as build a proper microservice platform, it was required to operate statelessly and move beyond NFS-based on-disk repository generation and updates, so we also built Yumrepofs.

Yumrepofs doesn't require a persistent host-based state for managing a YUM repository by swapping in and out necessary RPMs from a specific build or a configuration change on demand. This allows us to always have our updated repositories ready to release.

This also optimizes security by tamper-proofing the resulting build, artifacts, logs, and metadata by signing all of them, which guarantees provenance and trust of the entire operating system after Peridot has completed the builds.

We are so excited about Peridot because it:

- allows us to easily maintain multiple major versions of distribution forks/rebuilds
- allows community members to securely engage with SIGs
- enables multiple security measures to prevent private signing key breaching [and] package, artifact, and repository tampering
- [and offers] unprecedented levels of operating system build transparency, all while enabling engineers to push out updates in record speed.

LM: Where is Rocky Linux heading in the future?

RL: More community involvement via packaging, enhancements, SIGs, testing, and bug reports all ensure the long-term success of our community and users.

We continue to grow our community of sponsors and partners, with many amazing organizations eager to be a part of what we are doing.

We are currently codifying the formal board structure; more changes will be recognized in terms of decision making processes and leadership.

LM: Where can people volunteer to get involved?

RL: Everyone is welcome! We are a community of super-friendly people, and we have lots of opportunities for more people to contribute and take leadership roles. We actively seek leaders from within our community, and we are looking for more people interested in both the technical and nontechnical aspects of running a community. We invite all to help regardless of skill level. Join us on IRC or Mattermost at *https://chat.rockylinux.org*! ■■■

Info

[1] Rocky Linux: *https://rockylinux.org/*
[2] IRC 501(c)(6) organizations: *https://www.irs.gov/pub/irs-tege/eotopick03.pdf*
[3] Rocky Linux structure: *https://rockylinux.org/organizational-structure/*
[4] CIS: *www.cisecurity.org*
[5] OpenSCAP: *www.openscap.org*
[6] DISA STIG: *www.disa.mil*

Simplifying data backup on Android devices

Backup Assistant

Linux Android Backup offers an open source solution to backing up the most important content from your Android devices in just a few steps. *By Ferdinand Thommes*

Backing up data from a Linux installation doesn't pose a problem for most users thanks to open source solutions for almost every scenario, from a single machine to a cluster, as well as command line and GUI options. However, backing up Android devices such as smartphones or tablets can be a little trickier. Up until Android 4, backing up such devices was only possible if the bootloader was unlocked, and unlocking was no easy task because most manufacturers tried to prevent it. And if you did unlock the bootloader, you could void your warranty, and, even worse, risk bricking your device. An advance backup would have made sense, but it was only possible after rooting.

Starting with Android 4, `adb backup` [1] (which Google has since deprecated for security reasons) worked reliably without rooting, even between devices by different manufacturers, which differed from many other manufacturer tools or third-party apps. Today, the only official tool left for backups is Android Debug Bridge (ADB), which lets you control Android devices using a computer with Linux, macOS, or Windows. However, using ADB requires many manual steps and still comes with the risk of bricking the device.

Linux Android Backup (LAB), a fledgling project developed on GitHub, offers an alternative to ADB [2]. Under the hood, LAB, which consists of a small shell script and a Flutter app, greatly simplifies backing up Android devices. LAB works entirely without vendor lock-in or closed source software that could compromise your data. While the script is based on ADB, it does not use the deprecated `adb backup` command. Despite its name, LAB also works on macOS and Windows, in the Windows Subsystem for Linux (WSL) [3]. In fact, the developers are currently looking for a more suitable name.

Keeping It Simple

LAB aims to keep the backup process for Android devices to just a few steps with minimal interaction with the user. To take advantage of this simplification, you will need to prepare your device up front. However, once you launch the script, LAB requires virtually no intervention. LAB currently backs up installed apps (in APK file format [4]), photos, videos, and other downloads on your device. In contrast to the manual procedure via ADB, LAB also backs up saved contacts in vCard format by installing a small open source helper app. You can submit suggestions for further data to be backed up as an issue on GitHub [2].

The backed up data can be accessed as encrypted archives compressed to the max by 7-Zip; these archives remain accessible even if the script stops working. For this, you will need to install the *p7zip-full*, *adb*, *curl*, and *bc* packages on your computer. Listing 1 shows the commands for doing this on Debian and Fedora.

Developer Mode

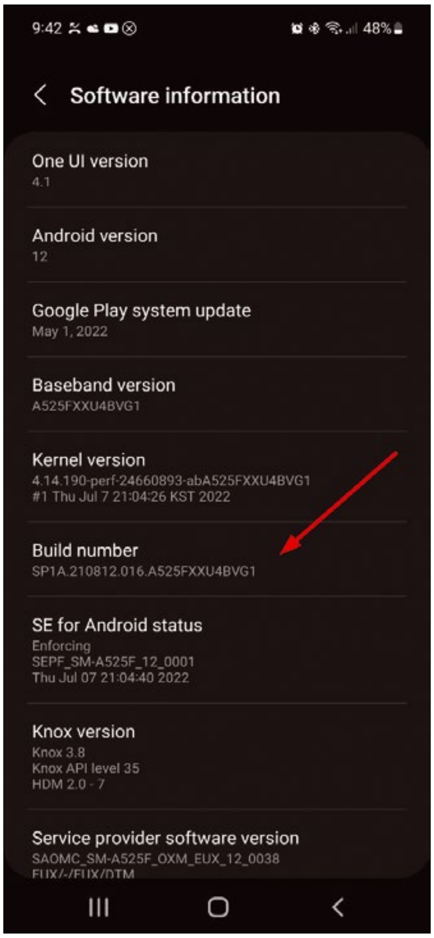
On your Android device, you now need to unlock developer mode. To do this, go to *About the phone* at the very bottom of

Listing 1: Installing Dependencies

```
### Debian
$ sudo apt update && sudo apt install p7zip-full adb curl bc
### Fedora
$ dnf update && dnf install p7zip p7zip-plugins adb curl bc
```

Photo by Jandira Sonnendeck on Unsplash

the settings, tap the *Build number* entry seven times (Figure 1), and then enter your PIN when prompted. The *You are now a developer* message flashes up briefly on the display. To copy data between the Android device and your computer or laptop, you also need to enable *USB debugging* (Figure 2). You will find this option hidden in the settings below *System & Updates* | *Developer options*.



9:42 48%

Software information

One UI version
4.1

Android version
12

Google Play system update
May 1, 2022

Baseband version
A525FXXU4BVG1

Kernel version
4.14.190-perf-24660893-abA525FXXU4BVG1
#1 Thu Jul 7 21:04:26 KST 2022

Build number
SP1A.210812.016.A525FXXU4BVG1

SE for Android status
Enforcing
SEPF_SM-A525F_12_0001
Thu Jul 07 21:04:40 2022

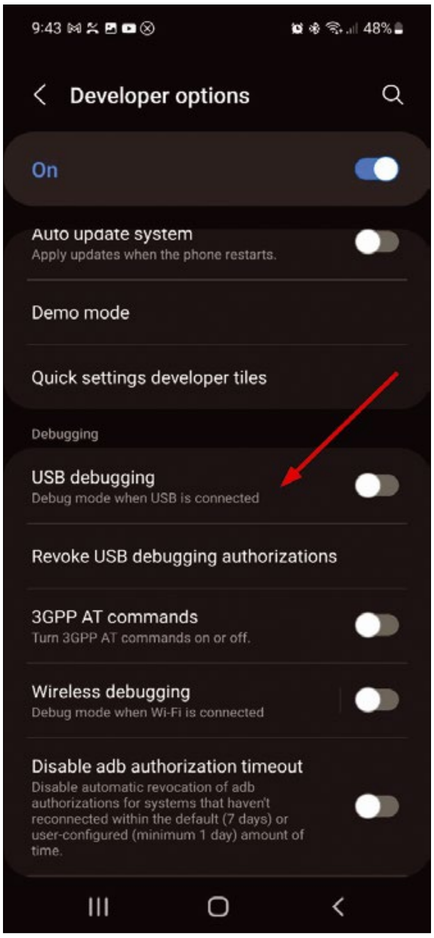
Knox version
Knox 3.8
Knox API level 35
HDM 2.0 - 7

Service provider software version
SAOMC_SM-A525F_OXM_EUX_12_0038
EUX/-/EUX/DTM

Figure 1: You must unlock developer mode on your Android device before you can exchange data between the device and your computer.

When done, your Android device is ready for backup.

On the PC or laptop where you plan to save your device's backup files, first download LAB [5], unpack the archive, and change to the unpacked directory at the command line. Make sure that your Android device is properly connected to the computer. A prompt then pops up on the connected Android device, asking you whether you want to load or copy data. Select *Copy*. After pressing *Enter* the connected device should be detected and tagged as *device*. If this doesn't happen, press Ctrl + C to cancel the script, check the connection, and restart the script (Figure 3).



9:43 48%

Developer options

On

Auto update system
Apply updates when the phone restarts.

Demo mode

Quick settings developer tiles

Debugging

USB debugging
Debug mode when USB is connected

Revoke USB debugging authorizations

3GPP AT commands
Turn 3GPP AT commands on or off.

Wireless debugging
Debug mode when Wi-Fi is connected

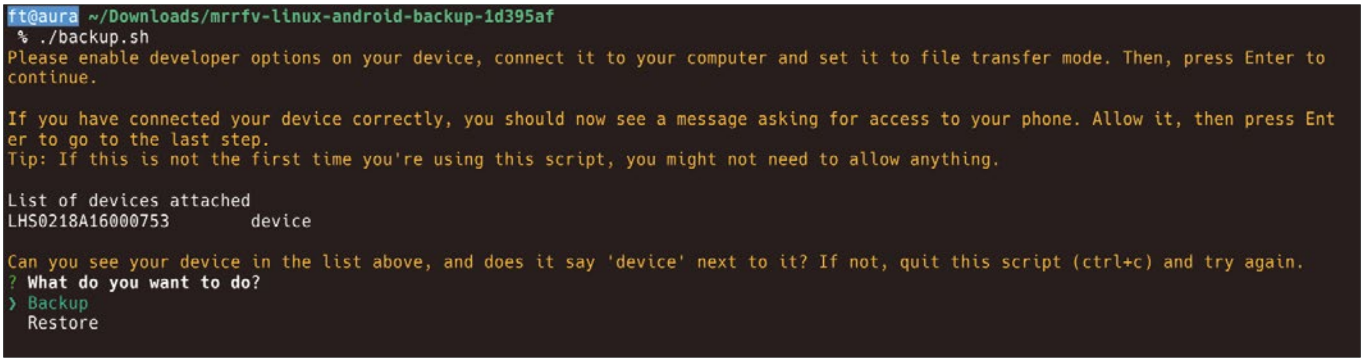
Disable adb authorization timeout
Disable automatic revocation of adb authorizations for systems that haven't reconnected within the default (7 days) or user-configured (minimum 1 day) amount of time.

Figure 2: The *USB Debugging* function can be hidden in various places in your Android device's settings. If you don't find it here, search the Internet with your device name to determine the exact location of this option.

Once your device is successfully detected, you will then see the two operating modes: *Backup* and *Restore* (*Backup* will be selected by default). By pressing Enter, you will install the helper app for backing up your contacts from the Android device. The script output (Figure 4) suggests the app has been opened on the smartphone, but this was not true in my test. After manually opening the *Linux Android Backup* app on your device, you will see an *Export Data* button, which lets you copy your contacts.

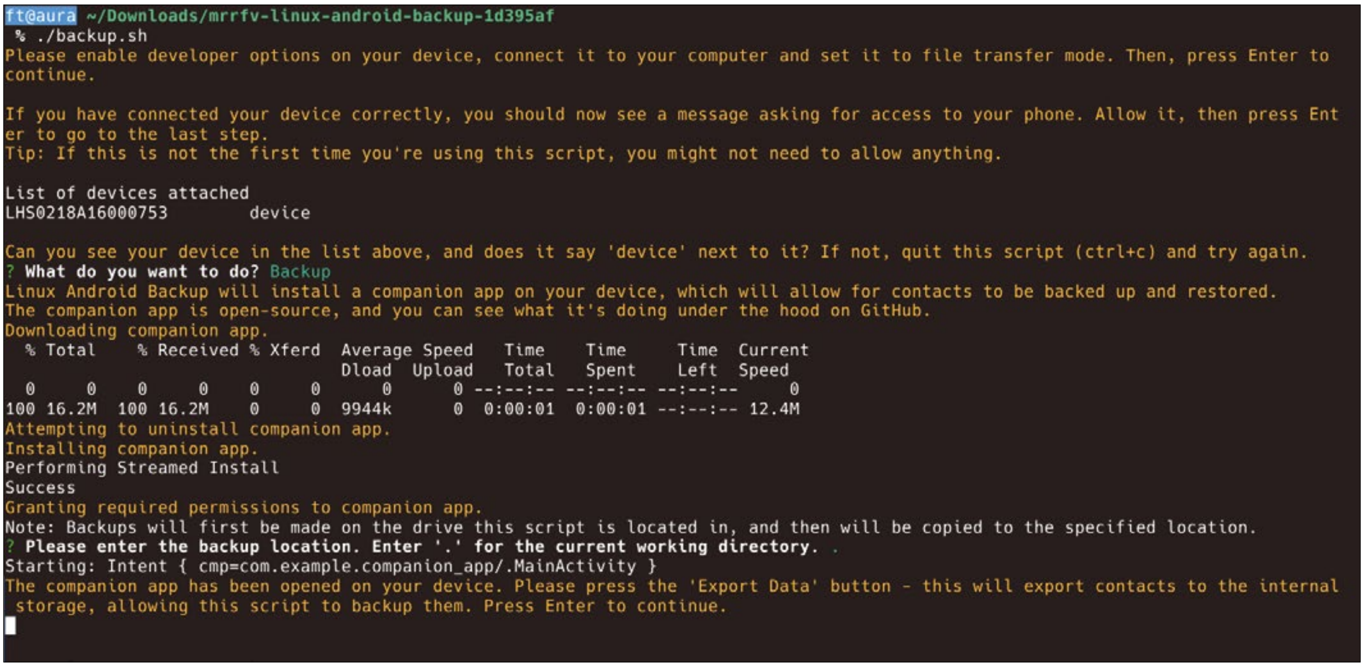
Next, you need to specify the directory where you want LAB to copy the backup data. For the current directory, simply press *Enter* and the data backup from Android to Linux now begins. Depending on the volume of data to be copied, this can take a few minutes. You can follow the progress in detail on the screen. In my first test with a Huawei Mate 20 Pro, transferring over 16,000 files with a total size of around 18GB took about eight minutes.

Finally, you need to set a password for the encrypted archive; afterward, LAB will create the encrypted archive within a few minutes. The entire process including downloading the app and configuration took about 30 minutes in my test. If successful, LAB will tell you *Everything is Ok* (Figure 5). The resulting archive from the Huawei Mate 20 Pro was 13GB. Make sure you save your password; without it you will not be able to access your data again. Don't forget to revert the settings you configured for USB Debugging and Developer



```
ft@aura ~/Downloads/mrrfv-linux-android-backup-1d395af
 % ./backup.sh
Please enable developer options on your device, connect it to your computer and set it to file transfer mode. Then, press Enter to
continue.

If you have connected your device correctly, you should now see a message asking for access to your phone. Allow it, then press Ent
er to go to the last step.
Tip: If this is not the first time you're using this script, you might not need to allow anything.

List of devices attached
LHS0218A16000753        device

Can you see your device in the list above, and does it say 'device' next to it? If not, quit this script (ctrl+c) and try again.
? What do you want to do?
❯ Backup
  Restore
```

Figure 3: After starting the script, the Android device is first detected and tagged as *device*. If this does not work, make sure your device is properly connected.



```
ft@aura ~/Downloads/mrrfv-linux-android-backup-1d395af
 % ./backup.sh
Please enable developer options on your device, connect it to your computer and set it to file transfer mode. Then, press Enter to
continue.

If you have connected your device correctly, you should now see a message asking for access to your phone. Allow it, then press Ent
er to go to the last step.
Tip: If this is not the first time you're using this script, you might not need to allow anything.

List of devices attached
LHS0218A16000753        device

Can you see your device in the list above, and does it say 'device' next to it? If not, quit this script (ctrl+c) and try again.
? What do you want to do? Backup
Linux Android Backup will install a companion app on your device, which will allow for contacts to be backed up and restored.
The companion app is open-source, and you can see what it's doing under the hood on GitHub.
Downloading companion app.
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 16.2M  100 16.2M    0     0  9944k      0  0:00:01  0:00:01 --:--:-- 12.4M
Attempting to uninstall companion app.
Installing companion app.
Performing Streamed Install
Success
Granting required permissions to companion app.
Note: Backups will first be made on the drive this script is located in, and then will be copied to the specified location.
? Please enter the backup location. Enter '.' for the current working directory. .
Starting: Intent { cmp=com.example.companion_app/.MainActivity }
The companion app has been opened on your device. Please press the 'Export Data' button - this will export contacts to the internal
 storage, allowing this script to backup them. Press Enter to continue.
```

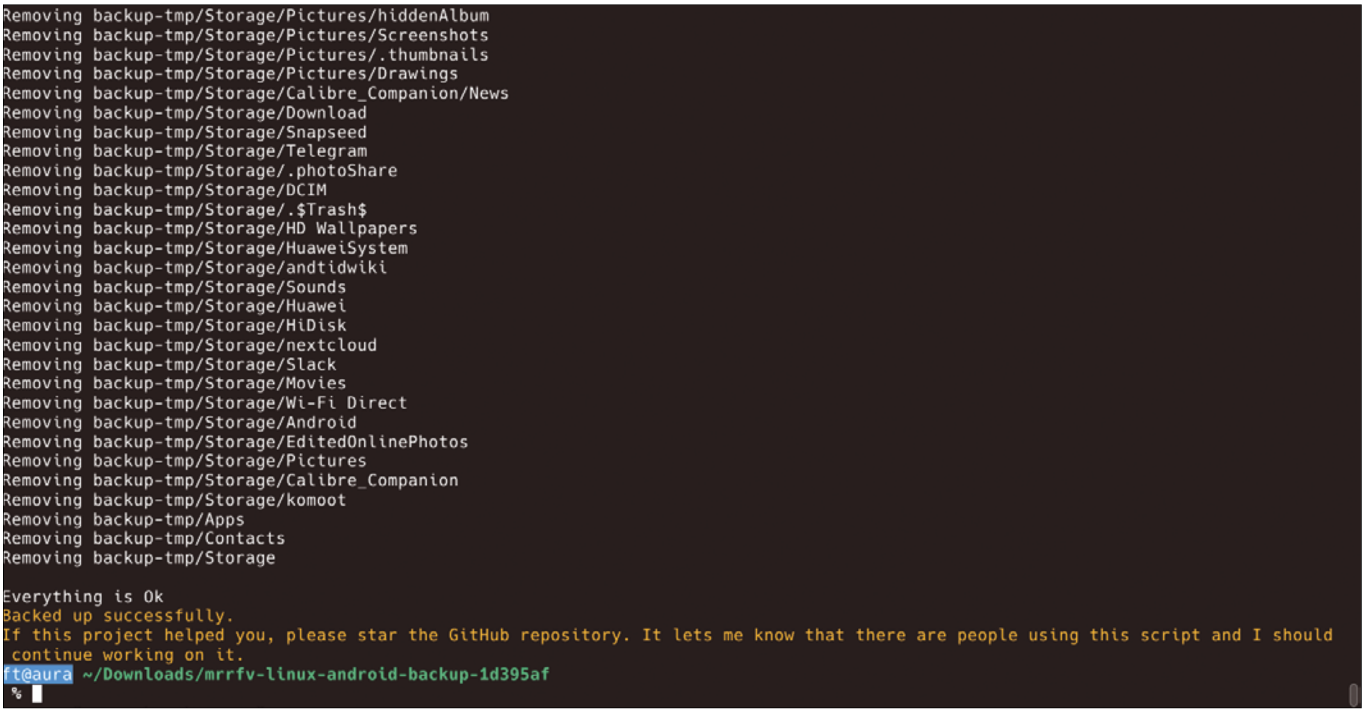
Figure 4: After confirming the backup, the helper app downloads and installs if it is not already present.

Mode, as they give more rights than you need in everyday use.

I also repeated this procedure with the new Murena One [6] smartphone, which uses /e/OS as its operating system. /e/OS is based on LineageOS [7], an Android custom ROM based on the Android Open Source Project (AOSP) [8]. I wanted to ensure that LAB also works with custom ROMs. Unlocking developer mode on the Murena One was the same as on the Huawei, but USB debugging was named *Android-Debugging* on the Murena One and resided in *System* | *Advanced* | *Developer Options*. Again, if you do not find the setting as described here, your best bet is to search the Internet for the specifics for your device.

Restore

Restoring data works the same way as backing up data. However, after the device is detected, do not use the default *Backup* option. Instead select *Restore* using the down arrow (Figure 6). First, the helper app should open on the Android device; this did not work on the Murena One either. Instead, I had to open the app manually and this time press *Auto-restore contacts*. After doing so, I only needed to specify the backup archive's location. You could also insert the archive into the script running in the



```
Removing backup-tmp/Storage/Pictures/hiddenAlbum
Removing backup-tmp/Storage/Pictures/Screenshots
Removing backup-tmp/Storage/Pictures/.thumbnails
Removing backup-tmp/Storage/Pictures/Drawings
Removing backup-tmp/Storage/Calibre_Companion/News
Removing backup-tmp/Storage/Download
Removing backup-tmp/Storage/Snapseed
Removing backup-tmp/Storage/Telegram
Removing backup-tmp/Storage/.photoShare
Removing backup-tmp/Storage/DCIM
Removing backup-tmp/Storage/.$Trash$
Removing backup-tmp/Storage/HD Wallpapers
Removing backup-tmp/Storage/HuaweiSystem
Removing backup-tmp/Storage/andtidwiki
Removing backup-tmp/Storage/Sounds
Removing backup-tmp/Storage/Huawei
Removing backup-tmp/Storage/HiDisk
Removing backup-tmp/Storage/nextcloud
Removing backup-tmp/Storage/Slack
Removing backup-tmp/Storage/Movies
Removing backup-tmp/Storage/Wi-Fi Direct
Removing backup-tmp/Storage/Android
Removing backup-tmp/Storage/EditedOnlinePhotos
Removing backup-tmp/Storage/Pictures
Removing backup-tmp/Storage/Calibre_Companion
Removing backup-tmp/Storage/komoot
Removing backup-tmp/Apps
Removing backup-tmp/Contacts
Removing backup-tmp/Storage

Everything is Ok
Backed up successfully.
If this project helped you, please star the GitHub repository. It lets me know that there are people using this script and I should
 continue working on it.
ft@aura ~/Downloads/mrrfv-linux-android-backup-1d395af
 % 
```

Figure 5: The *Everything is Ok* message tells you that the backup process is complete. Keep the assigned password safe for possible recovery.

terminal window using drag and drop. After pressing the Enter key, LAB asked for the password and then opened the archive. The data was then restored automatically (Figure 7).

I also tried out this process on a Google Pixel 3a running GrapheneOS, but the device was not recognized, which is probably due to the security features of this distribution. Upon closer inspection, I found that GrapheneOS comes with its own backup app named SeedVault. Even after disabling SeedVault, the Pixel 3a could not be backed up using LAB.

Conclusions

LAB creates backups for almost all Android devices and can also restore them. Using LAB is easy compared to some proprietary Android apps which often expect a device with an open bootloader. Moreover, LAB also backs up the saved contacts by installing a small app on your Android device during the process. Last but not least, LAB is open source: You can see what it does and improve or expand it.

The custom Android by Huawei and the Murena One with the custom ROM /e/OS backed up and restored without any issues. Only GrapheneOS went into lockdown due to the security settings, but it comes with its own backup app anyway. Next on the LAB developers' agenda is a GUI to make backing up and restoring data even easier. ■■■

Author

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

Info

[1] adb backup: *https://android-review.googlesource.com/c/platform/packages/modules/adb/+/1784571*

[2] Linux Android Backup: *https://github.com/mrrfv/linux-android-backup*

[3] WSL: *https://en.wikipedia.org/wiki/Windows_Subsystem_for_Linux*

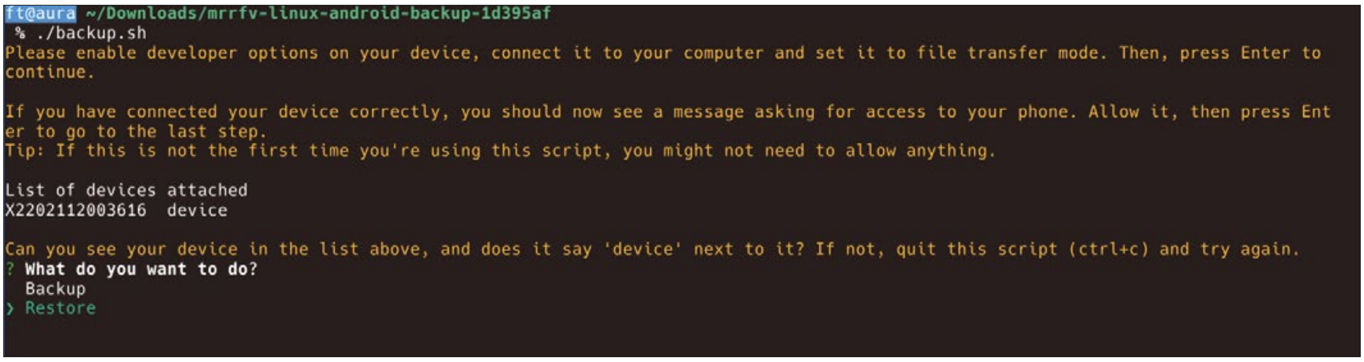
[4] APK: *https://en.wikipedia.org/wiki/Apk_(file_format)*

[5] Download: *https://github.com/mrrfv/linux-android-backup/archive/refs/heads/master.zip*

[6] Murena One: *https://linuxnews.de/2022/06/murena-one-mit-e-os-v1-erster-eindruck/* (German)

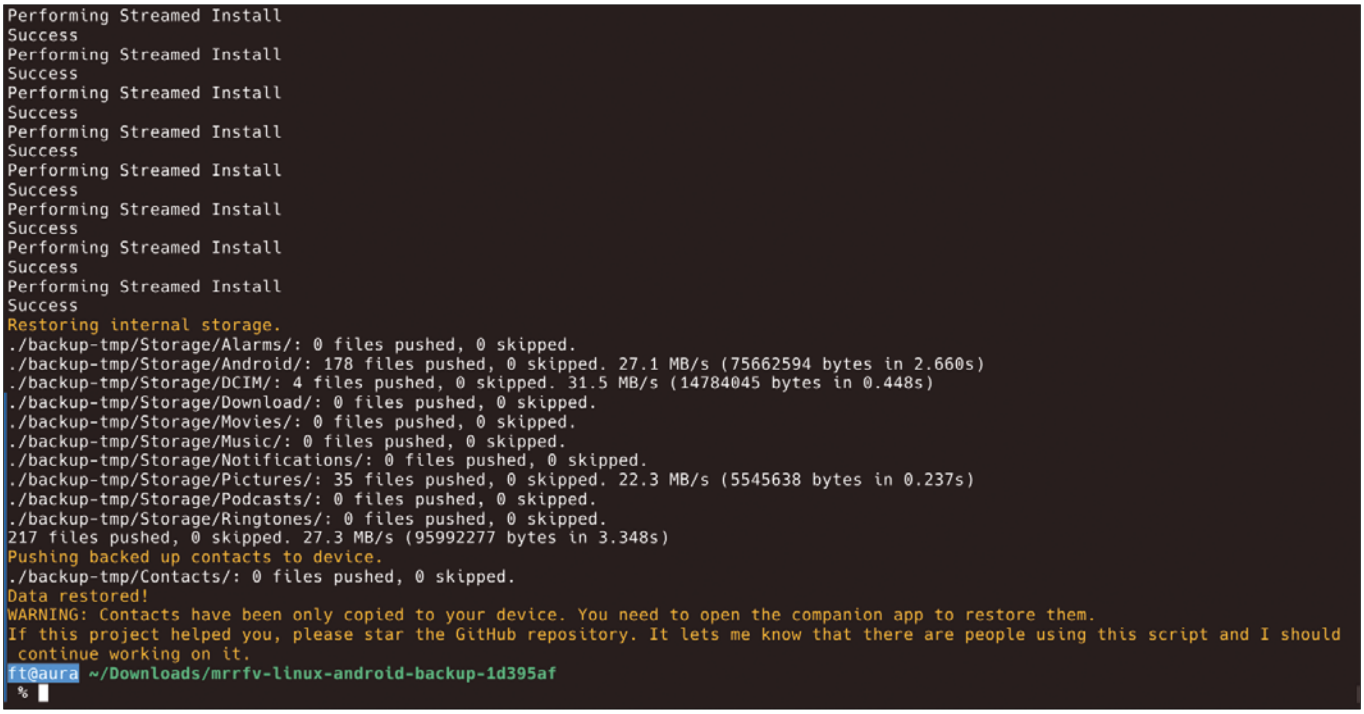
[7] LineageOS: *https://en.wikipedia.org/wiki/LineageOS*

[8] AOSP: *https://source.android.com/*



```
ft@aura ~/Downloads/mrrfv-linux-android-backup-1d395af
 % ./backup.sh
Please enable developer options on your device, connect it to your computer and set it to file transfer mode. Then, press Enter to
continue.

If you have connected your device correctly, you should now see a message asking for access to your phone. Allow it, then press Ent
er to go to the last step.
Tip: If this is not the first time you're using this script, you might not need to allow anything.

List of devices attached
X2202112003616  device

Can you see your device in the list above, and does it say 'device' next to it? If not, quit this script (ctrl+c) and try again.
? What do you want to do?
  Backup
› Restore
```

Figure 6: Restore works in the same way as backup, except that you select *Restore*.



```
Performing Streamed Install
Success
Performing Streamed Install
Success
Performing Streamed Install
Success
Performing Streamed Install
Success
Performing Streamed Install
Success
Performing Streamed Install
Success
Performing Streamed Install
Success
Performing Streamed Install
Success
Restoring internal storage.
./backup-tmp/Storage/Alarms/: 0 files pushed, 0 skipped.
./backup-tmp/Storage/Android/: 178 files pushed, 0 skipped. 27.1 MB/s (75662594 bytes in 2.660s)
./backup-tmp/Storage/DCIM/: 4 files pushed, 0 skipped. 31.5 MB/s (14784045 bytes in 0.448s)
./backup-tmp/Storage/Download/: 0 files pushed, 0 skipped.
./backup-tmp/Storage/Movies/: 0 files pushed, 0 skipped.
./backup-tmp/Storage/Music/: 0 files pushed, 0 skipped.
./backup-tmp/Storage/Notifications/: 0 files pushed, 0 skipped.
./backup-tmp/Storage/Pictures/: 35 files pushed, 0 skipped. 22.3 MB/s (5545638 bytes in 0.237s)
./backup-tmp/Storage/Podcasts/: 0 files pushed, 0 skipped.
./backup-tmp/Storage/Ringtones/: 0 files pushed, 0 skipped.
217 files pushed, 0 skipped. 27.3 MB/s (95992277 bytes in 3.348s)
Pushing backed up contacts to device.
./backup-tmp/Contacts/: 0 files pushed, 0 skipped.
Data restored!
WARNING: Contacts have been only copied to your device. You need to open the companion app to restore them.
If this project helped you, please star the GitHub repository. It lets me know that there are people using this script and I should
 continue working on it.
ft@aura ~/Downloads/mrrfv-linux-android-backup-1d395af
 % █
```

Figure 7: On the Murena One test device, only 27MB needed to be restored, and the process was impressively fast. The *Data restored* message indicates that the action has completed successfully.

Hone your skills with special editions!

Get to know Shell, LibreOffice, Linux, and more from our Special Edition library.

The *Linux Magazine* team has created a series of single volumes that give you a deep-dive into the topics you want.

Available in print or digital format

Check out the full library!

shop.linuxnewmedia.com

FREE DVD!
openSUSE Leap 15.3
JOIN THE LINUX REVOLUTION!
ALL THE SOFTWARE YOU NEED!
GETTING STARTED WITH
LINUX
• MORE POWERFUL • MORE SECURE • MO
LEARN HOW TO SET UP A LINUX SYSTE
○ Listen to Music ○ Play Games ○ Process P
○ Surf the Web ○ and Much More!
START
LINUX NEW MEDIA
WWW.LINUX-MAGA
301 BEST BASH COMMANDS
LINUX SHELL
HANDBOOK 2022 Edition
LINUX Special
SUPERCHARGE
YOUR LINUX SKILLS
Power at Your Fingertips
• Pipe and redirect output
• Monitor processes
FREE DVD!
LibreOffice Full Version
Dive deep into the world's greatest free office suite
2022/23 Edition
LibreOffice
Expert
Edit and Save MS Office Files
Write Your Own LO Macros
Save time and automate common tasks
LINUX NEW MEDIA USA
BUILD A RASP PI RADIO!
MakerSpace #02
HANDS-ON PROJECTS FOR MAKERS
Cool Tricks!
Charge up with a saltwater battery
Painting with Light
Sure you need a programmable light stick
PiMiga 2.0
Get your game on with this Amiga emulator
MORE FUN FOR FPGA GEEKS!
LINUX NEW MEDIA
ARCHIVE DVD
RASPBERRY PI GEEK
THE COMPLETE ARCHIVE
2,000 pages of maker projects and more!
FREE DVD $39.90 VALUE!
MakerSpace
HANDS-ON PROJECTS FOR MAKERS
TUNE YOUR LINUX SYSTEM
2022 EDITION
Cool Linux Hacks
84 HACKS
Tricks and shortcuts for Linux geeks
• Speed up downloads with Xtreme Download Manager
• Search text, PDF, and Office files with one tool
• Run VMs on a Proxmox server without installing client software
Discover the secrets of the experts
RETRO FUN:
Play DOS Games with DOSBox
AUDIO EXPERTS:
Tools for DJs, Musicians, Composers
WWW.LINUX-MAGAZINE.COM

New options for traditional Unix commands

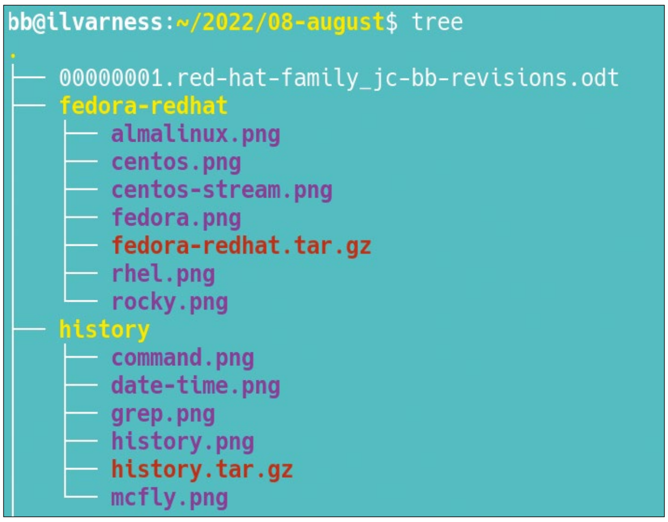
Make New Friends

If you are looking for modern display options or more speed at the command line, these alternatives to traditional Unix commands may be just what you need. *By Bruce Byfield*

The development of Unix began over 50 years ago. In contrast to commercial software, most of Unix's basic commands are still in use today. However, in the past couple decades, alternatives have been developed for a variety of reasons. For instance, restrictions such as lack of memory or color monitors or limits on file name lengths have long since disappeared, opening the way for more to be done with a command. Also, now that desktop environments are the norm, users demand more even at the command line. Sometimes, too, commands or some of their options have become obsolete or too numerous and too complex for their most common uses, resulting in the desire for simplification. At other times, developers want to try their hand at developing an improved version. Here are some of the new options for some common traditional commands.

tree or exa for ls

As one of the most commonly used commands for command-line navigation, `ls` is a popular choice for updating. If nothing else, the fact that many distributions routinely alias `ls` for `ls --color` highlights how much room there is for improvement. It's not surprising, then, that at least half a dozen replacements for `ls` exist, all of them



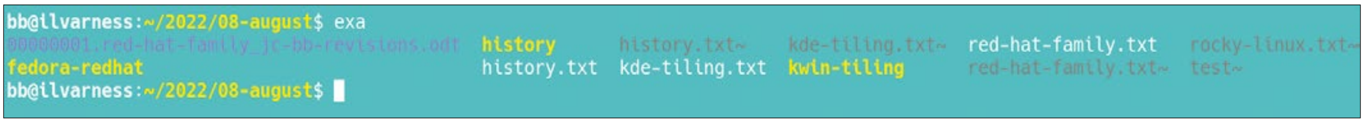
```
bb@ilvarness:~/2022/08-august$ tree
.
├── 00000001.red-hat-family_jc-bb-revisions.odt
├── fedora-redhat
│   ├── almalinux.png
│   ├── centos.png
│   ├── centos-stream.png
│   ├── fedora.png
│   ├── fedora-redhat.tar.gz
│   ├── rhel.png
│   └── rocky.png
└── history
    ├── command.png
    ├── date-time.png
    ├── grep.png
    ├── history.png
    ├── history.tar.gz
    └── mcfly.png
```

Figure 1: Similar to a file manager, `tree` lists directories and files in a color-coded directory tree.

Author

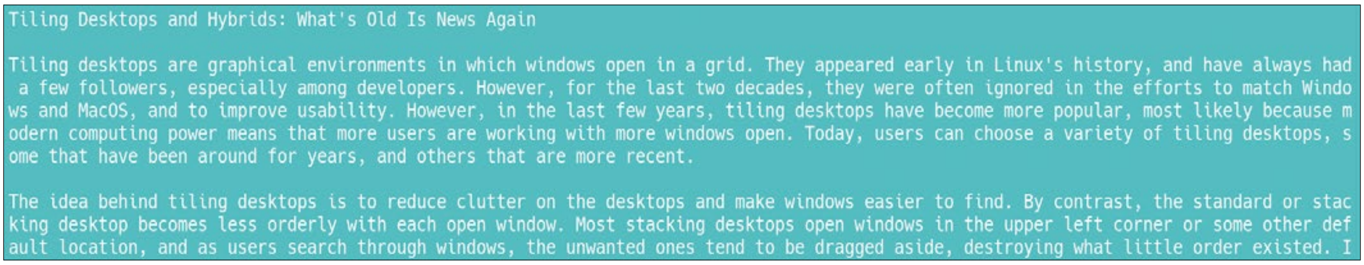
Bruce Byfield is a computer journalist and a freelance writer and editor specializing in free and open source software. In addition to his writing projects, he also teaches live and e-learning courses. In his spare time, Bruce writes about Northwest Coast art (*http://brucebyfield.wordpress.com*). He is also cofounder of Prentice Pieces, a blog about writing and fantasy at *https://prenticepieces.com/*.

Photo by Louis-Philippe Poitras on Unsplash



```
bb@ilvarness:~/2022/08-august$ exa
00000001.red-hat-family_jc-bb-revisions.odt  history      history.txt~  kde-tiling.txt~  red-hat-family.txt   rocky-linux.txt~
fedora-redhat                                history.txt  kde-tiling.txt  kwin-tiling      red-hat-family.txt~  test~
bb@ilvarness:~/2022/08-august$ █
```

Figure 2: `exa` **is a modern replacement for** `ls`**.**



```
Tiling Desktops and Hybrids: What's Old Is News Again

Tiling desktops are graphical environments in which windows open in a grid. They appeared early in Linux's history, and have always had
 a few followers, especially among developers. However, for the last two decades, they were often ignored in the efforts to match Windo
ws and MacOS, and to improve usability. However, in the last few years, tiling desktops have become more popular, most likely because m
odern computing power means that more users are working with more windows open. Today, users can choose a variety of tiling desktops, s
ome that have been around for years, and others that are more recent.

The idea behind tiling desktops is to reduce clutter on the desktops and make windows easier to find. By contrast, the standard or stac
king desktop becomes less orderly with each open window. Most stacking desktops open windows in the upper left corner or some other def
ault location, and as users search through windows, the unwanted ones tend to be dragged aside, destroying what little order existed. I
```

Figure 3: **When a file is long,** `bat` **automatically displays the file with** `less`**, making scrolling back and forth easier.**

routinely using color as well as other enhancements.

One popular replacement is `tree`. As the name suggests, `tree` displays results in a tree structure, making the results easier to read (Figure 1). With `tree`, you can specify the tree's depth, as well as the full path to each file. Conveniently, `tree` also includes five sorting orders. Also useful, `tree` allows for the customization of colors through environmental variables and shows the path to the original file for symlinks.

Another popular `ls` replacement, `exa` (Figure 2), starts with color and human-readable file sizes enabled, as well a variety of display options, ranging from one entry per line and a table for long forms. Like `tree`, `exa` has options for a tree view and different sorting orders according to different file attributes.

bat for cat

Although many users prefer the `less` command, `cat` remains a popular choice for viewing text files. As a complete replacement for `cat`, `bat` also adds syntax highlighting. If a file is longer than the screen, `bat` will display the file using `less` to make scrolling back and forth easier (Figure 3). For coders, `bat` integrates with `git`.

tldr for man

Man pages are the most common form of Linux documentation for commands. Unfortunately, their quality varies with the writers. Some man pages lack context and examples, and some are too technical for average users. Moreover, even the best-written can be long and complex. While `tldr` (which stands for "too long; didn't read," a common synonym for summaries) [1] does not replace man pages, it does give clear, concise explanations and covers the most common use cases (Figure 4). To install the `tldr` database, you need to install the Node.js client and run:

```
npm install -g tldr
```

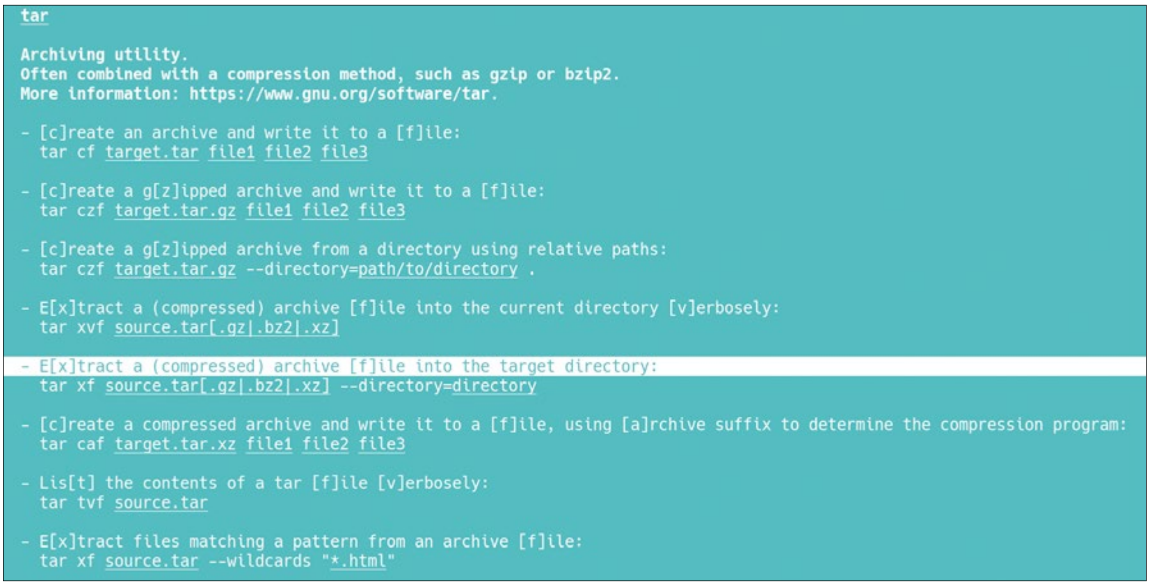
or you can install the Python client with:

```
pip3 install tldr
```

The first time you use `tldr`, there will be a short delay while the database is created. Once that is done, `tldr` has a similar or faster response time than that of `man`.

apt for apt-get

As the front end for dpkg, the venerable package manager for Debian and its derivatives, `apt-get` has been expanded with dozens of utilities over the years, becoming bewildering for new users. As a response to this expansion, `apt` simplifies `apt-get`'s basic commands while including the most useful options from utilities such as `apt-cache` and `apt-query` (Figure 5). In addition, forthcoming versions of `apt` will allow editing of



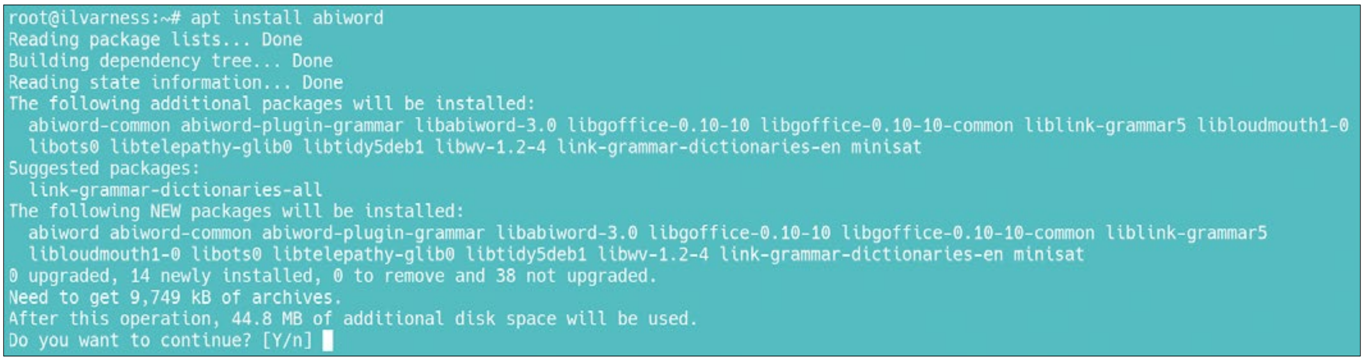
```
tar

Archiving utility.
Often combined with a compression method, such as gzip or bzip2.
More information: https://www.gnu.org/software/tar.

- [c]reate an archive and write it to a [f]ile:
  tar cf target.tar file1 file2 file3

- [c]reate a g[z]ipped archive and write it to a [f]ile:
  tar czf target.tar.gz file1 file2 file3

- [c]reate a g[z]ipped archive from a directory using relative paths:
  tar czf target.tar.gz --directory=path/to/directory .

- E[x]tract a (compressed) archive [f]ile into the current directory [v]erbosely:
  tar xvf source.tar[.gz|.bz2|.xz]

- E[x]tract a (compressed) archive [f]ile into the target directory:
  tar xf source.tar[.gz|.bz2|.xz] --directory=directory

- [c]reate a compressed archive and write it to a [f]ile, using [a]rchive suffix to determine the compression program:
  tar caf target.tar.xz file1 file2 file3

- Lis[t] the contents of a tar [f]ile [v]erbosely:
  tar tvf source.tar

- E[x]tract files matching a pattern from an archive [f]ile:
  tar xf source.tar --wildcards "*.html"
```

Figure 4: **A quick crib for man pages,** `tldr` **features useful examples.**

package sources. Once the command is entered, then feedback is the same as for `apt-get`. The result is so convenient that even long-time Debian users have taken to using `apt`.

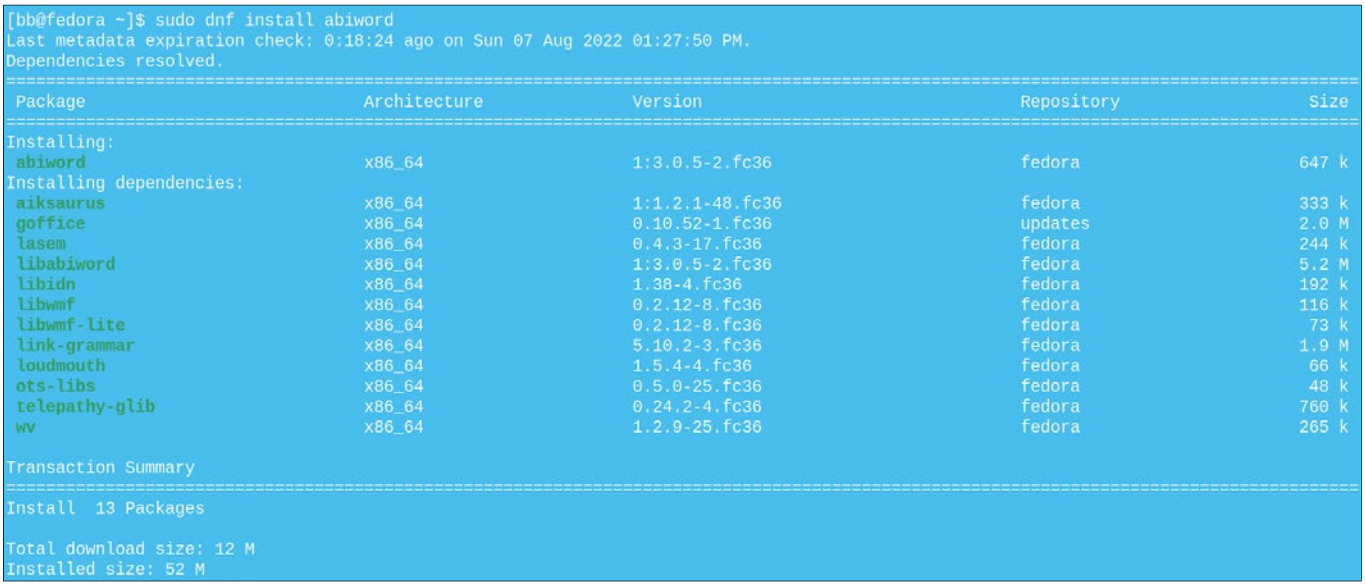
dnf for yum

Traditionally, `yum` has functioned as the package manager for RPM packages. Unfortunately, API documentation and expertise have long been lacking in `yum`, making bug fixing and improvements difficult. For this reason, `dnf` has been slowly replacing `yum` in the past few years (Figure 6). In addition to being a complete replacement, `dnf` allows



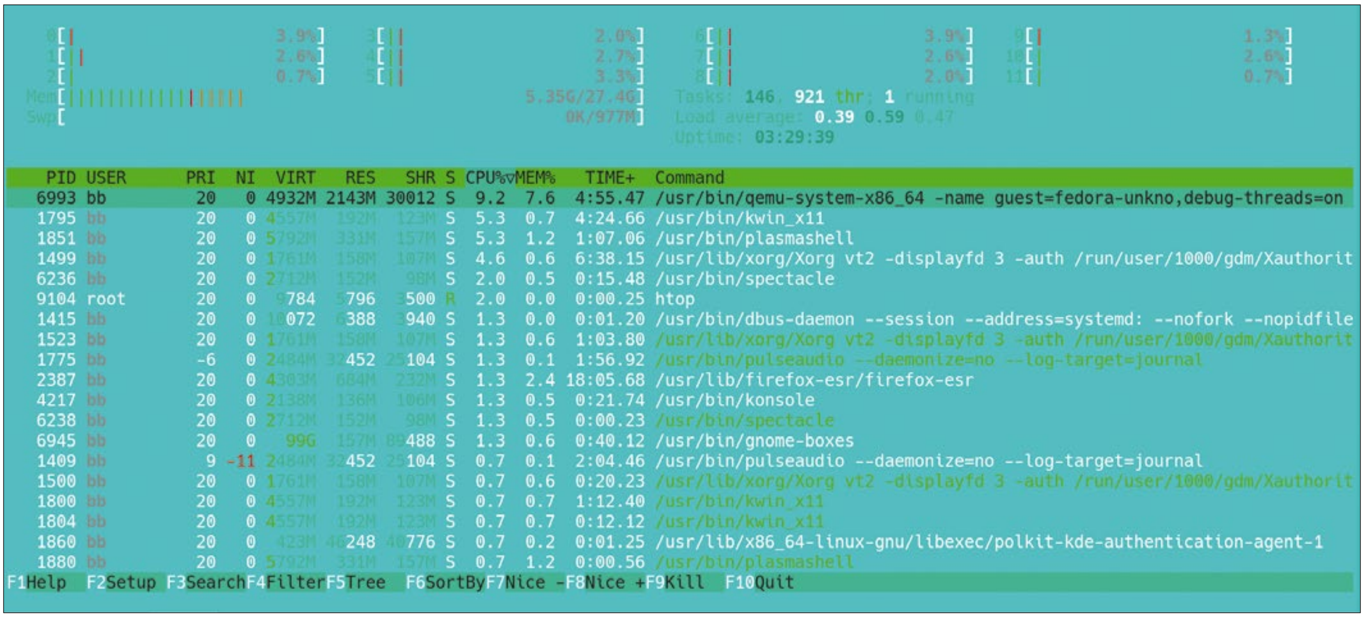
```
root@ilvarness:~# apt install abiword
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  abiword-common abiword-plugin-grammar libabiword-3.0 libgoffice-0.10-10 libgoffice-0.10-10-common liblink-grammar5 libloudmouth1-0
  libots0 libtelepathy-glib0 libtidy5deb1 libwv-1.2-4 link-grammar-dictionaries-en minisat
Suggested packages:
  link-grammar-dictionaries-all
The following NEW packages will be installed:
  abiword abiword-common abiword-plugin-grammar libabiword-3.0 libgoffice-0.10-10 libgoffice-0.10-10-common liblink-grammar5
  libloudmouth1-0 libots0 libtelepathy-glib0 libtidy5deb1 libwv-1.2-4 link-grammar-dictionaries-en minisat
0 upgraded, 14 newly installed, 0 to remove and 38 not upgraded.
Need to get 9,749 kB of archives.
After this operation, 44.8 MB of additional disk space will be used.
Do you want to continue? [Y/n] █
```

Figure 5: A simplification of `apt-get`, `apt` covers the most common uses.



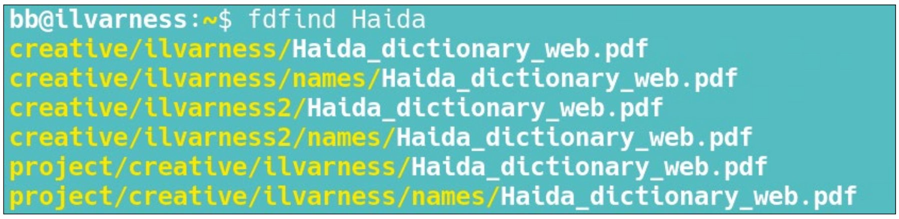
```
[bb@fedora ~]$ sudo dnf install abiword
Last metadata expiration check: 0:18:24 ago on Sun 07 Aug 2022 01:27:50 PM.
Dependencies resolved.
======================================================================================================================================
 Package                           Architecture               Version                                Repository                 Size
======================================================================================================================================
Installing:
 abiword                           x86_64                     1:3.0.5-2.fc36                         fedora                    647 k
Installing dependencies:
 aiksaurus                         x86_64                     1:1.2.1-48.fc36                        fedora                    333 k
 goffice                           x86_64                     0.10.52-1.fc36                         updates                   2.0 M
 lasem                             x86_64                     0.4.3-17.fc36                          fedora                    244 k
 libabiword                        x86_64                     1:3.0.5-2.fc36                         fedora                    5.2 M
 libidn                            x86_64                     1.38-4.fc36                            fedora                    192 k
 libwmf                            x86_64                     0.2.12-8.fc36                          fedora                    116 k
 libwmf-lite                       x86_64                     0.2.12-8.fc36                          fedora                     73 k
 link-grammar                      x86_64                     5.10.2-3.fc36                          fedora                    1.9 M
 loudmouth                         x86_64                     1.5.4-4.fc36                           fedora                     66 k
 ots-libs                          x86_64                     0.5.0-25.fc36                          fedora                     48 k
 telepathy-glib                    x86_64                     0.24.2-4.fc36                          fedora                    760 k
 wv                                x86_64                     1.2.9-25.fc36                          fedora                    265 k

Transaction Summary
======================================================================================================================================
Install  13 Packages

Total download size: 12 M
Installed size: 52 M
```

Figure 6: For distros that use RPM packages, `dnf` is well on its way to replacing `yum`.



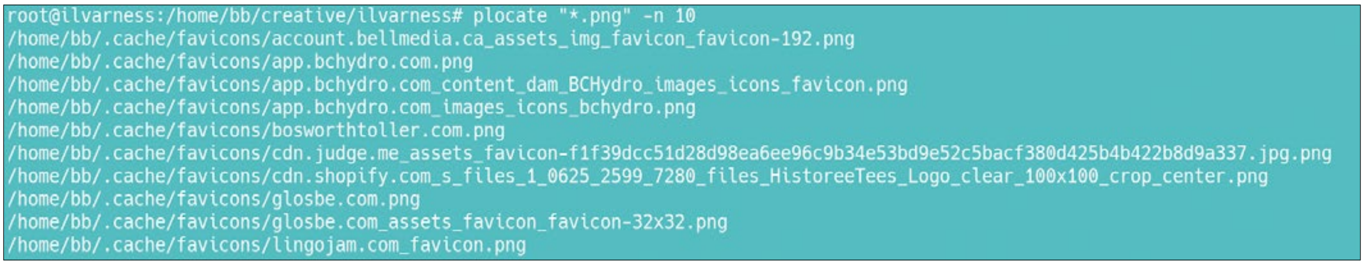
```
    0[|                      3.9%]   3[||                     2.0%]   6[||                     3.9%]   9[|                      1.3%]
    1[||                     2.6%]   4[||                     2.7%]   7[||                     2.6%]  10[|                      2.6%]
    2[|                      0.7%]   5[||                     3.3%]   8[||                     2.0%]  11[|                      0.7%]
  Mem[|||||||||||||||||||                          5.35G/27.4G]   Tasks: 146, 921 thr; 1 running
  Swp[                                               0K/977M]   Load average: 0.39 0.59 0.47
                                                                Uptime: 03:29:39

    PID USER       PRI  NI  VIRT   RES   SHR S CPU%▽MEM%   TIME+  Command
   6993 bb          20   0 4932M 2143M 30012 S  9.2  7.6  4:55.47 /usr/bin/qemu-system-x86_64 -name guest=fedora-unkno,debug-threads=on
   1795 bb          20   0 4557M  192M  123M S  5.3  0.7  4:24.66 /usr/bin/kwin_x11
   1851 bb          20   0 5792M  331M  157M S  5.3  1.2  1:07.06 /usr/bin/plasmashell
   1499 bb          20   0 1761M  158M  107M S  4.6  0.6  6:38.15 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthorit
   6236 bb          20   0 2712M  152M   98M S  2.0  0.5  0:15.48 /usr/bin/spectacle
   9104 root        20   0  9784  5796  3500 R  2.0  0.0  0:00.25 htop
   1415 bb          20   0 10072  6388  3940 S  1.3  0.0  0:01.20 /usr/bin/dbus-daemon --session --address=systemd: --nofork --nopidfile
   1523 bb          20   0 1761M  158M  107M S  1.3  0.6  1:03.80 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthorit
   1775 bb          -6   0 2484M 32452 25104 S  1.3  0.1  1:56.92 /usr/bin/pulseaudio --daemonize=no --log-target=journal
   2387 bb          20   0 4303M  684M  232M S  1.3  2.4 18:05.68 /usr/lib/firefox-esr/firefox-esr
   4217 bb          20   0 2138M  136M  106M S  1.3  0.5  0:21.74 /usr/bin/konsole
   6238 bb          20   0 2712M  152M   98M S  1.3  0.5  0:00.23 /usr/bin/spectacle
   6945 bb          20   0   99G  157M 89488 S  1.3  0.6  0:40.12 /usr/bin/gnome-boxes
   1409 bb           9 -11 2484M 32452 25104 S  0.7  0.1  2:04.46 /usr/bin/pulseaudio --daemonize=no --log-target=journal
   1500 bb          20   0 1761M  158M  107M S  0.7  0.6  0:20.23 /usr/lib/xorg/Xorg vt2 -displayfd 3 -auth /run/user/1000/gdm/Xauthorit
   1800 bb          20   0 4557M  192M  123M S  0.7  0.7  1:12.40 /usr/bin/kwin_x11
   1804 bb          20   0 4557M  192M  123M S  0.7  0.7  0:12.12 /usr/bin/kwin_x11
   1860 bb          20   0  423M 46248 40776 S  0.7  0.2  0:01.25 /usr/lib/x86_64-linux-gnu/libexec/polkit-kde-authentication-agent-1
   1880 bb          20   0 5792M  331M  157M S  0.7  1.2  0:00.56 /usr/bin/plasmashell
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Figure 7: As a replacement for `top`, `htop` offers improved readability of data.



```
bb@ilvarness:~$ fdfind Haida
creative/ilvarness/Haida_dictionary_web.pdf
creative/ilvarness/names/Haida_dictionary_web.pdf
creative/ilvarness2/Haida_dictionary_web.pdf
creative/ilvarness2/names/Haida_dictionary_web.pdf
project/creative/ilvarness/Haida_dictionary_web.pdf
project/creative/ilvarness/names/Haida_dictionary_web.pdf
```

Figure 8: **A faster version of** `find`**,** `fd` **displays color-coded results.**



```
root@ilvarness:/home/bb/creative/ilvarness# plocate "*.png" -n 10
/home/bb/.cache/favicons/account.bellmedia.ca_assets_img_favicon_favicon-192.png
/home/bb/.cache/favicons/app.bchydro.com.png
/home/bb/.cache/favicons/app.bchydro.com_content_dam_BCHydro_images_icons_favicon.png
/home/bb/.cache/favicons/app.bchydro.com_images_icons_bchydro.png
/home/bb/.cache/favicons/bosworthtoller.com.png
/home/bb/.cache/favicons/cdn.judge.me_assets_favicon-f1f39dcc51d28d98ea6ee96c9b34e53bd9e52c5bacf380d425b4b422b8d9a337.jpg.png
/home/bb/.cache/favicons/cdn.shopify.com_s_files_1_0625_2599_7280_files_HistoreeTees_Logo_clear_100x100_crop_center.png
/home/bb/.cache/favicons/glosbe.com.png
/home/bb/.cache/favicons/glosbe.com_assets_favicon_favicon-32x32.png
/home/bb/.cache/favicons/lingojam.com_favicon.png
```

Figure 9: **Here,** `plocate` **limits the results to the first 10 found.**

extensions in several programming languages and has faster dependency resolution than `yum`. Also, `dnf` has abilities that `yum` lacks, such as the ability to switch repositories if needed or to delete old kernels. In general, `dnf` is much quicker than `yum` and flexible enough to skip unnecessary steps to speed up installation. For instance, if you have just installed one package, `dnf` will not update the package list for a second one.

htop for top

To display applications that use the most memory or take up the most RAM on a system, `top` has been the classic choice. Installed by default on Linux systems, `top` displays only plain text – which is not surprising, considering it was developed in 1984. By contrast, `htop` (Figure 7), developed in 2004, uses color to make reading easier. Unlike `top`, `htop` supports horizontal and vertical scrolling, as well as searching and the use of a mouse. These features make replacing `top` with `htop` a no brainer for system administrators.

fd for find

If you are looking for a replacement for `find`, `fd` [2] (or `fdfind`, as Debian calls it) covers the most common uses of `find` (Figure 8). Especially if you use regular expressions, `fd` can produce results four or five times faster than `find`. However, `fd` offers other equally useful features, including a simpler syntax, color input, and defaults that include case-insensitivity and the ignoring of hidden files. Should you need case-sensitivity, all you need to do is use uppercase letters in the search pattern.

plocate for locate

Over the years, `locate` has had several different versions, including `mlocate`. All use a database to increase the speed of file searches. The most recent version, `plocate` (Figure 9), is a drop-in replacement that is starting to be the norm in many major distributions. The main advantage of `plocate` is that it can be three or four times faster than any of its predecessors on large searches, thanks mainly to the fact that it usually does not search through its entire database unless the search item is extremely small. Unlike its predecessors, `plocate` can also search for multiple strings at the same time, which can speed up the file search.

ripgrep for grep

Together with its variants, `grep` offers a comprehensive search function for text patterns. However, sometimes, its speed seems measured in eons. The main reason to use `ripgrep` is that it is much faster than `grep` for large searches (Figure 10). Much of this speed is obtained by ignoring hidden and binary files by default and by the use of a `.gitignore` file to list other exceptions. Even more important, while `grep` is confined to plain text files, `ripgrep` supports a number of different file types and Unicode encodings, which can either make a search more comprehensive or else restrict it. Other useful features include an enhanced regex engine. While `ripgrep` is not the only `grep` replacement, it outperforms alternatives such as `ag`, `git grep`, `ucg`, `pt`, and `sif` in most cases.

Old or New?

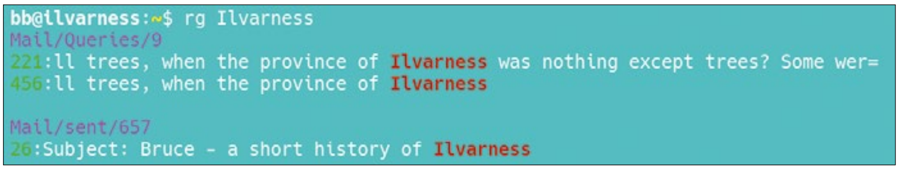
Most of the replacements listed here closely resemble the commands they replace, so switching to them is usually easy. However, whether the switch is worth your while depends on you and your computing. For instance, home users might find minimal use for `top`, let alone `htop`. Similarly, to the color blind, the popular color-coding may be of little use unless you can reconfigure it.

Just as important, old habits die hard: Even when you want to switch, you may still find yourself using the old command. In some cases, you might want to make the old command an alias for the new if your distribution does not already do so. But be careful: In cases such as `apt-get` and `apt`, there is enough difference between the old and the new that you will want to keep the commands distinct. Often, the new command covers only the most common cases. ■■■

Info

[1] tldr: *https://github.com/tldr-pages/tldr#how-do-i-use-it*

[2] fd: *https://github.com/sharkdp/fd#benchmark*



```
bb@ilvarness:~$ rg Ilvarness
Mail/Queries/9
221:ll trees, when the province of Ilvarness was nothing except trees? Some wer=
456:ll trees, when the province of Ilvarness

Mail/sent/657
26:Subject: Bruce - a short history of Ilvarness
```

Figure 10: **Designed primarily for speed,** `ripgrep` **modifies** `grep`**.**

Rewriting a photo tagger in Go

Digital Shoe Box

In honor of the 25th anniversary of his Programming Snapshot column, Mike Schilli revisits an old problem and solves it with Go instead of Perl.

By Mike Schilli

Hurray! This issue marks the 25th anniversary of my "Programming Snapshot" column, which first appeared in the German edition of *Linux Magazine* back in October 1997 (originally under the "Perl Snapshot" banner). Times have changed: Now the featured programs in this column mainly use Go, but you might also see Ruby, Python, or even TeX, as was the case recently.

For this dinosaur birthday party, I thought I might rewrite a tool I put together in Perl back in the dot-com era, but looking at it from today's perspective in Go. The photo tagger from 2003 (it was called Image Database [1] or `idb` for short) is something I've been wanting to use again for a long time.

The `idb` tool assigns one or more tags to a set of photo files, distributed over arbitrary subdirectories somewhere on the hard drive. Once tagged with the tool, the same program can retrieve the photos if you provide the name of the desired tag. The problem with the old Perl code, though, is that you need both the time and the inclination to go through the installation and dependency hell of all the Perl modules used by it. Moreover, many years have passed since then, and some CPAN module developers have broken backward compatibility by changing the original programming interfaces. Luckily, it's 2022, and Go has solved these kinds of installation problems for all time, as you can compile static binaries that run on similar architectures.

Author

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

Also, the old tagger script used a separate standalone MySQL server back in the day, but today – at least for tools that only run locally – I prefer to have everything bundled into a single binary, such as an embedded SQLite flat file database engine. Somewhat surprisingly, the rewrite with newfangled technology was remarkably quick.

SQLosaurus Rex

SQL databases are a bit out of fashion these days. If you only need a key-value store for your data, you are more likely to use a persistent cache or a server solution like Redis. However, for local data not exceeding a few megabytes, running an external process is unlikely to be worthwhile. Also, I tend to be suspicious of binary data in caches or key-value stores such as Berkeley DB. Instead, I prefer to take a direct look at the data myself from time to time. SQLite is an ideal database because it stores the data in a single file that a command-line tool such as `sqlite3` lets you browse. Plus, backing up a single file is easier than creating and backing up a dump of a running database.

On top of this, SQLite is one of the few open source tools that is truly in the public domain. This is why there's a Go module on GitHub like `mattn/go-sqlite3` that lets you legally include the SQLite source code in any program you write and distribute. The Go compiler then turns SQLite, the library, and the application code into a single binary that can be copied to other computers with a similar architecture and will run there without any installation hassles. It's the end of dependency hell as we know it – I never thought I'd get to see that! For the installation at least, recompiling legacy code can be a different story and subject to problems arising from non-backward-compatible changes.

Three Tables

So which relational data model is suitable for a photo tagger application? The `idb` tool assigns one or more tags to one or more files. Since the Stone Age of data processing, the three-table model has proven useful for many-to-many relations like this: two tables to assign index numbers to tag names and file paths, and then a third, two-column table that maps the index numbers to each other if a particular tag is attached to a particular file.

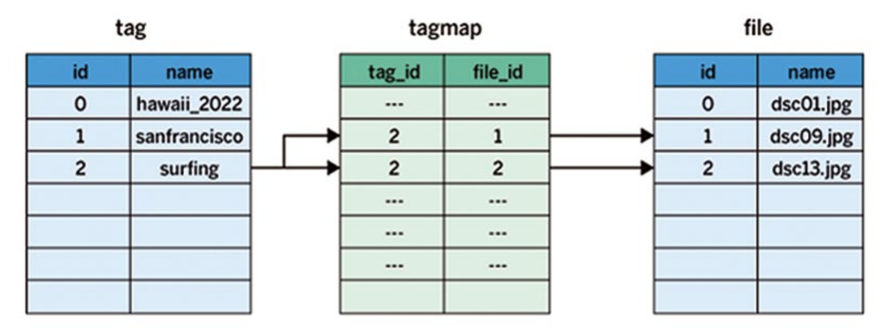
In this way, the database only needs to store the full tag or file name once in each case, a basic requirement for a normalized database. This has advantages beyond wasted disk space due to duplicate storage. Moreover, if the user corrects a typo in a tag, the database only has to correct it in one place, even if the tag is attached to thousands of files.

For example, to tag the `dsc13.jpg` photo file with the `surfing` tag (Figure 1), the tool first creates a new entry for the `surfing` tag in the `tag` table (on the left

Lead Image © Erik Reis, 123RF.com

of Figure 1) if the entry does not already exist. SQLite automatically assigns the associated sequential index number, 2 in this case, to the entry because entries start at an index of `0` and `surfing` is the third entry in the `name` column. In addition, the file name `dsc13.jpg`, if not already present, needs to be inserted into the `file` table – in Figure 1 it ends up in the third row and has an index number of 2 (again, an ascending index starting at `0`).

That takes care of the two lookup tables for tags and file names. Now you need the actual assignment of the tag to the photo. This is handled by an entry in the `tag map` table (center, Figure 1), which assigns a tag ID of 2 to the file ID 2. All done! Using typical SQL joins, it is then easy for the database to respond to the question as to which photos were tagged with `surfing`. An SQL query to this effect quickly yields `dsc13.jpg` and possibly others. In the opposite case, the query engine can also easily discover which tags are attached to the `dsc13.jpg` image file, again by joining the tables.



tag

id	name
0	hawaii_2022
1	sanfrancisco
2	surfing

tagmap

tag_id	file_id
---	---
2	1
2	2
---	---
---	---
---	---

file

id	name
0	dsc01.jpg
1	dsc09.jpg
2	dsc13.jpg

Figure 1: The SQL database schema for the photo tagger.

Table 1: Commands

idb --tag=foo image.jpg ...	Tag photos with *foo*
idb --tag=foo	Find photos with the *foo* tag
idb --tag=foo --xlink	Find photos with the *foo* tag and create a local symlink
idb --tags	List all tags

Homemade

The finished `idb` binary, linked together from the Go sources for this article, can carry out the commands listed in Table 1. The binary supports tagging files, searching for files with a specific tag, and listing all tags assigned so far. As a special treat, the `--xlink` option generates a directory full of symlinks pointing to the original photos for files found for a given tag. With a tool such as iNuke [2], featured in a recent column, the photos can then be viewed, and the best ones selected.

Molding and Casting

So, how does the `idb` tool work? First, `idb` contacts the tag database, which resides in the `.idb.db` file in the user's home directory. If `idb` does not find a database there, it creates a new one. To do this, `idb` uses the SQL commands in `createDBSQL()` starting in line 23 in Listing 1.

Some special SQLite features in the three table definitions will facilitate the insertion work later on. The `tag` and `file` tables, which assign numeric IDs to tag and file paths, define an integer with the `primary key` attribute as their first `id` column. This prompts SQLite to constantly increment the IDs of new entries by one – ideal for referencing them by their IDs later on in a subtle and unique way.

The `UNIQUE` specifier in the `string` column to the right for tag or file names determines that the table does not allow duplicate name entries. As a practical side effect, the tool can later use `INSERT OR IGNORE` to create entries in one fell swoop if they do not yet exist and leave existing ones there without any complaints.

Name to Number

The `name2id()` function in Listing 2 does just that: It expects a name, either a file name or a tag name, and inserts it into the `file` or `tag` lookup tables, depending

Listing 1: dbinit.go

```
01 package main
02
03 import (
04   "database/sql"
05   _ "github.com/mattn/go-sqlite3"
06   "os"
07   "path"
08 )
09
10 const DbName = ".idb.db"
11
12 func dbInit() (*sql.DB, error) {
13   homedir, err := os.UserHomeDir()
14   if err != nil {
15     return nil, err
16   }
17   dbPath := path.Join(homedir, DbName)
18   db, err := sql.Open("sqlite3", dbPath)
19   _, err = db.Exec(createDBSQL())
20   return db, err
21 }
22
23 func createDBSQL() string {
24   return `
25 CREATE TABLE IF NOT EXISTS tag (
26   id INTEGER PRIMARY KEY,
27   name TEXT UNIQUE
28 );
29 CREATE TABLE IF NOT EXISTS tagmap (
30   tag_id INTEGER,
31   file_id INTEGER,
32   UNIQUE(tag_id, file_id)
33 );
34 CREATE TABLE IF NOT EXISTS file (
35   id INTEGER PRIMARY KEY,
36   name TEXT UNIQUE
37 );`
38 }
```

on what the `table` parameter is set to. If the name does not yet exist, the `INSERT` command appends the new entry at the bottom of the table in line 10, and SQLite automatically generates a previously unused index integer number `id` for it.

After using `Prepare` to prepare the statement in line 11, `Exec()` in line 14 plants the `name` into the SQL command, using a technique that provides protection against SQL injection attacks, and then executes the query. If the file or tag already exists in the database, nothing happens thanks to `OR IGNORE`.

The subsequent name lookup using `SELECT` in line 19 looks for the same entry again and returns the associated `id` after line 21 has retrieved the only matching row. The `name2id()` function is nothing more than a convenient, persistent way to assign names to numeric IDs and return the latter on request.

Using the index numbers for the tag and file, `tagMap()` can glue a tag to a file starting in line 26, calling `INSERT` to append a new table row with the two numbers. If the file already has the tag, `OR IGNORE` will keep things tidy, as previously shown.

When Three Become One

To find all files with a certain tag, the `SELECT` statement in the `tagSearch()` function needs to join all three tables starting in line 35. The tag I am looking for in `tag` has an index number that `tagmap` assigns to the index number of a file, whose name in turn is indexed in the `file` table.

Because the tag I am looking for is already present as an index number, because of the previous call to `name2id()`, it is up to the query to define a condition that welds together the two tables, `tagmap` and `file`. The `file_id` of the `tagmap` table needs to match the `id` from the `file` table. Line 46 gobbles up the names of the matching files that come pouring in and appends them to the `result` variable in line 52. This is a slice of strings that the function returns to the caller when done.

Finally, the last function in Listing 2, `tagList`, lists all tag names found in the `tag` table after SQLite extracts them with the `SELECT` command starting in line 59.

To wrap it all up, the main program in Listing 3 processes flags such as `--tag` (set or query a tag) or `--tags` (list all defined tags) that the user includes in the call at the command line. It opens the

Listing 2: db.go

```
01 package main
02
03 import (
04   "database/sql"
05   "fmt"
06   _ "github.com/mattn/go-sqlite3"
07 )
08
09 func name2id(db *sql.DB, table string, name string)
   (int, error) {
10   query := fmt.Sprintf("INSERT OR IGNORE INTO %s(name)
   VALUES(?)", table)
11   stmt, err := db.Prepare(query)
12   panicOnErr(err)
13
14   _, err = stmt.Exec(name)
15   panicOnErr(err)
16
17   id := -1
18
19   query = fmt.Sprintf("SELECT id FROM %s WHERE name = ?",
   table)
20   row := db.QueryRow(query, name)
21   _ = row.Scan(&id)
22
23   return id, nil
24 }
25
26 func tagMap(db *sql.DB, tagId, fileId int) {
27   query := "INSERT OR IGNORE INTO tagmap(tag_id, file_id)
   VALUES(?, ?)"
28   stmt, err := db.Prepare(query)
29   panicOnErr(err)
30   _, err = stmt.Exec(tagId, fileId)
31   panicOnErr(err)
32   return
33 }
34
35 func tagSearch(db *sql.DB, tagId int) ([]string, error) {
36   result := []string{}
37   query := `
38     SELECT file.name FROM file, tagmap
39     WHERE tagmap.tag_id = ?
40     AND file.id = tagmap.file_id;`
41   rows, err := db.Query(query, tagId)
42   if err != nil {
43     return result, err
44   }
45
46   for rows.Next() {
47     path := ""
48     err = rows.Scan(&path)
49     if err != nil {
50       return result, err
51     }
52     result = append(result, path)
53   }
54
55   return result, nil
56 }
57
58 func tagList(db *sql.DB) {
59   query := `SELECT name FROM tag`
60   rows, err := db.Query(query)
61   panicOnErr(err)
62
63   for rows.Next() {
64     tag := ""
65     err = rows.Scan(&tag)
66     panicOnErr(err)
67     fmt.Printf("%s\n", tag)
68   }
69
70    return
71 }
```

Listing 3: idb.go

```
01 package main
02
03 import (
04   "flag"
05   "fmt"
06   "os"
07   "path"
08   "path/filepath"
09 )
10
11 func main() {
12   flag.Usage = func() {
13     fmt.Printf("Usage: %s --tag=tagname photo ...\n",
                path.Base(os.Args[0]))
14     os.Exit(1)
15   }
16
17   tags := flag.Bool("tags", false, "list all tags")
18   xlink := flag.Bool("xlink", false, "create links in
                   current dir")
19
20   tag := flag.String("tag", "", "tag to assign/search")
21   flag.Parse()
22
23   db, err := dbInit()
24   panicOnErr(err)
25   defer db.Close()
26
27   if *tags {
28     tagList(db)
29     return
30   }
31
32   if tag == nil {
33     flag.Usage()
34   }
35
36   tagId, err := name2id(db, "tag", *tag)
37   panicOnErr(err)
38
39   if flag.NArg() == 0 {
40     matches, err := tagSearch(db, tagId)
41     panicOnErr(err)
42     for _, match := range matches {
43       if *xlink {
44         err := os.Symlink(match, filepath.Base(match))
45         panicOnErr(err)
46       }
47       fmt.Println(match)
48     }
49   } else {
50     for _, file := range flag.Args() {
51       ppath, err := filepath.Abs(file)
52       panicOnErr(err)
53       fileId, err := name2id(db, "file", ppath)
54       panicOnErr(err)
55
56       fmt.Printf("Tagging %s with %s\n", ppath, *tag)
57
58       tagMap(db, tagId, fileId)
59       panicOnErr(err)
60     }
61   }
62 }
63
64 func panicOnErr(err error) {
65   if err != nil {
66     panic(err)
67   }
68 }
```

database in line 23, where `dbInit()` creates the SQLite file if it does not already exist. If there are no files following a `--tag=...` option, line 40 uses `tagSearch()` to fetch all matching file paths from the database, and the `for` loop starting in line 42 prints them out. If `--xlink` is also set, line 44 creates a symlink to each matching photo file in the current directory respectively. This way it's easy to create, view, and process collections of photos with identical tags in a temporary directory.

If the user has passed one or more files to the tag specification, the `else` branch tags the files starting in line 49. To do this, line 53 first uses `name2id` to insert the name of the image file into the index (if it is not already there); it then calls `tagMap()` with the index number retrieved in line 58, which establishes the relation in the `tagmap` table.

For error handling, the listings use `panicOnErr()` (from line 64) in the usual way. This makes short work of errors and aborts the program. In production software, errors are usually handled by returning to higher levels instead.

You can compile the whole thing, including all referenced GitHub packages, with the commands in Listing 4. The result is the binary `idb`, which offers you all of the functions discussed in this column.

Outlook

The `idb` tool assigns tags to photos, but it could also be used in other ways to mark any sort of file in the filesystem, such as important configuration files whose paths people always forget or the video files you last edited in the depths of the directory hierarchy. Once tagged, `idb` quickly finds them when needed, based on the tag used.

Listing 4: Generate Binary

```
$ go mod init idb
$ go mod tidy
$ go build idb.go db.go dbinit.go
```

The `idb` tool could easily be expanded. What about using a metadata table to store the GPS coordinates of all your photos? Or what about a transaction table that lets you untag accidentally tagged images with `--undo`? As always, there are no limits to creativity with DIY tools. ■■■

Info

[1] idb: *https://www.perlmeister.com/idb.html*

[2] "Programming Snapshot – Go and Fyne" by Mike Schilli, *Linux Magazine*, issue 254, January 2022, pp. 40-45

shop.linuxnewmedia.com

REAL SOLUTIONS *for* REAL NETWORKS

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

It isn't all Windows anymore – and it isn't all Linux.

A router is more than a router. A storage device is more than a disk. And the potential intruder who is looking for a way around your security system might have some tricks that even you don't know.

Keep your network tuned and ready for the challenges with the one magazine that is all for admins.

Improve your admin skills with practical articles on:

- **Security**
- **Cloud computing**
- **DevOps**
- **HPC**
- **Storage and more!**

ADMIN
Network & Security

SUBSCRIBE NOW
SAVE 30%

REAL-WORLD
PROBLEMS
SOLVED!

shop.linuxnewmedia.com

Graphic front ends for SSH

Distant View

Bring your remote management into focus with these graphical front ends for SSH. *By Erik Bärwaldt*

Network administrators rely on Secure Shell (SSH) every day to establish secure connections over unsecured networks. The only requirement is that the computer you wish to control must be running an SSH server service. The most popular SSH service on Linux is OpenSSH.

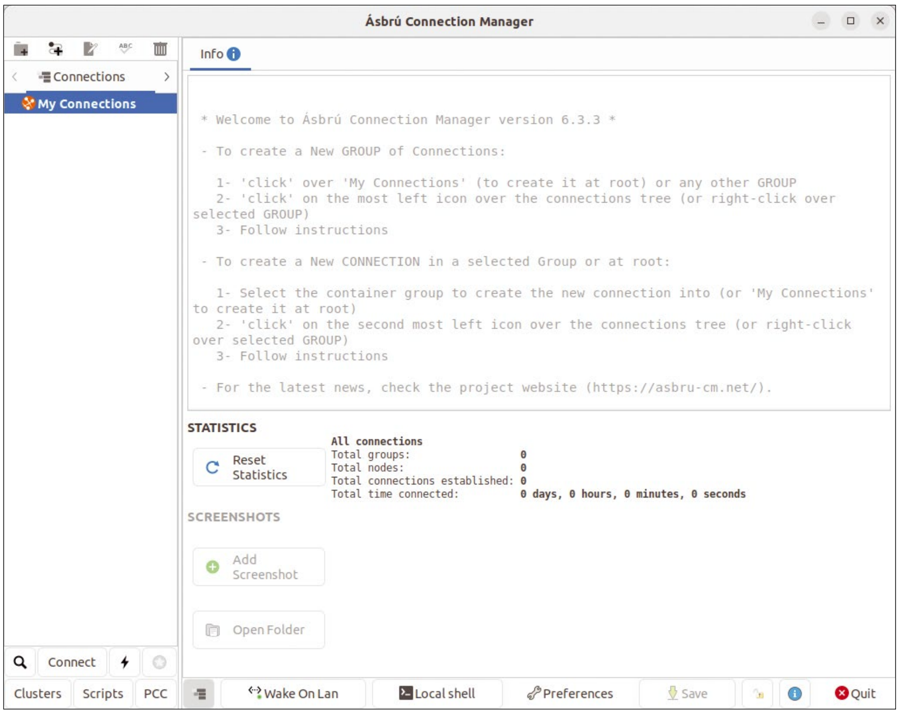
The SSH protocol also supports port forwarding, which allows you to tunnel connections for other applications, and SSH lets web admins upload files to a web server using secure SFTP and SCP connections.

Not in the Running

While many graphical SSH front ends have been created for Linux over the years, many of them have not been maintained for a long time. In addition to KSSH [4], which was developed for the KDE desktop, these include the ssh-gui [5] and secpanel [6] tools. While most distributions have banned ssh-gui and KSSH from their package sources by now, secpanel can still be found in the repos of Debian, Ubuntu, Fedora, and various BSD systems. However, like KSSH and ssh-gui, secpanel has not seen any updates for years, so we refrained from reviewing it.

SSH typically requires a terminal window. However, as you increase the number of servers and services managed through SSH connections, you also increase the risk. In such cases, it is better not to use the same usernames and passwords for all remote systems.

For convenient access, you can instead use a graphical SSH front end that lets you store the connection data of the individual terminal devices so that manual authentication is not required. SSH graphical user interfaces (GUIs) are a valuable aid, especially if you have to

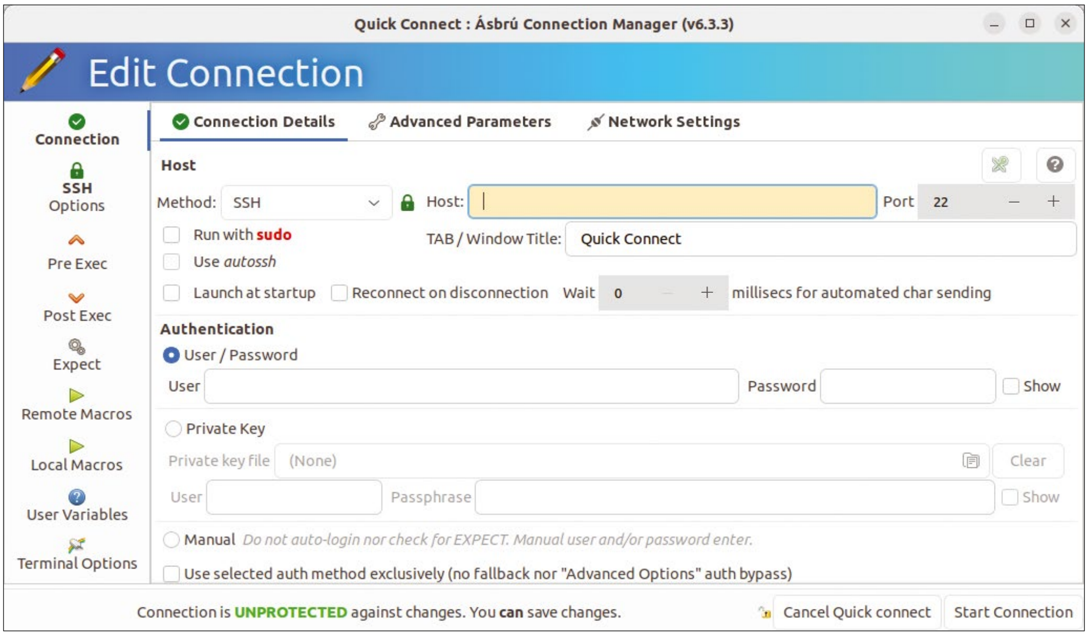


```
 * Welcome to Ásbrú Connection Manager version 6.3.3 *

 - To create a New GROUP of Connections:

   1- 'click' over 'My Connections' (to create it at root) or any other GROUP
   2- 'click' on the most left icon over the connections tree (or right-click over
selected GROUP)
   3- Follow instructions

 - To create a New CONNECTION in a selected Group or at root:

   1- Select the container group to create the new connection into (or 'My Connections'
to create it at root)
   2- 'click' on the second most left icon over the connections tree (or right-click
over selected GROUP)
   3- Follow instructions

 - For the latest news, check the project website (https://asbru-cm.net/).
```

```
All connections
Total groups:                 0
Total nodes:                  0
Total connections established: 0
Total time connected:         0 days, 0 hours, 0 minutes, 0 seconds
```

Figure 1: The Ásbrú Connection Manager window looks a little confusing at first glance.

Photo by mostafa meraji on Unsplash



Method: SSH
Host:
Port 22
☐ Run with **sudo**
☐ Use *autossh*
TAB / Window Title: Quick Connect
☐ Launch at startup ☐ Reconnect on disconnection Wait 0 millisecs for automated char sending

● User / Password
User
Password
☐ Show

○ Private Key
Private key file (None)
Clear
User
Passphrase
☐ Show

○ Manual *Do not auto-login nor check for EXPECT. Manual user and/or password enter.*

☐ Use selected auth method exclusively (no fallback nor "Advanced Options" auth bypass)

Connection is **UNPROTECTED** against changes. You can save changes.

Figure 2: You configure Ásbrú Connection Manager in its very detailed dialogs.

maintain groups of servers on the intranet. Linux has a long history with SSH front ends. This article takes a look at Ásbrú Connection Manager [1], EasySSH [2], and PuTTY [3]. For other options through the years, see the box entitled "Not in the Running."

Ásbrú Connection Manager

Ásbrú Connection Manager [1], a long-established free software tool, is licensed under GPLv3. The project's website provides detailed information about the installation. The routine seamlessly integrates the application into the menu structure of the existing working environment, allowing you to conveniently call Ásbrú Connection Manager with a mouse click.

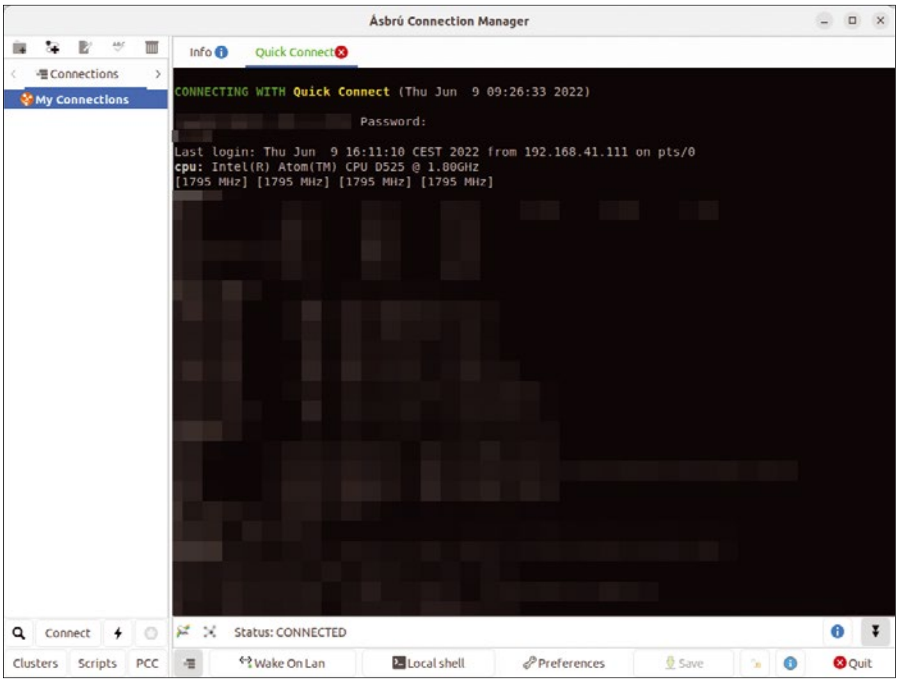
The capabilities of Ásbrú Connection Manager go well beyond managing and establishing SSH connections. You can execute random commands not only when activating a configured connection, but also after terminating a session. KeePassX integration also enables management of the stored authentication data. Thanks to modern encryption methods, there are no security worries involved with this. The Ásbrú Connection Manager is also suitable for use across a proxy server and has Wake-on-LAN capabilities. Written in Perl, Ásbrú Connection Manager provides a modern tabbed interface that lets users maintain multiple connections simultaneously if needed.

At first glance, the program window's design appears to be a little unconventional and confusing (Figure 1). After launching the software, you will find a large display area for information and statistical data in the right pane. On the left, the Ásbrú Connection Manager lists the individual configured connections one below the other. At the very bottom of the window, several buttons provide quick access to the most important functions. At the top left, there are five more buttons for managing the connection entries. In addition, in the desktop environment's system tray, you will find a small icon with a network connector symbol. You can use this to control the program during a session.

The vertical pane on the left is where you create individual connections and groups. After right-clicking on the default *My Connections* option and then selecting *Add Group* in the context menu that appears, you first need to create and name a group in a separate dialog.

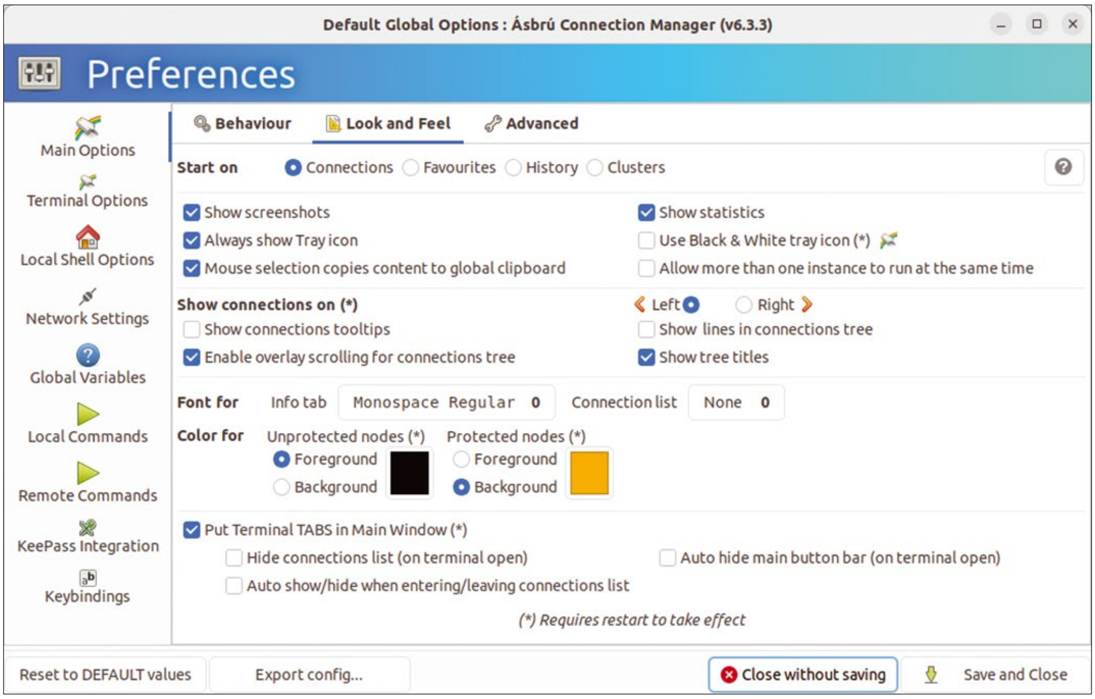
Another right-click on the newly created group lets you add individual connections. In the context menu that opens, select the *Add Connection* entry and assign a meaningful name for the connection. A comprehensive settings window then appears where you can enter the parameters for opening the connection in the *Connection Details* tab (Figure 2).

Choose between different protocols in the *Method:* selection field. The graphical client can work simultaneously with a wide variety of protocols, regardless of



```
CONNECTING WITH Quick Connect (Thu Jun  9 09:26:33 2022)
[redacted] Password:
Last login: Thu Jun  9 16:11:10 CEST 2022 from 192.168.41.111 on pts/0
cpu: Intel(R) Atom(TM) CPU D525 @ 1.80GHz
[1795 MHz] [1795 MHz] [1795 MHz] [1795 MHz]
[redacted]
```

Figure 3: Virtual terminals help you keep multiple connections open simultaneously in Ásbrú Connection Manager.



Start on ● Connections ○ Favourites ○ History ○ Clusters

☑ Show screenshots
☑ Always show Tray icon
☑ Mouse selection copies content to global clipboard
☑ Show statistics
☐ Use Black & White tray icon (*)
☐ Allow more than one instance to run at the same time

Show connections on (*) Left ● ○ Right
☐ Show connections tooltips
☑ Enable overlay scrolling for connections tree
☐ Show lines in connections tree
☑ Show tree titles

Font for Info tab `Monospace Regular 0` Connection list None 0

Color for Unprotected nodes (*) ● Foreground ○ Background ■ Protected nodes (*) ○ Foreground ● Background ■

☑ Put Terminal TABS in Main Window (*)
☐ Hide connections list (on terminal open)
☐ Auto show/hide when entering/leaving connections list
☐ Auto hide main button bar (on terminal open)

(*) Requires restart to take effect

Figure 4: Ásbrú Connection Manager offers users very detailed global settings.

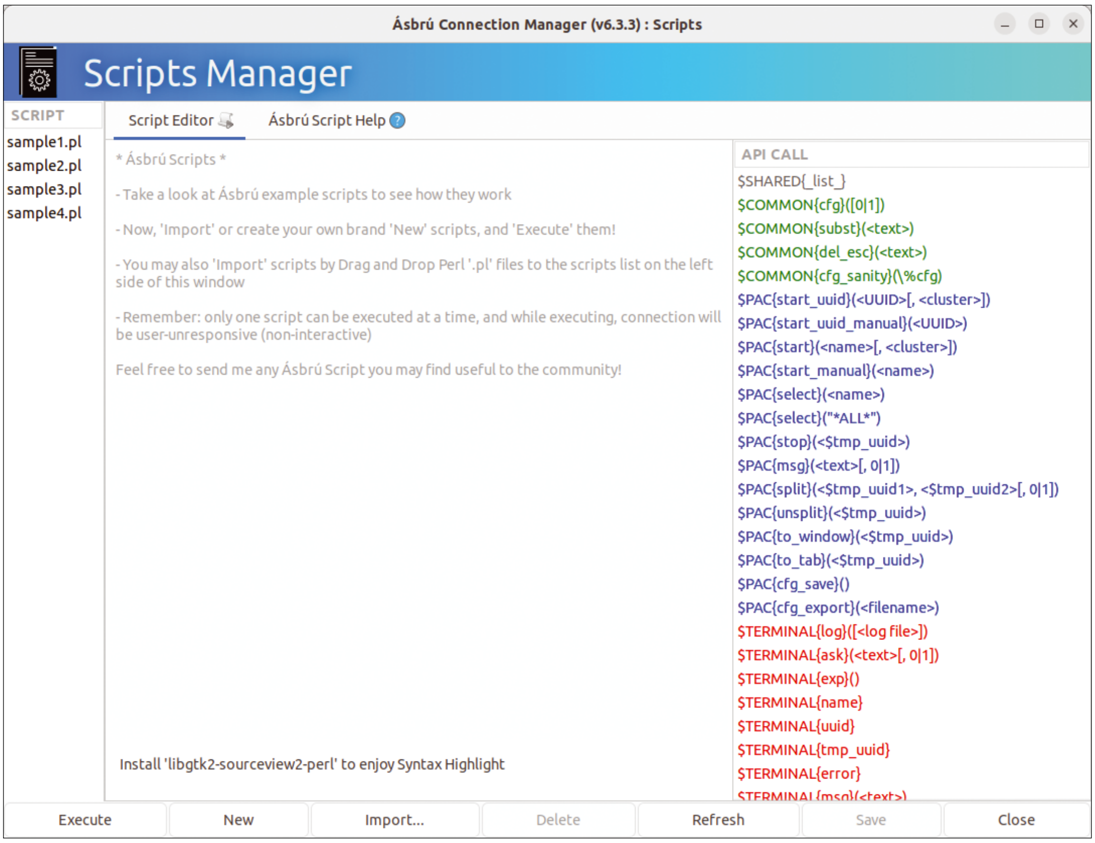
whether the other side transmits graphical content or just a prompt. Flexible deployment in heterogeneous environments is therefore possible at any time with Ásbrú Connection Manager. In addition, there is the option to choose between conventional authentication with a username and password and using a cryptographic key pair.

After completing the basic configuration for the particular connection in the *Advanced Parameters* tab, you can specify commands to be executed before or after the connection is established, if necessary. Once you have set all the options, complete the entry by clicking the *Save and Close* button bottom right in the window.

You can now open a new tab by double-clicking on the corresponding entry on the left side of the connection bar. The Connection Manager will either open the connection automatically or you will have to enter an administrator password in a separate dialog if the connection requires specific authorizations. If this is a terminal connection, the terminal appears in its own tab in the large window pane on the right (Figure 3).

Each click on one of the preconfigured connections opens a new tab, so you can manage multiple services simultaneously by simply switching the active tab. Because the tool labels the tabs with the cleartext names you assigned during the configuration step, the individual connections can easily be identified. A tab with a green text indicates an open connection, while red text notifies you that the connection attempt failed. If your connection attempt has failed, right-click on the machine in the connection list on the left and select *Edit connection* from the context menu. You can then edit the configuration in the same dialog used for entering a connection.

In the *Info* tab on the very left in the main window pane, you will also see



sample1.pl
sample2.pl
sample3.pl
sample4.pl

```
* Ásbrú Scripts *

- Take a look at Ásbrú example scripts to see how they work

- Now, 'Import' or create your own brand 'New' scripts, and 'Execute' them!

- You may also 'Import' scripts by Drag and Drop Perl '.pl' files to the scripts list on the left
side of this window

- Remember: only one script can be executed at a time, and while executing, connection will
be user-unresponsive (non-interactive)

Feel free to send me any Ásbrú Script you may find useful to the community!
```

Install 'libgtk2-sourceview2-perl' to enjoy Syntax Highlight

```
$SHARED{_list_}
$COMMON{cfg}([0|1])
$COMMON{subst}(<text>)
$COMMON{del_esc}(<text>)
$COMMON{cfg_sanity}(\%cfg)
$PAC{start_uuid}(<UUID>[, <cluster>])
$PAC{start_uuid_manual}(<UUID>)
$PAC{start}(<name>[, <cluster>])
$PAC{start_manual}(<name>)
$PAC{select}(<name>)
$PAC{select}("*ALL*")
$PAC{stop}(<$tmp_uuid>)
$PAC{msg}(<text>[, 0|1])
$PAC{split}(<$tmp_uuid1>, <$tmp_uuid2>[, 0|1])
$PAC{unsplit}(<$tmp_uuid>)
$PAC{to_window}(<$tmp_uuid>)
$PAC{to_tab}(<$tmp_uuid>)
$PAC{cfg_save}()
$PAC{cfg_export}(<filename>)
$TERMINAL{log}([<log file>])
$TERMINAL{ask}(<text>[, 0|1])
$TERMINAL{exp}()
$TERMINAL{name}
$TERMINAL{uuid}
$TERMINAL{tmp_uuid}
$TERMINAL{error}
$TERMINAL{msg}(<text>)
```

Figure 5: Using Perl scripts, you can add automated functions to Ásbrú Connection Manager.

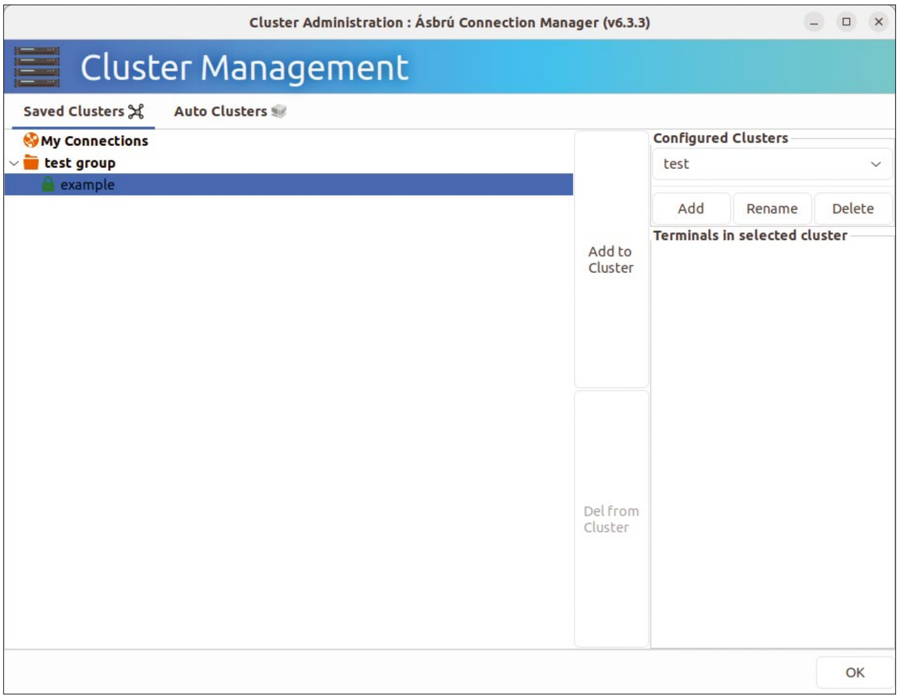


Figure 6: Ásbrú Connection Manager uses clusters to address several servers simultaneously.

Listing 1: Install and Start EasySSH

```
$ flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo
$ flatpak install flathub com.github.muriloventuroso.easyssh
$ flatpak run com.github.muriloventuroso.easyssh
```

some statistical information for the existing connections.

Pressing the *Preferences* button bottom center in the program window lets you set basic options that are valid for all tabs. In different categories in the Preferences dialog, you can configure the appearance of the application and the displayed terminals, the network settings, and the size of the windows (Figure 4). You also can customize individual functions of the local shell if needed.

KeePass Integration in the left sidebar lets you to integrate the KeePassX password manager with Connection Manager so that you can access its database while working. Like in the connection settings dialog, you finish your work in the Preferences dialog by clicking on the *Save and Close* button located bottom right.

Network Settings lets you modify the global network settings. If necessary, you can specify a jump server that supports an automated connection via the SSH console to the individual servers on the intranet. The jump server acts much like a proxy and has the role of an intermediary between the client and the individual SSH servers.

Various keyboard mappings are predefined in Ásbrú Connection Manager for fast function calls. You can adjust them to suit your needs via *Keybindings*.

Once you have completed all the settings, click *Save and Close* to finish your work. If you want to make changes to the configuration later, you can reach the settings dialogs for the selected connection by pressing the button with the pencil icon in the top left corner of the program window.

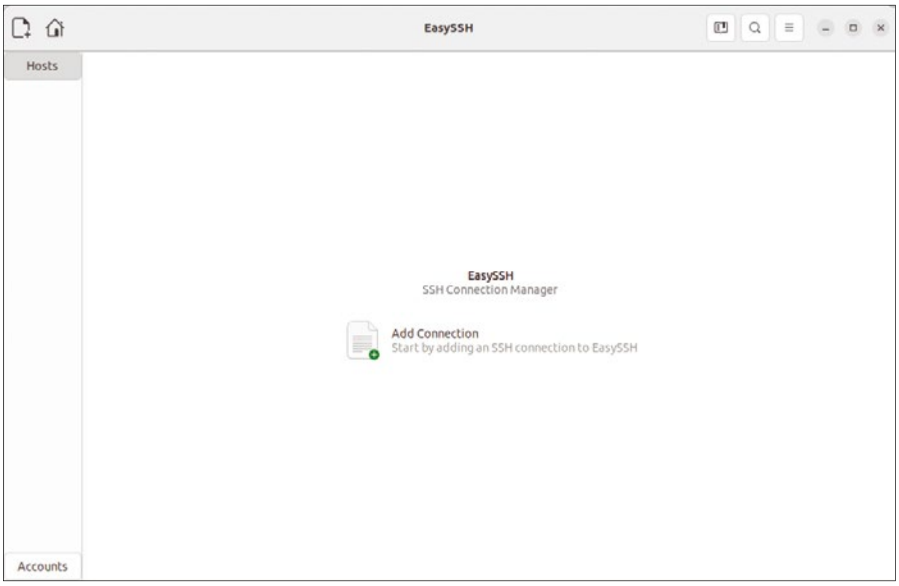
To execute commands automatically, click the *Scripts* button bottom left in the program window. In the *Scripts Manager* that opens, create Perl scripts and modify them if needed. The developers provide documentation on the available commands and their usage directly in the window (Figure 5).

One of Ásbrú Connection Manager's particularly convenient features is its ability to group several similar servers in clusters. You then can execute the same commands simultaneously on all the servers in the cluster. You can manage the cluster shell in the *Cluster Management* dialog, which can be accessed by pressing the *Clusters* button bottom left in the main window. You can then combine the individual servers from the existing infrastructure (Figure 6).

EasySSH

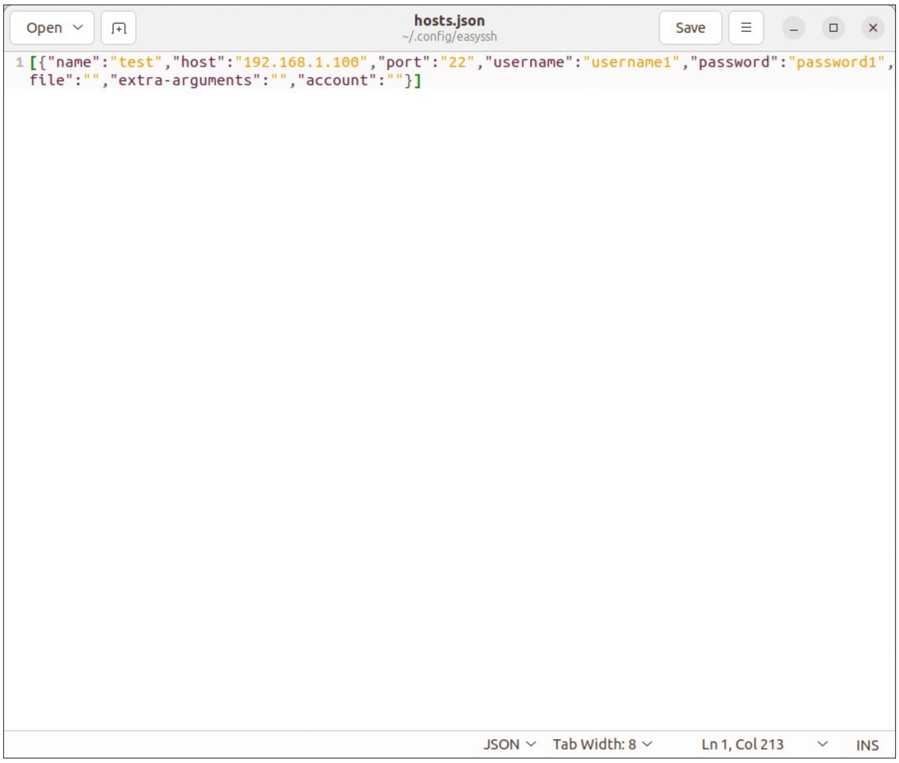
The EasySSH project [2], which was launched only a few years ago, sets out to manage SSH connections in the simplest possible way.

So far, only a few distributions actually have the software in their repositories (e.g., Debian, Ubuntu, and Solus). On other Linux derivatives, you need to install EasySSH as a Flatpak, which you



Start by adding an SSH connection to EasySSH

Figure 7: EasySSH comes with a self-explanatory interface.



```
[{"name":"test","host":"192.168.1.100","port":"22","username":"username1","password":"password1",
file":"","extra-arguments":"","account":""}]
```

Figure 8: A massive vulnerability: EasySSH stores all the connection data in plain text.

can pick up from the project's GitHub page. You can install it with the code in the first two lines of Listing 1. Then start the software in the terminal with the command from the last line of Listing 1.

After the call, a three-part window opens (Figure 7). The titlebar contains some controls, while a column on the left side of the window shows the target system entries. If necessary, you can divide the entries into groups to improve the overview. On the right, a large area remains free for the remote computer systems' terminals that you want to open.

To configure an initial connection to a remote SSH server, either press the *Add connection* button in the middle of the large pane or press the plus button in the top left corner of the titlebar. A settings dialog then appears on the right, where you can select the name of the server, its IP address, and the port. You specify the authentication data in the same dialog box. After clicking *Advanced*, you can configure some additional settings relating to the terminal's appearance.

After adding a new connection, you can launch it by pressing *Connect*. EasySSH now displays the terminal of the connected SSH server in the right window pane. At the same time, a tab appears above the large right pane. If you need to manage multiple SSH servers, you can enable additional simultaneous SSH connections; EasySSH manages these in separate tabs. This means that you can quickly switch from one server to the next without having to close and reopen connections each time.

Select the *Settings* entry in the hamburger menu top right to adjust EasySSH's configuration. I recommend enabling the slider for data encryption.

EasySSH not only supports username and password-based authentication but also asymmetric keys. Using a key pair with a public and a private key significantly enhances security. You can add an existing SSH key to the application by checking the *Change password to key file* box in the connection settings dialog. Then, in the file manager that appears,

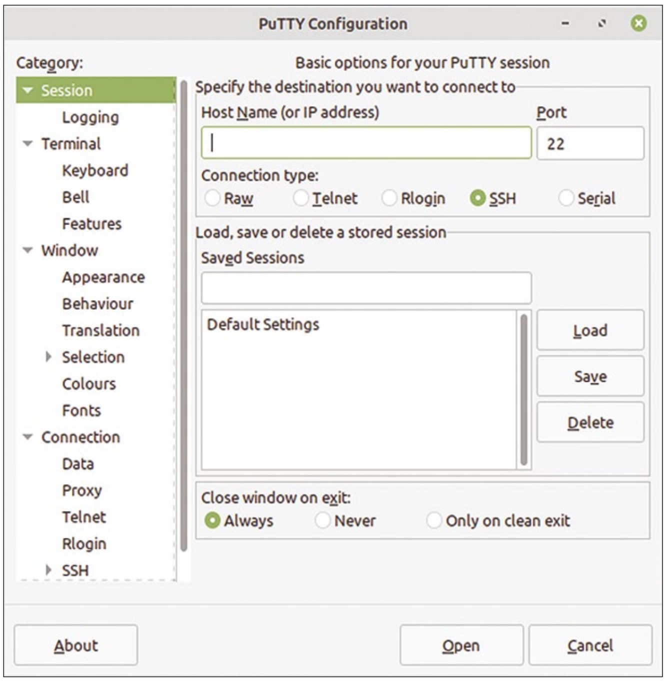


Figure 9: PuTTY looks a little old-fashioned.

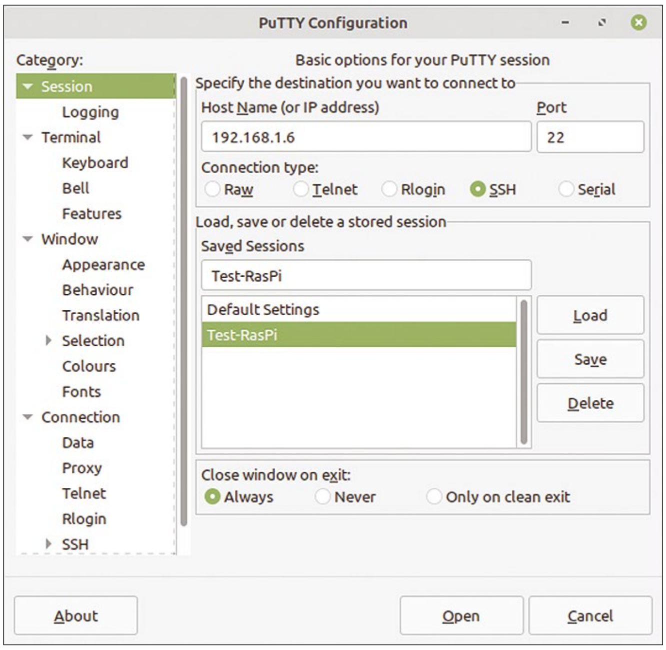


Figure 10: Profiles help PuTTY to connect to multiple servers.

Table 1: Graphical SSH Front End Features

	Ásbrú Connection Manager	EasySSH	PuTTY
License	GPLv3	GPLv3	MIT License
Functions			
Multiple simultaneous connections	Yes	Yes	No
Built-in virtual terminals	Yes	Yes	No
Clusters	Yes	No	No
Groups	Yes	Yes	No
Authentication via key pair	Yes	Yes	Yes
X11 forwarding	Yes	Yes	Yes

select the required `.pub` public key file. This is then used in combination with the private key to authenticate the current user.

EasySSH does have a serious vulnerability: It saves all the connection data in plain text (Figure 8). The `hosts.json` file contains all the access credentials including unencrypted passwords, IP addresses, and ports. In this respect, EasySSH can only be used in truly secure environments where access by unauthorized third parties can be ruled out.

PuTTY

PuTTY, from the Windows world, is one of the older graphical SSH front ends [3] with more than 20 years of development. Most distributions include it in their package sources.

After starting the application, you are first taken to a configuration window where you can create a new connection. A sidebar on the left of the dialog offers numerous settings for the appearance, the terminal displayed in the application, the protocol function configuration, and the connections (Figure 9).

PuTTY not only supports SSH-based access but also insecure protocols such as Telnet and rlogin. Like EasySSH and Ásbrú Connection Manager, PuTTY supports X11 forwarding: This means that you can use graphical applications on remote servers in addition to text-based ones. Once you have made the necessary settings for the target server, click *Open* to start the SSH session. PuTTY now opens an X terminal where you can enter the authentication data for the SSH server.

To avoid having to manually enter regularly-used connection parameters at the beginning of each session, you can save the settings you have made in profiles that you can call up again later (Figure 10). Use the first settings page, under *Saved Sessions*, to assign a name for the connection to be saved, then press the *Save* button to save the connection. The individual connections are listed in the corresponding window segment below, where you can select an entry and load it into the configuration dialog by pressing *Load*. Then open the session by pressing *Open* at the bottom of the window.

In addition to simple credentials, PuTTY also supports authentication via asymmetric encryption. You can enter the keys for the key exchange procedure in the corresponding configuration window dialogs. PuTTY additionally provides a key generator, which you run as a CLI application in the terminal on Linux. When done, integrate the generated keys into the application. Note that PuTTY uses a proprietary key format; the keys generated here cannot be used in other applications.

Conclusions

The graphical SSH front ends discussed in this article all do their jobs without problems, but there are significant differences in the features each front end offers (see Table 1). PuTTY shows its age visually as well as in terms of the functions it offers: You will not find tabs for open connections or integrated virtual terminal windows. In addition, accessing remote systems with hopelessly antiquated protocols such as rlogin or Telnet also seems obsolete.

EasySSH lets users establish connections very quickly and without detours via detailed settings dialogs, but – as a Flatpak application – it is very slow. More importantly, EasySSH has a significant security hole caused by storing the connection data without encryption in plain text on the client, effectively inviting attackers to steal the access credentials.

Ásbrú Connection Manager offers the most balanced and state-of-art, graphical SSH front end. Its modern interface and ability to work with clusters makes Ásbrú Connection Manager the tool of choice for professional use. ■■■

Info

[1] Ásbrú Connection Manager: *https://www.asbru-cm.net*

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

[3] PuTTY: *https://www.putty.org*

[4] KSSH: *https://sourceforge.net/projects/kssh/*

[5] ssh-gui: *https://sourceforge.net/projects/ssh-gui/*

[6] secpanel: *https://sourceforge.net/projects/secpanel/*

imgp: Batch editing for image files
LINUX MAGAZINE
ISSUE 252 – NOVEMBER 2021
Locked Down!
New tools for tighter security
Miniature Photography with a webcam and Rasp Pi
ISSUE 254 – JANUARY 2022
Phone Hacks
No more vendor updates? Put a free OS on your smartphone and keep it calling
EasyNAS
Turn your old computer into a network storage device
MergerFS
Join disks, volumes, and arrays into a single file system
Repurpose Your Old Router
3 easy projects for makers!
ISSUE 253 – DECEMBER 2021
OpenBSD
Is this meticulous
Machine Learning Workshop
Find a lost visitor in your house
ISSUE 255 – FEBRUARY 2022
Break It to Make It
Code safer with automated fuzz testing
Clean Up your Photo Metadata
ISSUE 256 – MARCH 2022
MakerSpace Create a Light Show with the Rasp Pi DMX Module
ISSUE 257 – APRIL 2022
Encryption
Protect your secrets
Maker Makeover
Add a GPIO to your Linux desktop or laptop system
ISSUE 258 – MAY 2022
Clean IT
SuperTuxKart: Winning tips for this classic free game
ISSUE 261 – AUGUST 2022
USB Boot
Stash multiple boot images on a single stick
Zeit: Easy automation with this cool cron GUI
Storyboards in Krita
Shell Redirects: Better scripting with better pipes
Dosbian: Run retro DOS games on your Rasp Pi
Finding and fixing broken links
FOSSPICKS: DECOMPILE YOUR CODE WITH SNOWMAN
WWW.LINUX-MAGAZINE.COM
TILING WINDOW MANAGERS
THIS CLASSIC TOOL IS ATTRACTING A NEW GENERATION OF FANS
ISSUE 259 – JUNE 2022
Zero Trust
Reimagining security for today's threats
Bash Tricks
Roll your own download mana
ISSUE 260 – JULY 2022
Privacy
Keeping your secrets secret
Devuan: Defiant Debian fork with old
FREE DVD

Linux Magazine **is your guide to the world of Linux. Look inside for advanced technical information you won't find anywhere else!**

Expand your Linux skills with:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- Insightful news on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

If you want to go farther and do more with Linux, subscribe today and never miss another issue!

WHAT'S NEW IN UBUNTU 22.04
LINUX MAGAZINE
ISSUE 262 – SEPTEMBER 2022
FREE DVD
Beyond 5G
Imagining an open future for mobile networks
TANTALIZING FOSS TOOLS
FREE ARCHIVE DVD
ALL 262 ISSUES ON A SEARCHABLE DISC!
HUGE SAVINGS! £39.90 VALUE!
LINUX MAGAZINE
ISSUE 263 – OCTOBER 2022
Build an IoT Linux
Shrink your OS to fit the device
Tools for Detecting Fake News
ImageMagick Tricks: Paint an image in a script
Read RFID Tags with a Rasp Pi
Create a User-Friendly Front End for Your Bash Scripts
0 A.D.: Bring the past to life with this free strategy game
10 MORE KILLER FREE TOOLS!
WWW.LINUX-MAGAZINE.COM

Subscribe now!

shop.linuxnewmedia.com/subs

GET IT NOW!
FAST DELIVERY WITH OUR PDF EDITION

MakerSpace

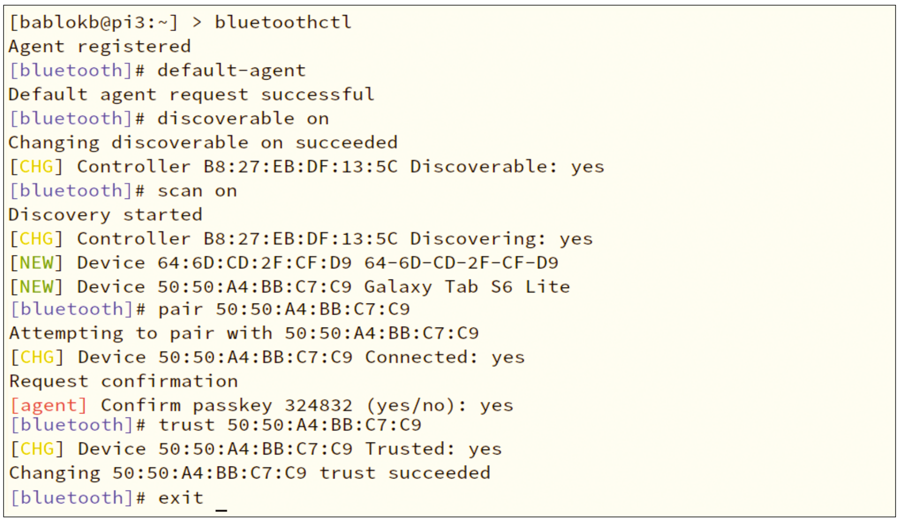
Connect Pi devices and a smartphone with Bluetooth

Small Talk

We use a Raspberry Pi, a Pi Pico, and a smartphone to communicate over Bluetooth. *By Bernhard Bablok*

Because the Raspberry Pi comes with both WiFi and Bluetooth, most programs choose to rely on WiFi, with Bluetooth being more of a wallflower. However, the Pi Pico mixes things up and makes Bluetooth a desirable option.

Bluetooth should be a familiar technology by way of your smartphone, and this topic has been investigated for the Raspberry Pi in a previous article [1], so I will be sticking to the bare minimum in terms of the basic technology in this article. The focus here is on various scenarios in which the Raspberry Pi, Pi Pico, and smartphones use Bluetooth to communicate. The Pico stands in for almost any microcontroller, as long as it supports serial communication.



```
[bablokb@pi3:~] > bluetoothctl
Agent registered
[bluetooth]# default-agent
Default agent request successful
[bluetooth]# discoverable on
Changing discoverable on succeeded
[CHG] Controller B8:27:EB:DF:13:5C Discoverable: yes
[bluetooth]# scan on
Discovery started
[CHG] Controller B8:27:EB:DF:13:5C Discovering: yes
[NEW] Device 64:6D:CD:2F:CF:D9 64-6D-CD-2F-CF-D9
[NEW] Device 50:50:A4:BB:C7:C9 Galaxy Tab S6 Lite
[bluetooth]# pair 50:50:A4:BB:C7:C9
Attempting to pair with 50:50:A4:BB:C7:C9
[CHG] Device 50:50:A4:BB:C7:C9 Connected: yes
Request confirmation
[agent] Confirm passkey 324832 (yes/no): yes
[bluetooth]# trust 50:50:A4:BB:C7:C9
[CHG] Device 50:50:A4:BB:C7:C9 Trusted: yes
Changing 50:50:A4:BB:C7:C9 trust succeeded
[bluetooth]# exit _
```

Figure 1: Pairing and trust with `bluetoothctl`. After power on and scanning, you will see many different devices and their MAC addresses depending on the environment.

Basic Technology

The two basic prerequisites for Bluetooth communication between partners is pairing and trust between the parties. After you establish both once, the devices usually remember the trust relationship. After pairing, each of the two partners can establish a connection. The initiator is referred to as the host and the other partner as the device. However, each component can assume either role.

The default user *pi* on the Raspberry Pi should be a member of the *bluetooth* and *dialout* groups; you can check their status with the `id` command. If you are missing membership in one or both groups, make the change with the command

```
$ sudo usermod -a ⮐
  -G bluetooth,dialout pi
```

and then log off and log on again.

The counterpart must be visible for pairing to succeed. On smartphones, the required settings are in the Bluetooth section of the preferences. On the Raspberry Pi, use the `bluetoothctl` command (Figure 1), which starts its own small shell with a limited command set. The `help` command calls up an overview.

Lead Image © Andrey Kiselev, 123RF.com

Pressing Tab completes commands and Bluetooth MAC addresses, which saves a great deal of typing.

System Requirements

In addition to individual pairing, the Raspberry Pi must meet a few system requirements. Communication between the smartphone, Pi Pico, and Raspberry Pi relies on the serial port protocol (SPP), which is based on Radio frequency communication (RFCOMM) over the Logical Link Control and Adaptation Protocol (L2CAP). This protocol stack is part of legacy Bluetooth. Other profiles such as the File Transfer Protocol (FTP) or Object Exchange (OBEX) push protocol also rely on RFCOMM. OBEX is used by smartphones for direct file exchange.

For RFCOMM to be available on the Raspberry Pi, you need to start the Bluetooth daemon with the `-C` compatibility option. To do this, copy the service definition from `/lib/systemd/system/bluetooth.service` to `/etc/systemd/system`. There, you replace the `ExecStart` statement with:

```
ExecStart=⮒
  /usr/lib/bluetooth/bluetoothd -C
ExecStartPost=/usr/bin/sdptool add SP
```

The second line adds SPP to the service catalog of available protocols. Now, enter

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart ⮒
  bluetooth.service
```

to restart the daemon.

Blue Keyboard

The first sample application is a media player on a TV located in the living room. If it fails to work correctly, just restart the server process. Independently, you might also need to check a couple of important parameters. Instead of using an editor and SSH, you can use a smartphone or tablet. Android devices run the Serial Bluetooth Terminal [2] app for this purpose; a similar app is available for iOS, as well.

You can access the paired devices from the app's menu; clicking on a device starts the connection setup. For this to work, however, the Raspberry Pi must wait for a connection. You can initiate this with the command:

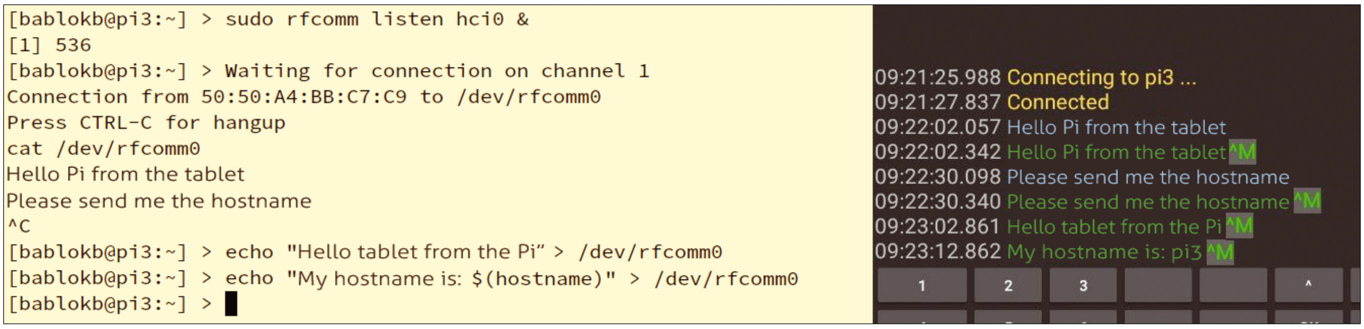
```
sudo rfcomm listen hci0 &
```

The ampersand (`&`) at the end bundles the command off into the background and frees up the console for input again.

Figure 2 shows the principle of communication after the connection is established. The tablet (right) sends commands (in blue), and the Raspberry Pi uses `cat` to read them from `/dev/rfcomm0` (left). This also works the other way around: The Rasp Pi uses `echo` to write to `/dev/rfcomm0`, and the output appears on the tablet (green). Once the connection is established, it doesn't matter who set it up: The data flow is always bidirectional. Because of an error in the app, the commands from the tablet are echoed in the response from the Raspberry Pi.

To make the steps you performed manually on the Raspberry Pi shown in Figure 2 automatic, set up a systemd service with the definition from Listing 1. The `watch` argument in line 7 works like `listen`. After establishing the connection, RFCOMM starts the `/usr/local/sbin/bt-control.sh` script. With a connection failure, `watch` ensures that RFCOMM returns to `listen` mode.

You can see the corresponding control program in Listing 2. The example is a



```
[bablokb@pi3:~] > sudo rfcomm listen hci0 &
[1] 536
[bablokb@pi3:~] > Waiting for connection on channel 1
Connection from 50:50:A4:BB:C7:C9 to /dev/rfcomm0
Press CTRL-C for hangup
cat /dev/rfcomm0
Hello Pi from the tablet
Please send me the hostname
^C
[bablokb@pi3:~] > echo "Hello tablet from the Pi" > /dev/rfcomm0
[bablokb@pi3:~] > echo "My hostname is: $(hostname)" > /dev/rfcomm0
[bablokb@pi3:~] > █
```

```
09:21:25.988 Connecting to pi3 ...
09:21:27.837 Connected
09:22:02.057 Hello Pi from the tablet
09:22:02.342 Hello Pi from the tablet^M
09:22:30.098 Please send me the hostname
09:22:30.340 Please send me the hostname^M
09:23:02.861 Hello tablet from the Pi^M
09:23:12.862 My hostname is: pi3^M
```

Figure 2: Communication flow between the Raspberry Pi and an Android tablet over Bluetooth.

Listing 1: Service Definition

```
[Unit]
Description=BT-Control service
After=bluetooth.service
Requires=bluetooth.service

[Service]
ExecStart=/usr/bin/rfcomm watch rfcomm0 1 /usr/local/sbin/bt-control.sh

[Install] 10: WantedBy=multi-user.target
```

Listing 2: Control Program

```
01 #!/bin/bash
02
03 process_request() {
04   case "$1" in
05     "hostname") hostname -f >&3 ;;
06     "mem")      free -h >&3 ;;
07     "disk")     df -h   >&3 ;;
08     *)          echo "Unknown command" >&3 ;;
09     esac
10 }
11
12 # --- Main program -----------------------------
13
14 # Open device for reading and writing
15 exec 3<>/dev/rfcomm0
16
17 while true; do
18   if read -u 3 request; then
19     process_request "$request"
20   else
21     break
22   fi
23 done;
24
25 # Close file descriptor
26 exec 3>&-
```

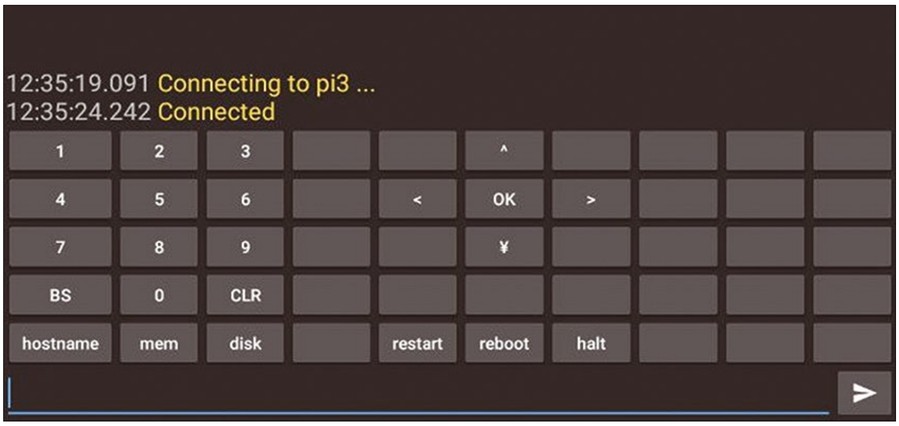
simple shell script, but any other programming language would be just as good. The script reads continuously from the RFCOMM interface (line 18) and processes the predefined commands in the `process_request()` function starting in line 3.

The Android app makes the control task very simple by letting you assign commands directly to buttons (Figure 3). The number of lines is defined in the settings. A long press on a button lets you enter edit mode, and you can change the title, command, and format. However, individual buttons cannot be hidden, nor can the font size be changed.

Quite a few apps in the Google Play store are based on the same RFCOMM technology. For example, of special interest is the Bluetooth Control Panel [3], which dynamically creates an interface on the basis of data sent to it by the Raspberry Pi or a microcontroller.

Contactless Data

In the first use case, the smartphone was the host and the Raspberry Pi was the device. Now, the small-board computer (SBC) is going to assume the host role. (SBC will always refer to the Raspberry Pi, not the Pi Pico.) A typical use case is data logging over Bluetooth. For this purpose, the microcontroller writes its output to the serial port in the usual way, but instead of a cable, an HC-05 serial Bluetooth breakout module is connected (Figure 4). This device implements universal asynchronous receiver/transmitter (UART) communication over Bluetooth. The big advantage is that you don't have to change the program on the microcontroller because it already writes its data to the serial port.



12:35:19.091 Connecting to pi3 ...
12:35:24.242 Connected

Figure 3: The Android app supports quick entry of commands with predefined buttons.

The breakout comes in different versions. Exactly which you have hardly matters in the end. The board typically has a button, an LED, and a voltage converter, which makes the power supply flexible. However, the signal level on the RX/TX lines is always 3.3V, so if you connect an Arduino, you should adjust the voltage with a resistance division circuit (5V to 3.3V) between the Arduino TX pin and the module RX pin.

The HC-05 supports the command and data modes. The first mode is for configuration; for new devices, it is important to complete the steps described in the "Configuring HC-05" box. You also need to pair the Raspberry Pi and the breakout once, as described earlier, remembering the MAC address of the HC-05.

After completing the preparation work, the application is child's play. Connect the HC-05 to the microcontroller; it outputs its data in the usual way with a `print` or `printf` function. You now connect the Raspberry Pi to the HC-05 and redirect incoming data to a file with `cat`:

```
$ sudo rfcomm bind hci0 ➥
  <aa>:<bb>:<cc>:<dd>
$ cat /dev/rfcomm0 > <MyData>.log
```

The value `<aa>:<bb>:<cc>:<dd>` is the MAC address of the breakout. If the microcontroller also processes input, commands can be sent (e.g., to `/dev/rfcomm0`) by calling a terminal program with the device file as an argument.

Instead of the Raspberry Pi, you can connect to the HC-05 from the Android app, which proves useful for a quick test, because you can immediately see whether the Pi Pico is sending data. Also, the Raspberry Pi or app can send data to the microcontroller (e.g., to turn LEDs and motors on/off, etc.) by the HC-05. However, the microcontroller needs to both write to the serial interface, and read from it.

Pico as a Host

The previous scenario showed how the Raspberry Pi fetched data from the Pi Pico, and it works well as long as the microcontroller is already running and the connection from the Raspberry Pi to the Pico works. However, if the microcontroller is not running, the Raspberry Pi constantly has to try to re-establish the connection in a loop.

Regardless, you have another problem: If the Pi Pico sends data only now and then (e.g., because of large intervals between reading sensors), then the Raspberry Pi can only reliably retrieve data by keeping the connection constantly open. In this case, it makes more sense to let the Raspberry Pi and the Pico swap roles so that the microcontroller actively creates the connection. The Raspberry Pi then runs a service that accepts connection requests and fields and stores the data.

For this scenario, the HC-05 must play host and establish the connection, which is precisely what can turn out to be a major obstacle because the breakout comes in different versions. Many postings online describe the procedure with firmware from 2010. However, all of my devices use a firmware version from 2017 that behaves differently when you get into the details. The procedure described here will not necessarily succeed with your device, but you can query the firmware version in command mode with the `AT+VERSION?` command.

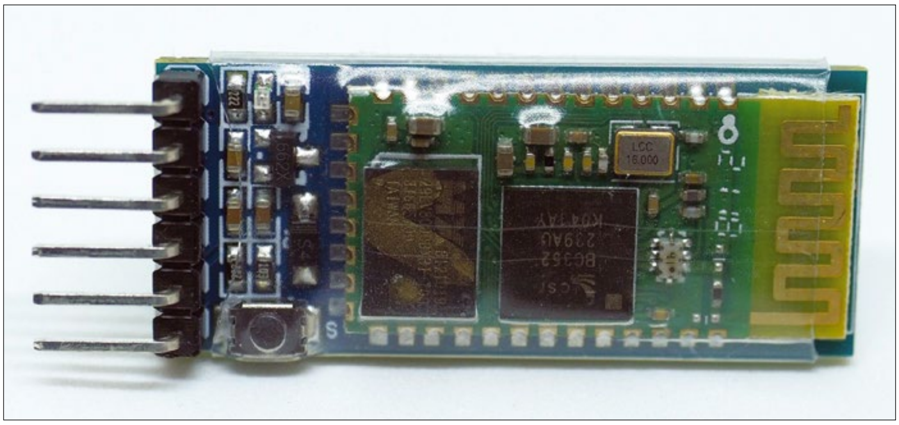


Figure 4: The HC-05 Bluetooth module for microcontrollers.

Also in command mode, you can change the role of the HC-05:

```
AT+CMODE=0
AT+ROLE=1
AT+BIND=4D65,4D,CA6612
```

The first line tells the chip to only connect to a specific address (given in line 3), the second line sets the host mode, and the third line binds the target device address – the Bluetooth MAC address of the Raspberry Pi, which you query with:

```
echo list | bluetoothctl
```

However, instead of using the colon-separated format seen earlier, the `BIND` command requires the information in a slightly different format, so `4D:65:4D:CA:66:12` becomes `4D65,4D,CA6612`.

Before rebooting, make sure the Raspberry Pi is ready to connect, and it works exactly as described in the first use case. Because microcontrollers usually send data and not commands, you do not need an application program with logic on the Raspberry Pi side, so in the simplest case, you only need a redirection to a file:

```
$ sudo rfcomm watch hci0 | ↩
  cat /dev/rfcomm0 >> pico-data.csv &
```

If you now start the Pico with the HC-05 connected, all data should end

Configuring the HC-05

The HC-05 almost always comes with the same preset username and password and with the transfer rate preconfigured for data mode at a meager 9,600 baud. You can change that with a few `AT` commands. The easiest approach is a USB-to-UART adapter that supports 3.3V. Alternatively, you can use the Raspberry Pi. However, its serial port occupies the system console, so you have to free this up first. To do so, remove the `console=serial0,115200` string in the `/boot/cmdline.txt` file and add the `enable_uart=1` line to `/boot/config.txt`.

After a reboot, connect GPIO14 (TX) to RX on the HC-05 and GPIO15 (RX) to TX and GND. Before applying the voltage (3.3V or 5V), press and hold the small button on the breakout. The LED should now flash slowly. The exact pattern depends on the version; in my case, it was a double flash at one-second intervals. Although the transfer rate in data mode is 9,600 baud, the HC-05 operates at 38,400 baud in command mode. You can connect to the device in the terminal program of your choice with

```
$ miniterm --eol CRLF -e /dev/ttyUSB0 38400
```

and then execute the following commands to configure the HC-05:

```
01 --- Miniterm on /dev/ttyUSB0  38400,8,N,1 ---
02 --- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
03 AT
04 OK
05 AT+NAME?
06 [...]
07 AT+NAME=PICO_42
08 [...]
09 AT+PSWD?
10 [...]
11 AT+PSWD="9999"
12 AT+UART?
13 [...]
14 AT+UART=115200,0,0
```

Pay attention to case. The first command should return *OK* (line 4) – if it does not, then something is wrong with the connection. The `AT+<XXX>?` command queries the current value in each case and resets them. The commands in lines 11 (pin) and 14 (transfer rate) are especially important. Even though Bluetooth does not have a long range, the signal will often reach the neighboring apartment, which is why changing the default PIN is recommended. I chose the transfer rate in line 14 to match the default for the Pi Pico. If you use a different microcontroller, adjust the value accordingly.

up on the Raspberry Pi, unless it is not available, then the HC-05 will abort the connection. Even if the Raspberry Pi is online later, the connection is not established because the HC-05 does not retry. For a robust process, the Pi Pico needs to switch on the HC-05 before each connection attempt and then switch it off again. If the microcontroller reads and sends data from a sensor at fixed intervals, this approach also saves power.

Collecting data from multiple Picos with the Raspberry Pi requires a bit more effort. The RFCOMM server [4] could be useful as a blueprint. Note that the Raspberry Pi only maintains one connection at any given time. Theoretically, more connections would be possible with Bluetooth and the SBC, but both the HC-05 and the Android apps always contact the device on default SPP channel 1. Of course, this would be fine for sensors that only occasionally send short signals, as long as the Pi Pico quickly releases the connection again after the data has been delivered.

Conclusions

From the third generation onward, the Raspberry Pi comes with Bluetooth built in. Therefore, you can control the SBC from an external device and control any microcontroller with the Raspberry Pi. The HC-05 does a good job, because the chip removes the need for you to plumb the depths of the Bluetooth protocol. Apart from a one-off configuration, you're left with just a little wiring. Older Raspberry Pis can be upgraded for a small charge by plugging in a USB Bluetooth dongle.

Classic Bluetooth, though, has disadvantages. The power consumption in particular makes the technology uninteresting for IoT devices. Bluetooth Low Energy (BLE) steps in with a protocol variant optimized for minimal power consumption. How you get the various devices in your home IT zoo to work with it is the topic for a future article. ■■■

Info

[1] "Setting Up Bluetooth on the Raspberry Pi 3" by Bernhard Bablok, *Raspberry Pi Geek*, issue 18, 2016, *https://www.raspberry-pi-geek.com/Archive/2016/18/Setting-up-Bluetooth-on-the-Raspberry-Pi-3/*

[2] Serial Bluetooth Terminal: *https://play.google.com/store/search?q=serial%20bluetooth%20terminal&c=apps*

[3] Bluetooth Control Panel: *https://play.google.com/store/apps/details?id=com.martyncurrey.Bluetooth_Control_Panel*

[4] PyBluez: *https://github.com/pybluez*

Author

Bernhard Bablok works at Allianz Technology SE as an SAP HR developer. When he's not listening to music or out and about, he's busy with topics related to Linux, programming, and small-board computers. You can contact him at *mail@bablokb.de*.

MakerSpace

Using clean code principles for better code

Good Housekeeping

Clean code principles can improve the readability of your source code, making life easier for both you and your users.

By Martin Mohr

Clean code is a set of rules and procedures that make it easier to read and understand source code regardless of the programming language used. Many clean code strategies fit all languages, but some more far-reaching strategies only prove useful in the object-oriented world. The clean code concept is not new or revolutionary. Back in 2008, Robert C. Martin described the procedures in his book, *Clean Code: A Handbook of Agile Software Craftsmanship* [1].

With today's highly complex IT applications, ensuring that source code has a clean structure has becoming increasingly important. The clean code concept has evolved in recent years, and the procedures described in Martin's book have been expanded. To get you started writing cleaner code, I'll describe some of the core ideas and rules of clean code.

The Scout's Rule

"Always leave code cleaner than you found it." If you take this rule to heart, a project's source code will get better and better over time. You don't need to be afraid of breaking something, because the previous version can always be restored thanks to the version management (i.e., Git). Modern IDEs also provide many tools for reengineering code. Renaming classes or methods doesn't pose so much of a risk anymore.

Typically, the compiler doesn't care what names you assign the individual program components. Whether you use uppercase as opposed to lowercase or underscore instead of CamelCase, as long as the name fits together somehow, the compiler will build an executable program.

However, this freestyle kind of code can be extremely difficult to read. Instead, you should to stick to the style conventions that apply to the programming language being used. In Java, for example, class names should always start with an uppercase letter, while methods should start with a lowercase letter. Modern IDEs usually have a feature to automatically format source code: You should definitely use this. If several people are working on a project, they need to agree on using the same format.

Mnemonic Names

A program's components should always be given mnemonic names. This rule applies to classes, methods or functions, and variables. The names should not be too long, but they still should describe precisely what is happening.

You also need to keep in mind the variables' validity range. It is perfectly fine to follow conventions when naming the loop variables, as in, `i` or `j`. And `x`, `y`, and `z` are certainly the most meaningful

Lead Image © donatas1205, 123RF.com

labels for a coordinate system. But if a variable or object is used in a larger context, you should definitely give some thought to the name.

Avoid including the class name in the method names. All programmers should understand that a method always works on the object in which the method is defined. While you should always use nouns as class names because they represent entities, method names should include verbs. Ideally, avoid using abbreviations in the names; there are some exceptions to this (e.g., VAT).

While it may sound peculiar at first, make sure the names you choose are pronounceable. People will need to talk about a class in meetings or code reviews. Not having overly complex or similar sounding names will make talking about a class easier.

As a universal rule, a comment should never be necessary to explain a component's purpose. Instead, the name should speak for itself. In other words, do not use meaningless generic names such as manager, helper, handler, processor, and so on. A class is always intended for a specific purpose, so name the class for that purpose. For customer projects, make sure you use the customer's designations for the names; otherwise, misunderstandings will repeatedly occur.

Choose names to show the idea, responsibility, and limits. For instance, if you use the Model-View-Controller architectural pattern (which subdivides software into the data model, view, and program control components), then name the classes `Model`, `View`, and `Controller`. Additionally, include the concrete function in the name (e.g., `LoginController` or `ContractSaveController`)

Type names do not belong in method and variable names. Instead of writing

```
List customerList = getCustomerList
```

use

```
List customers = getCustomers
```

You can already tell from the type that it is a list. In addition, you should always use the same name for an entity (such as `Login`, `User`, or `Account`). Filler words such as `to`, `from`, and the like should be avoided.

Comments

"If the compiler understands the code, so must the developer." This rule means that comments are unnecessary. This statement, which seems somewhat short-sighted at first glance, makes perfect sense in a slightly modified form in the clean code context. If you feel the need to add a comment to a program section, you should think again about whether the code is really clean. The same applies to the names of classes, methods, and variables.

I'm not saying you can't include comments in your code anymore. Instead, comments should prompt programmers to consider rewriting a code passage in a more easily understandable way. A comment needs to describe what the developer was thinking or trying to accomplish when writing it.

On the other hand, to-do comments are allowed, assuming that somebody actually does the work. The developer can use a to-do comment while still working on the code. Strictly speaking, however, a to-do comment should never make it into the repository. If it does, it must include a specific target date for completion.

Tools such as Javadoc can be extremely useful in many cases. You can automatically generate the necessary documentation, more or less as a side effect of using the tool. However, this automatically generated documentation is only useful as long as the comments in the source code are up to date. Pay special attention to this: Almost all developers rely on up-to-date documentation.

Variables

Always define variables just before using them in the program code. This approach offers several advantages:

- The program code and the matching variable definition fit on one screen page and can be read together.
- The variables' scope is clearly defined. Writing all of the variables at the beginning of a program is by far one of the worst ideas, because it would make all the variables global.
- Variables and objects take up memory and should therefore only be defined when they are actually needed (something that many developers no longer consider).

In the end, unclean code needs more system resources. This results in users needing more hardware, which in turn requires more power and pollutes the environment. Saving physical resources is obviously easier than saving virtual ones.

Methods

When it comes to using methods or functions, you can do a number of things to improve your code. Always choose meaningful names for the transfer parameters and the return values. If the IDE generates these names automatically, make sure you rename them to reflect the actual function.

Methods should only do one task at a time, but they should do it completely. Try to get into the habit of always using the same order for the methods in a class (e.g., the constructor first, then the public methods, and then the private methods).

Methods and functions will ideally contain no more than seven statements. If a method becomes longer, think seriously about breaking it down into several smaller ones. It is also useful to remove complex logical expressions from the `if` command and move them to a `private` method.

Methods operating on private variables should either set a value or return one (`get` or `set`), but not both. Do not use flags to change the way methods work. Instead, write two methods rather than including special versions of a method.

Methods should not have too many parameters. If you end up with four or more parameters, you need to consider whether it would make more sense to split up the method. Another possible solution is the use of parameter objects. Methods should not modify the transferred parameters or objects if possible. It is cleaner to create new objects and return them.

More Tips

You can employ side effects when programming code so that one command performs several tasks at the same time. This practical approach makes the code very short and compact. However, this approach can pose risks, because not every developer can easily recognize these side effects in the code. Try to write simple and understandable programs rather than “clever” ones.

Another option, the To approach, is a top-down process where you break down a task into increasingly smaller sub-steps. In the end, the individual steps consist of simple, short methods. You can also start by first writing comments to describe what you want to happen, and then implement the methods themselves. In the To approach, a method’s results are passed to the next method.

A class corresponds to an entity and must describe the entity completely. Try to create as few classes as possible. When doing this, keep classes as small as possible, and handle only one task per class where practical. Coupling of classes needs to be as loose as possible. This makes it much easier to replace a single class.

All good programmers are proud when they create a working loop for the first time. Unfortunately, this leads to the habit of using Magic Numbers, which are numbers that appear in the code without any comments and control some loop or specify a limit value in a condition. These numbers have no real meaning for the reader, but the code does what it is supposed to do in a magical way. You should replace these numbers with constants using mnemonic names, which will result in more readable code.

When you start developing the source code, don’t optimize for speed first. Instead, work on making the code readable. If it turns out that program execution is too slow, you can always optimize the offending code passages.

And always be aware that you may be prone to forgetfulness yourself. The rules and tips mentioned here are intended to generally build source code in a way that makes it easier to understand. It’s not only others that will benefit from this, but almost certainly your future self as well. Nothing is more annoying than not understanding your own code after a few months.

Being Professional

I only covered a few of the aspects that are important for creating clean source code. If you’d like to learn more, check out Martin’s book, *Clean Code* [1], for invaluable information.

Even though software is often created with the help of tools, these tools have only a minor influence on the product’s quality. The developer, on the other hand, bears the larger responsibility of delivering quality software. For that reason, it is extremely important for the developer to work efficiently and cleanly. Another book by Martin, *Clean Coder*, provides great insights into the developer’s daily grind [2]. The book describes the typical problems in software projects and shows ways to solve them. You not only need to know how to create clean code, but also how to become a developer with a clean work approach.

Conclusions

Building programs using the clean code approach is more of a marathon than a sprint. You can’t implement all these tips and rules overnight and write only clean programs. However, to paraphrase the scout rule, it is possible to program a little better every day. The advantages are obvious: Easily understandable code is easier to use and maintain. On top of that, it’s also much more fun to work with clean code. Even when pottering with Raspberry Pi projects, clean code can make your life easier, so start today! ■■■

Info

[1] Martin, Robert C. *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson, 2008: *https://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882/ref=sr_1_1?crid=3FXG99Y3WB2HV&keywords=clean+code+by+robert+c.+martin&qid=1663013391&sprefix=clean+code+%2Caps%2C88&sr=8-1*

[2] Martin, Robert C. *The Clean Coder: A Code of Conduct for Professional Programmers*. Pearson, 2011: *https://www.amazon.com/s?k=clean+coder&crid=3VZI0AMPCQBE3&sprefix=clean+coder%2Caps%2C113&ref=nb_sb_noss_2*

Author

Martin Mohr has experienced the complete development of modern computer technology in real time. After completing his studies, he mainly has developed Java applications. The Raspberry Pi has rekindled his old love of electronics.

If you didn't like to tinker, you wouldn't be using Linux, right? Well … mostly. Linux has gotten more mainstream over the years, so you don't have to be a hacker to enjoy it, but that do-it-yourself spirit lives on. Proprietary hardware vendors don't always mix well with do-it-yourselfers, but a little ingenuity can always open doors. This month we show you how to access and share iTunes metadata using XML and a little bit of Python. Also in this issue, sync your files with Seafile and explore the Ryzom multiplayer role-playing game.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE

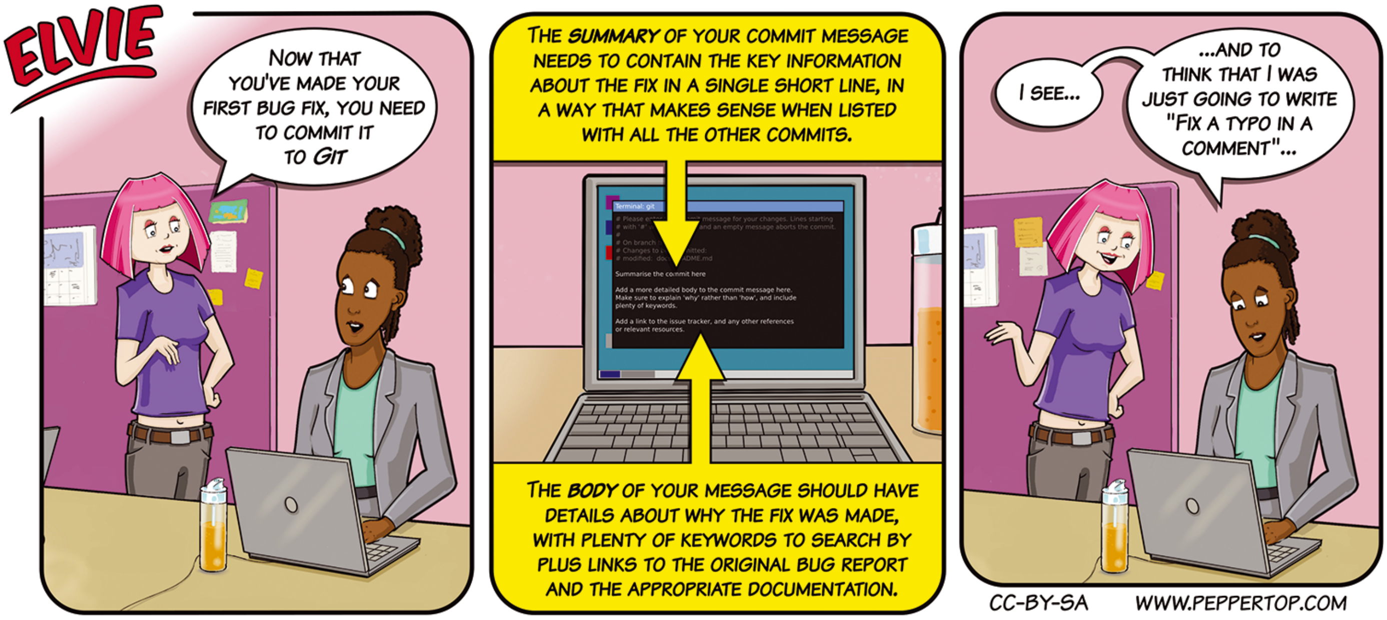
Doghouse – Portability and Costs 72
Jon "maddog" Hall
Choosing the right server computing solution for your company, and minimizing CPU overhead from the start, can help lower costs and maximize portability over time.

Ryzom 74
Daniel Tibi
Fancy an adventure? Travel to the futuristic planet Atys, where many exciting missions await.

Seafile 9 80
Ferdinand Thommes
This open source file storage tool is more than a match for Nextcloud and ownCloud.

FOSSPicks 86
Graham Morrison
This month Graham looks at Furnace, Node-RED 3, Exquisite, Corrscope, darktable, Overgrowth, and more!

Tutorial – iTunes XML 92
John Cofield
Read and manipulate data from your iTunes XML to share music metadata between applications and across devices.


ELVIE
NOW THAT YOU'VE MADE YOUR FIRST BUG FIX, YOU NEED TO COMMIT IT TO GIT
THE SUMMARY OF YOUR COMMIT MESSAGE NEEDS TO CONTAIN THE KEY INFORMATION ABOUT THE FIX IN A SINGLE SHORT LINE, IN A WAY THAT MAKES SENSE WHEN LISTED WITH ALL THE OTHER COMMITS.
THE BODY OF YOUR MESSAGE SHOULD HAVE DETAILS ABOUT WHY THE FIX WAS MADE, WITH PLENTY OF KEYWORDS TO SEARCH BY PLUS LINKS TO THE ORIGINAL BUG REPORT AND THE APPROPRIATE DOCUMENTATION.
I SEE...
...AND TO THINK THAT I WAS JUST GOING TO WRITE "FIX A TYPO IN A COMMENT"...

CC-BY-SA WWW.PEPPERTOP.COM

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®.

MADDOG’S DOGHOUSE

Minimizing CPU overhead from the beginning can help you lower costs and maximize portability over time.

BY JON “MADDOG” HALL

Stay nimble

A number of years ago cloud computing came on the scene, with one of the first suppliers being Amazon. Amazon’s use of their data farms peaked during the Christmas season, which left them with the capacity the rest of the year to sell Internet-accessible server computing to companies who needed it, at a fraction of what it would cost for a small company to supply it themselves. An industry was born.

Almost overnight (at least by most industry terms) companies started to turn over their server computing to other companies (AWS, Google, Microsoft, and others) who had the computers, staff, physical plant, security, and so on necessary to do the work.

I have advocated for the use of many of these cloud services when a fledgling company is starting out. In the open source space you could think of places like SourceForge, GitHub, GitLab, and others as “cloud services” that make the development and collaboration of software (and even hardware) easier and less expensive.

The problem comes with two issues: lock-in and growth. Both of these have been with the computer industry for decades.

Lock-in can happen when the developer uses interfaces or platform features that are nonstandard. Even in the days of simpler programs there were standards that allowed programs to be moved from computer to computer if the programmer only coded using the formal standard. In almost any commercial compiler, there were “extensions” to the standard that offered the programmer easier methods of coding or more efficient execution of the code. These extensions were usually documented in a gray (or some other color) documentation with the warning that this was an extension, allowing the programmer to stay away from that extension if they wanted portable code.

A good programmer might then code what is called a “fallback,” which would execute using only the standard code, but if operating in an environment where the extension existed then the extension could be used, either with a recompile or a runtime determination.

I have used a simple example, but these “extensions” to standards occur at every level, from programming languages to library and system calls, to the interfaces of your cloud systems, which cloud service providers sometimes offer as their advantage over their competitors.

All of this might be fine if your cloud service is guaranteed to always be the least expensive, most stable, give you all the services you need as your company grows, and so on. I have, however, worked for some of the largest companies on earth, which I thought would be there “forever,” and which are now completely gone. You have to be ready to move and sometimes comparatively quickly.

The second reason for being nimble with regard to portability is the growing expense of cloud computing versus the costs of running your own server systems, especially in certain environments. These costs can be both economical and political. The more your company grows in size and scale, the more you should plan for the contingency of having to move or separate your server loads.

This article was inspired by a whitepaper recently published showing how the cloud service costs of some users rapidly increased over time due to the growth of the data or computer load to the point where it might have been practical for some large companies to create their own data centers again, after moving to cloud services several years ago. Unfortunately, moving even to another cloud service, much less your own physical plant, is often complex and expensive.

Besides maintaining flexibility through portability, working with tools to modify the amount of CPU, data storage, data transfer, and Internet usage can reduce (sometimes dramatically) these charges from whatever cloud supplier you use.

Recently a programmer that I work with went through some older code on their project and found an application that did some dynamic allocation of memory needed for each partial transaction. For reasons too complex to explain here, this caused an overhead of 1,200 milliseconds (yes, if you do the math this means 1.2 seconds) for each transaction … painfully slow for the user but also putting an unnecessary strain on the server. The programmer changed the algorithm to calculate how many allocations would have to be done and then just allocated the space one time, and (in effect) the CPU overhead dropped to zero. This savings is one thing if there is only one user of this program, or if it is running on a laptop, but if there are hundreds of users utilizing it in a server environment, the CPU utilization mounts up quickly.

In summary, what is often inexpensive in small quantities can rapidly become a big expense in larger quantities, so plan to try and minimize the overhead from the very beginning and remember that “performance” is not just how fast your application runs, but how portable your code is and how many programming resources it can save if you have to move your code. ■■■

Public Money
Public Code

Modernising Public Infrastructure with Free Software

Free Software Foundation Europe

Learn More: https://publiccode.eu/

An exciting mix of fantasy and science fiction with Ryzom MMORPG

Spirit of Adventure

Fancy an adventure? Then travel to the futuristic planet Atys, where a large world with many exciting missions awaits.

BY DANIEL TIBI

Ryzom is a free massively multiplayer online role-playing game (or MMORPG – an online role-playing game in which a large number of players interact simultaneously over the Internet in a virtual world) with a complex story and many episodes. In the game, your adventure starts centuries in the future on the planet Atys.

The planet is home to various peoples. The Kami are nature-loving shape shifters and skilled mages. The Karavan, on the other hand, are a highly technical people with powerful weapons.

Both the Kami and Karavan help the homins, whose role the player assumes. The homins are divided into four nations. The Fyros are a warlike nation with their homes in the desert, and the craft-oriented Matis live in the forest. The Tryker, explorers and discoverers, live by the sea. Finally, the Zoraï are mages who inhabit the jungle.

Finally, the underground primordial root system of Atys is home to the insect-like Kitin, who advanced to the surface in 2481 and began a war against the homins, who suffered a crushing defeat. The few homin survivors have built new cities in remote areas of the planet.

Off to Atys

Your journey to Atys starts when you install the Ryzom [1] MMORPG. You will find a download link directly on the home page of Ryzom's website. Download the installation file, unzip it, and run `Install Ryztart`. A wizard will guide you through the installation. If successful, you will find an icon that you can then use to start the game. See Ryzom's user manual for further help [2].

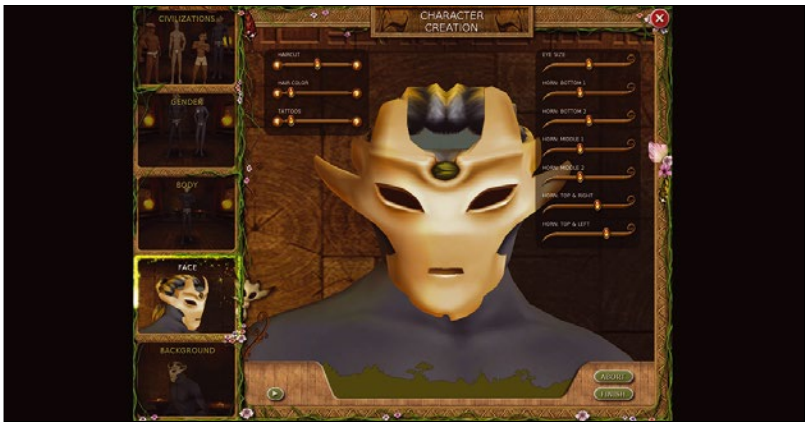


Figure 1: Start by creating your character and choosing one of the four nations.

When you play Ryzom for the first time, you must create a new user account. Ryzom is fully playable with a free account. With a paid premium account, you'll support the game's development and get various rewards, such as higher levels you can reach or more experience points. The source code for the underlying framework – and thus the basis of the game – is provided by the developers in the form of Ryzom Core under open source licenses [3].

Creating a New Character

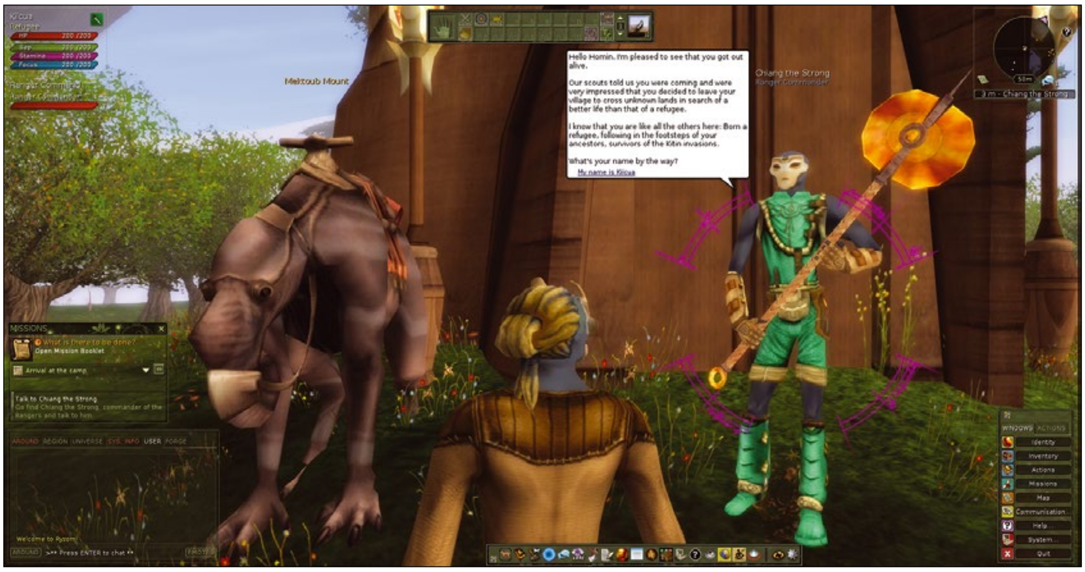
Before you get started, you must first create a new character (Figure 1). To do this, first decide on one of the four nations. While each of them is known for specific skills, you do not have to stick with this in Ryzom. Each member of each nation can fully learn magic, combat, resource hunting and mining, and crafting. Once you have chosen a nation, you design the gender, physique, and face of your character.

Next you specify which keys you want to use to control the character. It is just a matter of which hand you want to use to control the character's movements. If you are right-handed, the best way to move your hero is with the *W* (forward), *A* (left), *S* (back), and *D* (right) keys. This leaves your right hand free for the mouse, which you use to initiate other actions with the character. If you are left-handed, you can use the arrow keys to control the movements.

Last but not least, choose a name for your character. You either get creative yourself or let the computer generate a random name.

Welcome to Silan!

Your journey on the planet Atys will start on the island of Silan, where you'll receive your training


Hello Homin. I'm pleased to see that you got out alive.

Our scouts told us you were coming and were very impressed that you decided to leave your village to cross unknown lands in search of a better life than that of a refugee.

I know that you are like all the others here: Born a refugee, following in the footsteps of your ancestors, survivors of the Kitin invasions.

What's your name by the way?

My name is Kilcua

Figure 2: Chiang the Strong welcomes you to the Ranger camp on Silan Island. He will explain what you need to do next.

before heading to the main land for the actual missions. You will meet many characters on Silan. The game displays the names of other people above their heads. If the name is in white, this is a computer-controlled non-player character. A name in blue, on the other hand, stands for another human player. You can right-click on the character to chat with them or form a team.

Your character is a refugee and a descendant of the survivors of the war against the Kitin. The Ranger faction wants all homins to unite in the fight against the Kitin. The Ranger camp is located in the south of the island, where you will first meet Chiang the Strong, the leader of the Ranger camp on Silan. Right-click on Chiang to talk to him and inform him of your arrival (Figure 2).

After greeting you, Chiang the Strong points you to Unnoirin Cepao, the head of the camp. At top right, you will see a compass. Its needle either points north, or – if you are on a mission – to the destination of your mission. Press *M* to display a map of the island with a white arrow marking your position. Your mission target is marked by a flag. Right-click on the map to add markers yourself to find specific locations later.

To find Unnoirin Cepao, simply follow the compass. It is not far, but the camp leader is quite sleepy. You will probably have to wake her up. Once awake, she tells you more about the races and nations on Atys and refers you to the instructors, who will teach you skills in magic, combat, resource hunting and mining, and crafting.

Each of the four nations has its own space in the Ranger camp. To the east, you will find the Zoraï. With the Zoraï, you will learn the art of magic from Nomis Merclao. In the north, you will encounter the Fyros, who provide combat training from Guilan Guiter. To the west, you will find the Trykers, where Milles Dodoine will introduce you to the art of finding and mining raw materials. Finally, the Matis are encamped in the south. This nation trains you as a craftsman. Which skills you acquire and how far you develop them are up to you. The more you learn, the better – but you won't get far without combat training.

The basic principle on which your training works in all areas is that you first go to the respective instructor, who sends you on a mission. Once you have completed the mission, you will receive experience points for it. You exchange these points with the teacher, who you will always find right

Figure 3: Learn the art of magic to attack your opponents or heal yourself and other players.

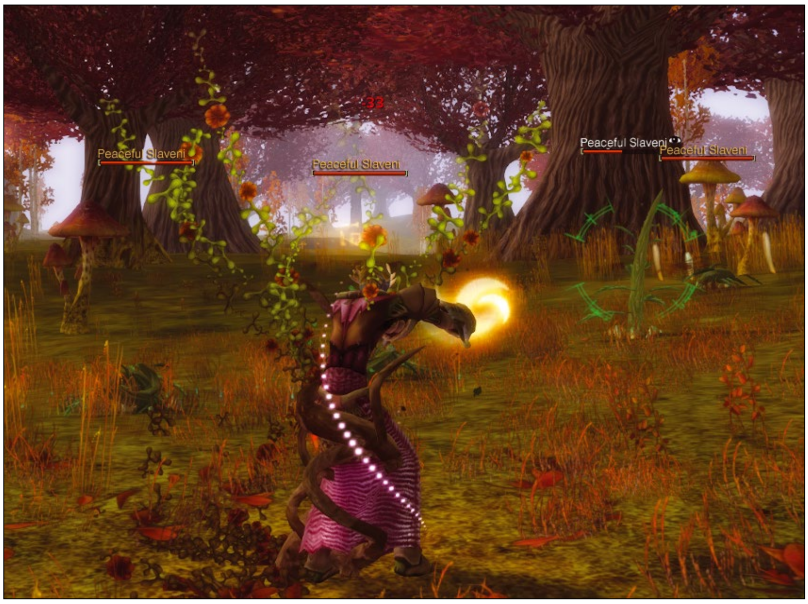


Figure 4: Plants are best attacked with the decay spell.

next to the respective instructor, for new skills, which in turn form the basis for further missions.

The Art of Magic

The homins learned the art of magic from the Kami (Figure 3), and this art is especially cultivated by the Zoraï. Nomis Merclao first teaches you offensive magic, which you'll use to attack your opponents. Your first target is the baby Yubos in the north of the camp. Click one of the little animals with the left mouse button to select it as a target. At the top of the screen, you will find an action bar with small icons for the individual actions you can perform. Choose *acid damage*.

The homins derive their magical ability from the sap, a liquid from the interior of Atys. In the upper left corner you will see four bars under your character's name. The green one indicates your sap supply. When you cast spells, your sap supply decreases, but it automatically replenishes over time. You start with 200 points of sap. Over time, you'll exchange your experience points for magic with the magic teacher to boost your sap points. This means that you have more sap available, and your sap supply regenerates faster.

Figure 5: Most animals on Silan (like the Mektoub shown here) are peaceful and do not attack on their own. Few attack you without you having attacked them first.

Once you have completed a mission, your magic experience points will increase. Return to Nomis Merclao to report your success. He will send you on more missions where you learn more about magic.

At the top of the screen, you will see icons for the right hand and for the left hand. You probably still have your dagger in your right hand. Click on the right hand icon and select the magic enhancers to put on your hands. Your character now puts down the dagger and puts the magic enhancers on both hands instead. This makes your magic even more effective.

You will also learn how to compose and modify spells yourself. In addition to the acid attack, study the rooting spell to root your opponents to the ground for a short time, the decay spell to attack plants (Figure 4), and the fear spell to terrify your opponents into flight.

Besides offensive magic, there is also defensive magic. At top left, you will see a red line showing your life points. If you are attacked, these points are reduced. If the line reaches zero, your character dies.

Fortunately, the Kami are skilled in the art of resurrection, so in such a case your character will wake up in the Kami enclave in the northeast of the camp. However, your development will incur a penalty point, which you need to work off by earning experience points (see the "Redeeming Penalties" box).

Defensive magic helps you regenerate your own health and that of other players faster, among other things. For example, you can support a team member fighting a powerful opponent. Nomis Merclao also introduces you to team building and your role in serving others. All missions on Silan can definitely be solved alone, so team building, which you are instructed to do in individual missions, is always optional. However, many tasks are more easily completed in a group.

Combat Training

Combat is the domain of the Fyros, with Guilan Guiter as your trainer. In the beginning, you'll

Redeeming Penalties

If your character dies, they will be revived but will be subject to a penalty. The more trained your character is, the greater the penalty will be. You can reduce the penalty by earning experience points. The least dangerous way to do this is to mine raw materials.

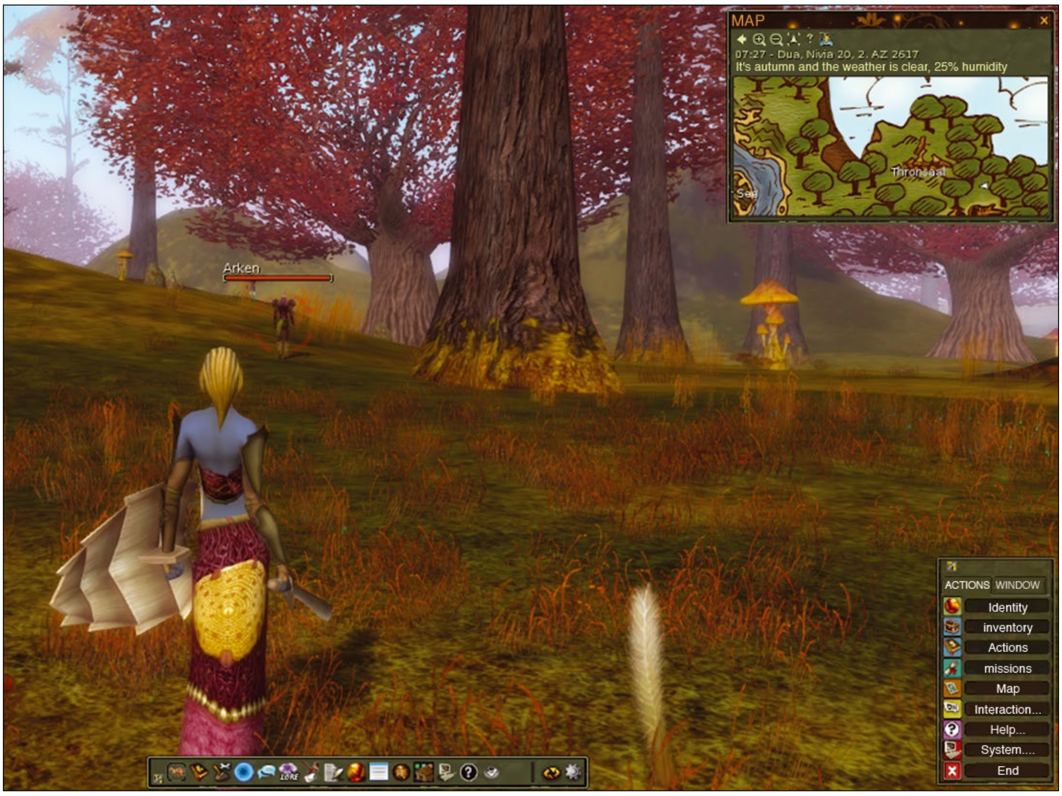


Figure 6: The final mission of your combat training is to put the bandit leader Arken out of action. He hides out on a hill between the ruins and the throne room.

only have a dagger, but later on you'll be given other weapons and a shield as a reward for completing missions. Click the right hand icon in the top bar to pick up a weapon. There are one-handed and two-handed weapons. If you choose a one-handed weapon, your left hand remains free for a dagger, which increases your attack power, or for a shield, which boosts your health points and your defense.

Start by training your fighting skills during a hunt (Figure 5). Left-click on your target and choose one of the combat actions from the bar at the top of the screen. Apart from standard attacks, you can choose a precision attack, which reduces the opponent's chance to dodge, as well as an attack type with greater damage, which helps you finish off your opponent faster. Once you have chosen a target, you will see its life bar in the upper left corner. If the life bar drops to zero, your attack was successful. What you will also see there is the skill level that your opponent has in the fight. Be careful and do not attack anyone whose level is too high for your own abilities.

Fighting is taxing on your hero's stamina, the level of which you can see from the purple line top left (see the box "Faster Recovery"). Over time, you'll be able to trade experience points with your combat instructor for increased stamina and faster stamina recovery.

The last mission Guilan Guiter sends you on is the fight against the bandit leader Arken (Figure 6). He hides out on a hill between the ruins and the throne room, where he is protected by his gang. Your best chances against Arken are in a team with other players. If you are facing him alone, it is best to wait until he is standing on his own at the edge of the hill. Lure him away from the others by attacking him with magic. He will pursue you while his protectors stay behind on the hill. Once you have mastered this last mission, you are given a flaming two-handed sword as a reward.

Faster Recovery

Your character regenerates faster when sitting – press the Insert key to do this. However, you cannot perform any other actions while sitting and are more vulnerable in this position. Always retreat to a safe corner to recover.

Raw Materials and Crafts

The Tryker are masters at finding and mining raw materials. Milles Dodoine will introduce you to this art. One mission takes you to a resource deposit in a region teeming with Javings. These birds of

Figure 7: Silan is rich in raw materials.

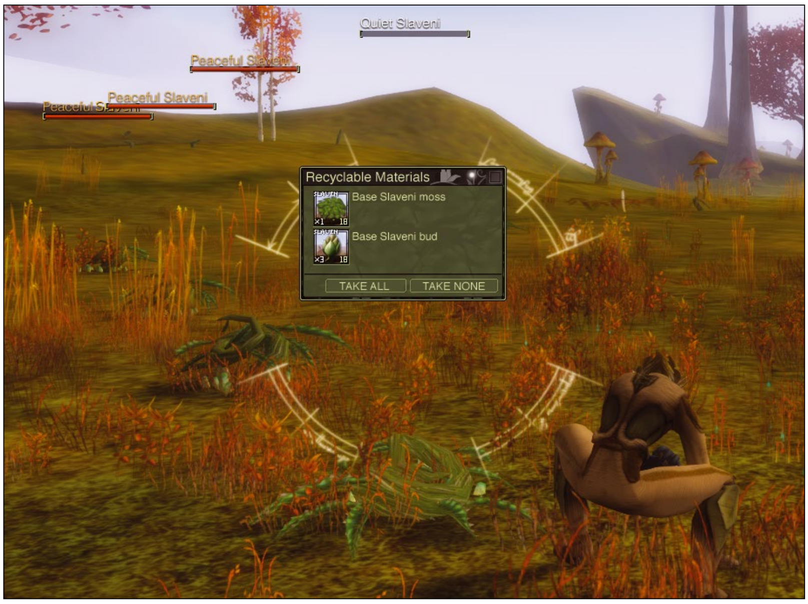


Figure 8: You can collect materials for your crafts from raw material sources by hunting animals and gathering plants. Alternatively, you can buy whatever you need from the traders in the Tryker camp.

prey are some of the few animals on Silan that attack you without provocation. However, if you have completed your magic training, you will know how to face these pesky birds.

The crab-like Cray on the island in Sparkling Lake, where another mission takes you, may seem scary, but they won't hurt you as long as you don't attack them. You can exchange the raw materials you collected for Dappers, the currency on Atys, with the raw material traders in the Tryker camp (Figure 7). You can then spend the money you earned on whatever you need from other merchants. You can also keep the raw materials and refine them.

The most skilled craftspersons on Atys are the Matis. Sterga Hamla will instruct you in this art, but you will have to flatter him terribly in the dialogs before he decides to pass on his skills to you. You can either collect the materials for your crafts yourself or buy them from the raw materials trader in the Tryker warehouse (Figure 8). Once you have sufficient crafting skills, you can create your own armor to boost your health points and give you better protection against attacks. To find out which materials you need for which craft product, click the book icon with the description *Craft Recipe Book* in the bar at the bottom of the screen.

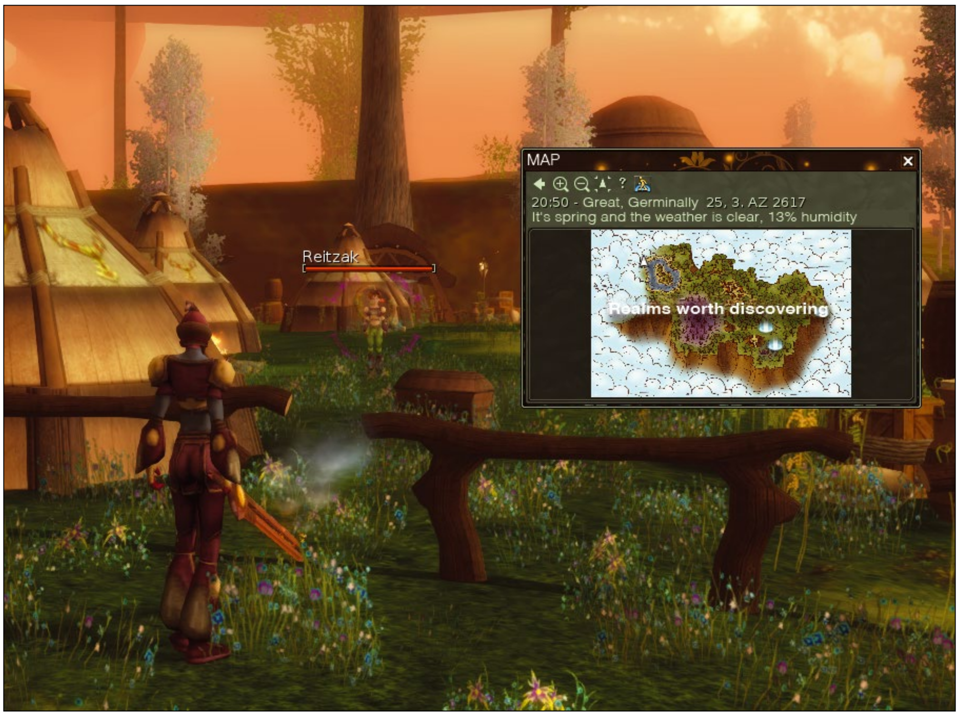
Ready for Action

It is up to you which, and how many, training missions you complete. Once you have trained enough, you can venture out on your first real missions on Silan. You can get these from the counselor at the Kami Enclave, from the communications officer at the Karavan Embassy, and from Chiang the Strong.

The missions of the Kami and the Karavan take you to the Blight Zone, an area contaminated with the little-researched substance Goo. It seeps up from the interior of the planet, destroys the land and makes the animals go crazy. The Kami give you the task of rendering some of these contaminated animals harmless. As a reward, you will receive plenty of Dapper, and your reputation with the Kami will grow. The Karavan sends you out to collect various Goo samples, which you either find by searching for resources around the Blight Zone or take from animals you have killed around the Blight Zone. Again, you will receive plenty of Dappers, and your reputation with the Karavan will increase.

It makes sense to embark on the missions that Chiang the Strong has in store for you at the end of your training because they mark the transition from the end of your time in Silan to your departure to the mainland. The first mission Chiang gives you is: "The bandits have a secret. Discover it." In detail, this very broadly worded mission means that you need to search for and take out a bandit with special skills in the area between the Blight Zone and the Sparkling Lake near the Rendor herd. You capture a Goo extractor from them and deliver it to Chiang.

Your next mission is to show the device to the four camp leaders of



20:50 - Great, Germinally 25, 3. AZ 2617
It's spring and the weather is clear, 13% humidity

Figure 9: One of the missions Chiang the Strong gives you means that you have to face Reitzak, the leader of the Chlorogoo bandits, whose camp is located north of Sparkling Lake.

the four nations, which you do in a quick tour of the camp. From now on, you can use the Karavan teleporter, which will save you a lot of running around, because it will take you from the Ranger camp directly to the Karavan outpost in the north near Sparkling Lake.

The next mission takes you to the Kami outpost at the lake southeast of Sparkling Lake. After all the running around, you are again given a real task there: To take out Reitzak, the leader of the Chlorogoo bandits (Figure 9). The bandits' camp is located north of Sparkling Lake. It is a good idea to form a team to complete this mission because there are other bandits in the camp who will defend their leader in case of a direct attack. But if you are alone, wait until Reitzak steps to the edge of the camp and attack her with magic. She will run after you and leave the camp while her protectors stay behind. As a reward, Chiang will give you a generous helping of experience points.

The last mission Chiang the Strong sends you on sees you face the Kitin for the first time. You swim into the Kitin Jungle through a cave southwest of Sparkling Lake. When you get there, you collect sap from the abundant sources of raw materials and kill a number of the Kitin. Your final opponent here is a Kirosta (Figure 10). Again, the easiest way to defeat it is to work together as a team. But it can be done alone, if you have sufficient training. As a reward, Chiang will give you a full set of ranger armor. This means that you have completed all your missions on Silan and are ready to make your way to the mainland.

Off to the Mainland

Silan is primarily for training and developing your character. Once you have learned enough, talk to Chiang the Strong, who will show you the way to the mainland. He first sends you to the camp leaders of the four nations and then to the Kami enclave, from where you are teleported to one of the four capitals of the nations. You are free to choose a city.

Figure 10: The most powerful enemy on Silan is the Kirosta, found in the Kitin Jungle.

Make your selection wisely. The cities are interconnected, but the route from each city to the others is long and dangerous, so you can't easily change your choice. It is best to travel to your own nation's capital first. The traders and the other non-player characters will be more friendly to you (Figure 11). Theoretically, you can leave at any time, but you will want to complete all the missions on Silan up front in order to be best prepared for the adventures that await you.

It's really hard to believe that you can play such an elaborately designed MMORPG like Ryzom for free. The background story is imaginative, the missions are exciting across the board, and the developers have put a great deal of TLC into the graphics. Ryzom is a clear recommendation from us. But watch out: It's addictive. ■■■

Info

[1] Ryzom: *https://ryzom.com*

[2] User manual: *https://en.wiki.ryzom.com/wiki/User_Manual*

[3] Ryzom Core: *https://wiki.ryzom.dev*

Figure 11: Once you have learned enough on Silan, you are ready to leave for one of the four capitals of the four nations. Here, a newcomer to the Zoraï capital first talks to the city's welcomer, who explains what needs to be done next.

Fast cloud storage for in-house hosting

Sea Freight

Seafile offers file sharing and synchronization like Nextcloud and ownCloud, but its speed leaves the competition far behind. BY FERDINAND THOMMES

When it comes to file synchronization and collaborative work, the kings of the hill in the open source universe are ownCloud and Nextcloud. But because other vendors have attractive products too, let's take a look at Seafile [1], which bills itself on its website as an open source file synchronization and sharing solution designed for high reliability, performance, and productivity. Like with Nextcloud or ownCloud, users retain full control over their data. Seafile tends to hide its light under a bushel. This article tries to clarify why this is so, because, in terms of functionality, there is no need for Seafile to hide at all.

Differences

Let's start by clarifying the main differences between Seafile and the aforementioned competitors. Seafile's strengths lie in its file sharing and synchronization features. Nextcloud and ownCloud see themselves more as full-fledged groupware solutions with many apps and continually-expanding feature sets. One technical difference that only becomes apparent when you read the specifications for the first time is that while Nextcloud relies on PHP, Seafile is written in C and Python, with the Django framework working in the background.

The software is developed by Chinese vendor Seafile Ltd. and based on the client-server principle. The free Community Edition was developed in 2012 from the Seafile Professional Server Edition and reached version 9 in December 2021. Since then, there have been several minor updates to Seafile Server 9.0.5, which is the version on which our test is based.

If you have any concerns about the software coming from China, the Community Edition of Seafile is completely open source and is published under the GPLv2. You can view the code on GitHub and follow its development [2]. Until 2016, there was a German company named Seafile GmbH, which had to change its name after a dispute over rights with the Beijing-based Seafile Ltd., and it has since operated as Syncwerk GmbH.

Editions

On the server side, the Professional Edition only offers support for Linux, while the Community Edition also offers servers for Windows and Raspberry Pi OS. Clients of both editions are available for Linux, macOS, and Windows, as well as Android and iOS. In addition, there is a web interface named Seahub. The project's website offers a checklist comparison of the respective features of the two editions [3]. You can check out whether the Community Edition is okay for your use case or whether you need to splash out some cash on the Pro Edition. Among other things, the Community Edition lacks full-text searching via ElasticSearch and LDAP integration.

You can explore a demo of Seafile from the company's homepage [1]. If you limit Seafile to just three users, you can use the Pro version for free if needed. And if you want to try Seafile Pro Edition with more than three users, you can contact the company (see under "How to order" [3]) and receive the program and a trial license for three weeks.

Seafile's web interface is clear-cut and the learning curve is short. The button functions are usually self-explanatory, with additional tooltips popping up when you mouse over the buttons.

Seafile uses libraries to manage uploaded files or any files created in Seafile. Each library can be synchronized, encrypted client-side, and shared separately. In the libraries, you can create hierarchical directory structures with folders and subfolders as needed. Synchronization supports a location-independent work approach with clients always kept in sync on the remote devices.

Installation

If you decide to use Seafile, you first need to install it. On the download portal [4], you will find servers for Linux for both editions and for the Raspberry Pi in the Community section. The clients are broken down into Desktop Syncing Clients and Desktop Drive Clients, the Drive Clients giving users access to files on the server without synchronizing them with the local disk. For each of the two variants, clients are available for Linux (GUI and terminal), macOS, and Windows. Clients for mobile devices are available for Google's Android platform and Apple's iOS.

The Linux server offers three installation scenarios. Starting in version 8.0, the project recommends installing in a Docker container. The setup is automatically handled by a Docker compose file [5]. The second method, in the form of an installation script, sets up the server with MariaDB, Memcached, and NGINX as a reverse proxy in a few minutes, but the script is considered deprecated [6].

With the third method you do everything manually [7]. Although this offers the greatest knowledge gains, it takes the longest and entails the risk of getting many things wrong. If you prefer this way, you should go for MySQL/MariaDB rather than the SQLite option, because you will reach SQLite's limits sooner or later. Docker benefits: For one thing, the installation works with any distribution that supports Docker. On the other hand, the containers offer more flexibility because they can be copied faster and easily moved to another host.

With or Without Docker

I will be looking at the Docker variant on Ubuntu Server 20.04. Listing 1 shows the installation of Docker on this system. Then, using the commands in Listing 2, you install Docker Compose using `docker-compose --version` to check that the desired version is installed.

Whether you install Seafile on a Raspberry Pi, an old computer, or a server does not matter at first, provided you have downloaded the correct architecture for the hardware in question. ARM-based Docker files are available for the Rasp Pi [8]. You will want to install the server variant for Ubuntu or Debian. The system does not require a desktop environment, because you will be working with the web interface later on.

To install Seafile itself, download the `docker-compose.yml` [9] file off the web using the commands in Listing 3 and drop it into your home directory in the `/home/seafile/` folder that you create up front.

Then open the file in an editor and adjust the configuration to meet your needs. To do this, assign passwords for the `MYSQL_ROOT_PASSWORD`, `DB_ROOT_PASSWD`, and `SEAFILE_ADMIN_PASSWORD` lines. If necessary, adjust the two paths under `volumes` if you prefer a different location in the filesystem tree. Store an email address in `SEAFILE_ADMIN_EMAIL`. You will need this to log in later on.

If you want the server to be accessible and encrypted on the road, you also need to set `SEAFILE_SERVER_LETSENCRYPT` to `true`. Then enter a registered domain name for `SEAFILE_SERVER_HOSTNAME`. If you do not have a registered domain, you can use a service such as DynDNS. For an initial try-out, you will want to call the Seafile installation without SSL (i.e., without `https://` in the URL) to eliminate possible sources of error.

In the Browser

Then, working in the `/home/seafile/` directory, run:

```
sudo docker-compose up -d
```

This downloads the required images and dumps them into the appropriate containers. This will take a few minutes. Once the prompt reappears (Figure 1), log in to the web interface from any computer on your network. The URL is composed of the IP address of the server and the corresponding port.

Depending on the installation method, this can be `http://`*IP*:80, `http://`*IP*:8000, or, if you have integrated a certificate for SSL transmission, `https://`*IP*:443. If you do not have a static IP address for the server and your router allows this, you should always assign the same IPv4 address to the server.

Listing 1: Installing Docker (Ubuntu 20.04)

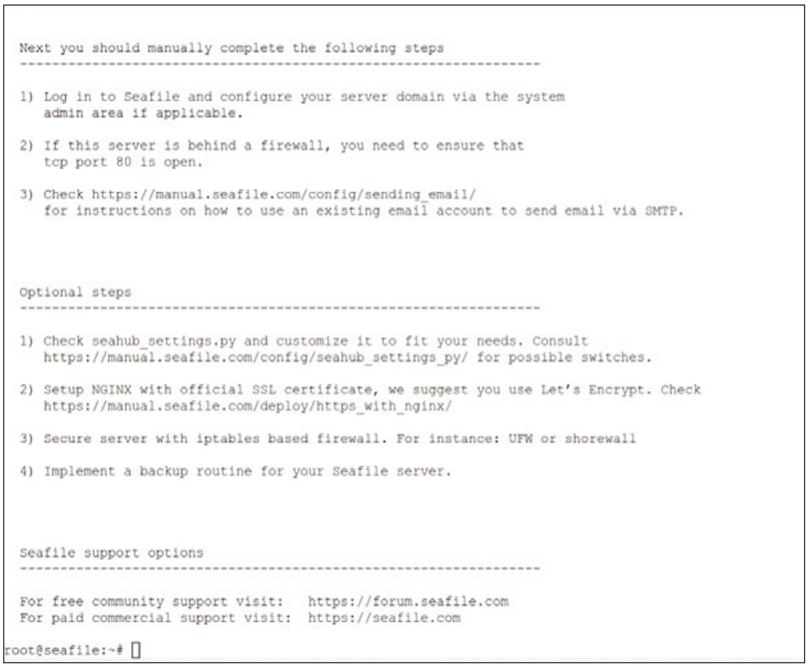
```
$ sudo apt update
$ sudo apt install apt-transport-https ca-certificates curl
  software-properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key
  add -
### For other releases replace "focal" with the respective code name
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/
  linux/ubuntu focal stable"
$ sudo apt update
$ sudo apt install docker-ce
$ sudo systemctl status docker
$ sudo usermod -aG docker ${USER}
```

Listing 2: Installing Docker Compose (Ubuntu 20.04)

```
### Replace version 2.6.0 with a newer release if applicable.
$ sudo curl -L "https://github.com/docker/compose/releases/download/
  v2.6.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/
  docker-compose
$ sudo chmod +x /usr/local/bin/docker-compose
$ docker-compose --version
```

Listing 3: Seafile via Docker Compose

```
$ sudo mkdir /home/seafile
$ cd /home/seafile
$ sudo wget -O docker-compose.yml https://download.seafile.com/
  d/320e8adf90fa43ad8fee/files/?p=/docker/docker-compose.yml
```



```
Next you should manually complete the following steps
----------------------------------------------------------------

1) Log in to Seafile and configure your server domain via the system
   admin area if applicable.

2) If this server is behind a firewall, you need to ensure that
   tcp port 80 is open.

3) Check https://manual.seafile.com/config/sending_email/
   for instructions on how to use an existing email account to send email via SMTP.


Optional steps
----------------------------------------------------------------

1) Check seahub_settings.py and customize it to fit your needs. Consult
   https://manual.seafile.com/config/seahub_settings_py/ for possible switches.

2) Setup NGINX with official SSL certificate, we suggest you use Let's Encrypt. Check
   https://manual.seafile.com/deploy/https_with_nginx/

3) Secure server with iptables based firewall. For instance: UFW or shorewall

4) Implement a backup routine for your Seafile server.


Seafile support options
----------------------------------------------------------------

For free community support visit:   https://forum.seafile.com
For paid commercial support visit:  https://seafile.com

root@seafile:~# 
```

Figure 1: After Docker Compose has set up the three containers, an instruction panel explains your next options.

Clearly Structured

Seafile's interface is clear-cut and well laid out; all the options are immediately understandable (Figure 2). From the left panel, you can access libraries, groups, connected devices, and share management (Figure 3). You can use the avatar icon top right to access your profile and system administration. This where you create additional users, groups, and libraries. You can then share them or assign them to another user.

To check if the network connection to the server is working, manually upload a file as a test. To do this, use the *Upload* button, which you will find on the left side of the top bar in every library. Alternatively, drag the files into the open library.

If a network error occurs during the upload (and this is probably more of a Docker issue than a Seafile one), go to the *Settings* subitem in System Administration and take a closer look at the first two lines, `SERVICE_URL` and `FILE_SERVER_ROOT`. Do they contain your IP address and, if applicable, the port? If not, add the IP that you used to launch the web interface here. You need to keep the `/seafhttp` bit.

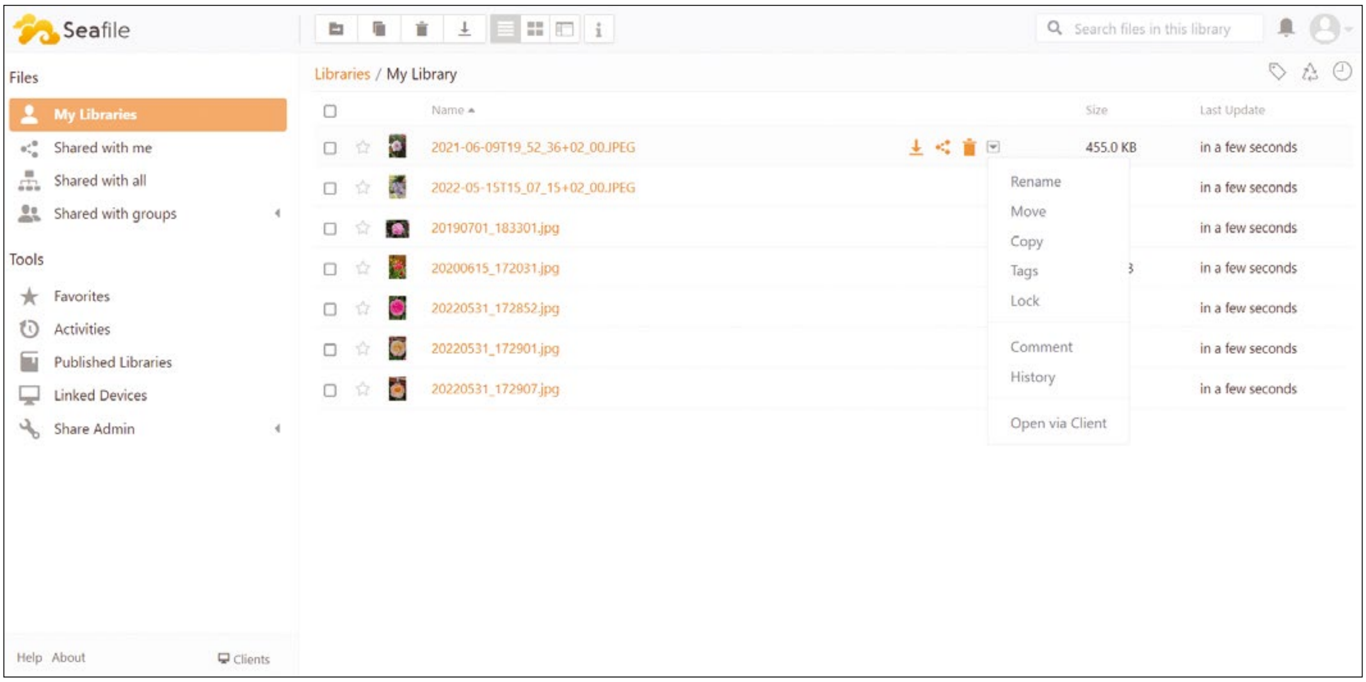
After changing the first line, check the box at the end. Do the same thing for line 2. After pressing F5 to refresh the page content, you should now be able to upload files.

Clients

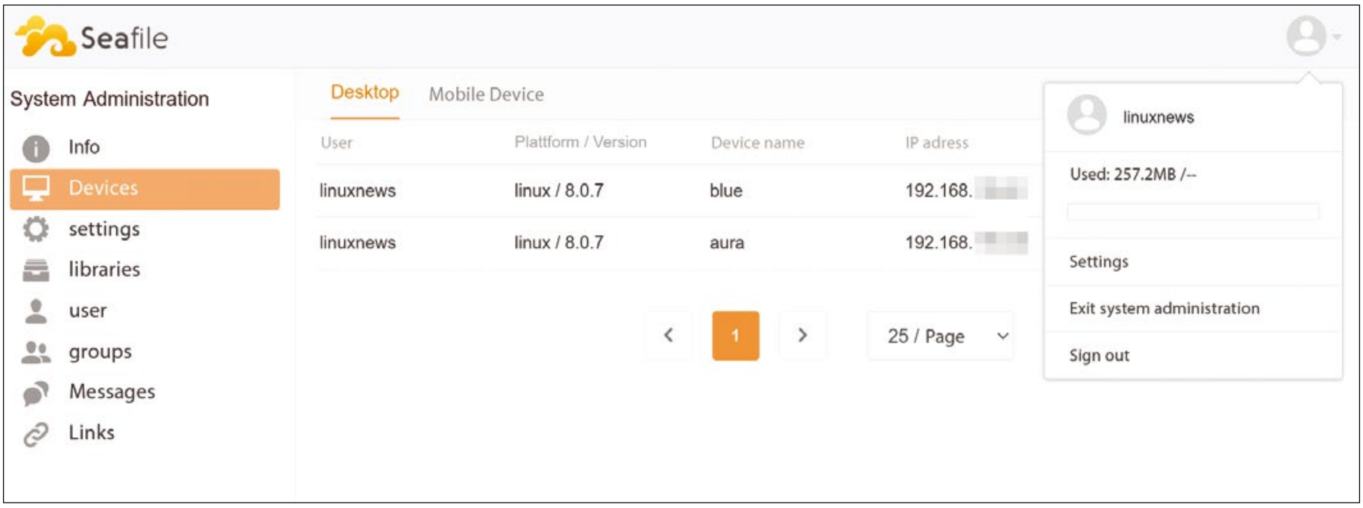
Seafile offers desktop clients for Linux, macOS, Windows, Android, and iOS. A client can connect to multiple Seafile servers. The Linux client can be found in the repositories of many distributions, but these are often older versions. In Ubuntu 20.04, for example, the version counter has only reached 8.0.7. To fetch an up-to-date version of the client, run the commands from Listing 4 on Ubuntu 20.04. For Ubuntu 22.04, change `focal` to `jammy` in the first command, and for Debian 11 change `focal` to `bullseye`.

Seafile can also be integrated directly into file managers such as Dolphin (KDE Plasma) or Nautilus (Gnome) via WebDAV instead of using a client [10]. Clicking on the IP in the client will take you to the web interface in the browser. The client can also be operated in the terminal emulator after installing the terminal client on servers without X or with the *seaf-cli* package [11].

Figure 2: The clearly structured Seafile web interface gives you access to the full feature set via the sidebar. The context menu shows the options for individual files.



Name ▲	Size	Last Update
2021-06-09T19_52_36+02_00.JPEG	455.0 KB	in a few seconds
2022-05-15T15_07_15+02_00.JPEG		in a few seconds
20190701_183301.jpg		in a few seconds
20200615_172031.jpg		in a few seconds
20220531_172852.jpg		in a few seconds
20220531_172901.jpg		in a few seconds
20220531_172907.jpg		in a few seconds



User	Plattform / Version	Device name	IP adress
linuxnews	linux / 8.0.7	blue	192.168.[redacted]
linuxnews	linux / 8.0.7	aura	192.168.[redacted]

Figure 3: Among other things, System Administration provides an overview of devices currently connected to the server. It is also used to create additional users and groups.

Listing 4: Installing the Client

```
$ sudo bash -c "echo 'deb [arch=amd64 signed-by=/usr/share/keyrings/seafile-keyring.asc] https://
  linux-clients.seafile.com/seafile-deb/focal/ stable main' > /etc/apt/sources.list.d/seafile.list"
$ sudo apt update
$ sudo apt install -y seafile-gui
```

Seafile's specialties are file synchronization and content sharing (Figure 4). After installation, the client synchronizes all data stored in the `Seafile/` folder with the server. Seafile detects even the smallest differences and synchronizes only the changes. Changes to files also trigger very practical versioning (Figure 5).

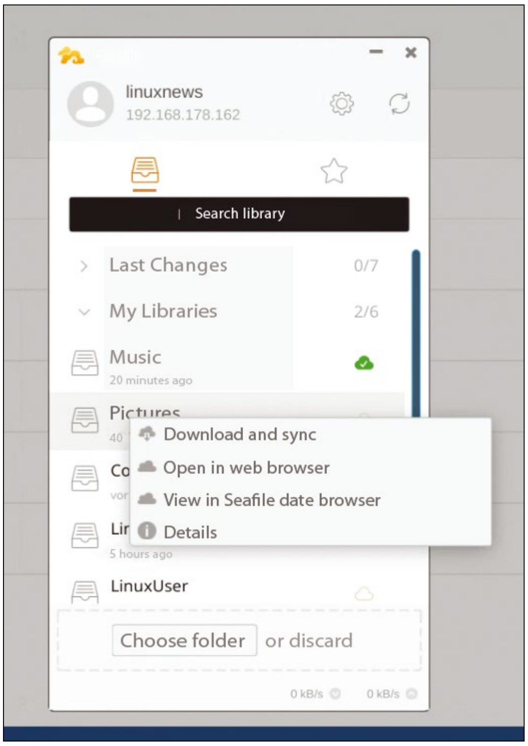
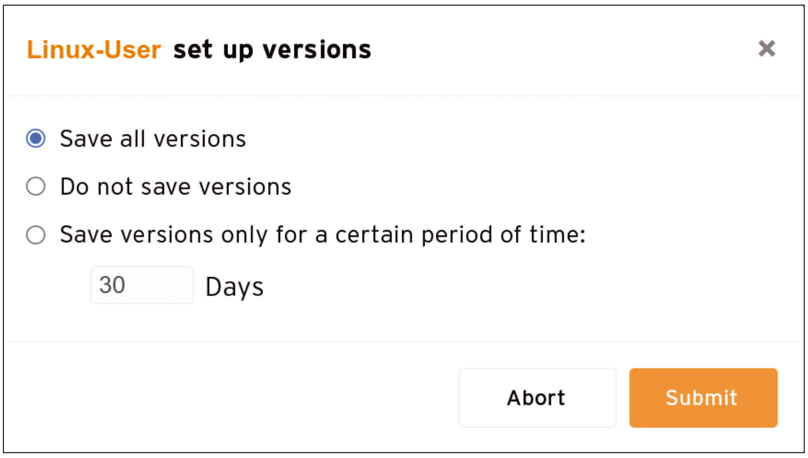


Figure 4: The Seafile client is available for both desktop and mobile devices. The desktop client shown here reveals its options in the context menu when you press the right mouse button.

This proves to be very useful, especially when several users are working on one file. Much like Git, this makes it possible to track who made what changes and when. Upgrading to a new version of Seafile is done in a separate folder, so that you can switch back to the old version in case of problems.

Content sharing relies on file shares that can be restricted to individual Seafile users. File and folder shares for third parties rely on public links like in Nextcloud. To collaborate on documents with other users, you may want to integrate OnlyOffice's online editors [12] or connect to Collabora Office [13] (Figure 6). If you don't need a full office suite, you can use the built-in Markdown editor for composing.

Figure 5: A library's context menu lets you save different versions of a file permanently or for a certain period of time. Use the *Set up versions* menu item for this.



Linux-User set up versions

● Save all versions

○ Do not save versions

○ Save versions only for a certain period of time:

30 Days

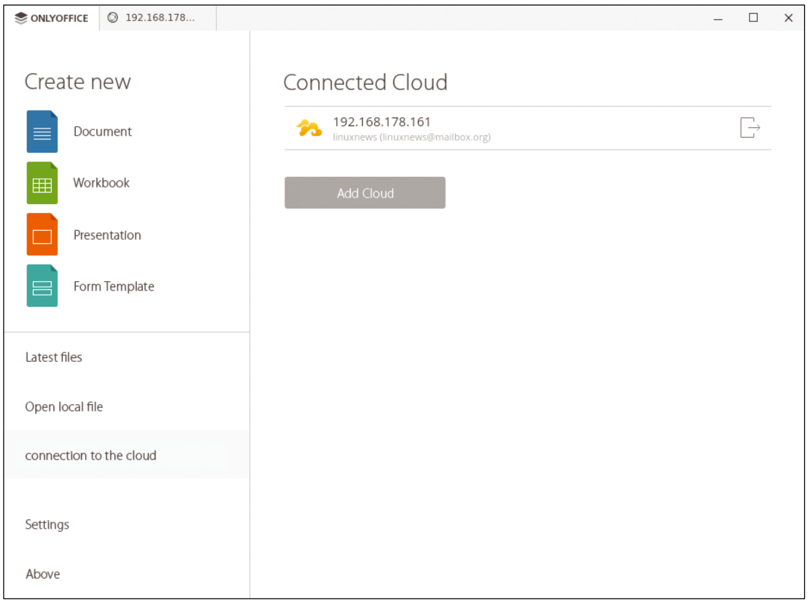


Figure 6: For collaborative work on documents, Seafile integrates the OnlyOffice and Collabora Online office suites.

Conclusions

If you're more concerned with file storage, synchronization, and content sharing than collaboration, Seafile is probably the right choice for you. However, collaborative work also works well, even though integrating the Office programs is more complicated than with Seafile's competitors. The documentation is detailed [14]. If you speak German (or can work with YouTube's auto-translate function for captions), there is a 10-part YouTube series in German [15] where you can also learn more about the program.

The biggest advantage of Seafile over Nextcloud or ownCloud is its superior speed for synchronization and access. This is mainly due to the memory model used. Unlike its competitors, Seafile uses block storage [16] instead of file storage. On the other hand, Seafile does not have any additional apps like ownCloud or Nextcloud do. So if you need to integrate a calendar, email, and other components, you will need to opt for a different solution. Otherwise Seafile is a good option. ■■■

Info

[1] Seafile: *https://www.seafile.com/en/home*
[2] GitHub: *https://github.com/haiwen/seafile*
[3] Seafile editions: *https://www.seafile.com/en/product/private_server/*
[4] Download: *https://www.seafile.com/en/download/*
[5] Docker: *https://manual.seafile.com/docker/deploy_seafile_with_docker/*
[6] Install script:*https://github.com/haiwen/seafile-server-installer*
[7] Manual installation: *https://manual.seafile.com/deploy/using_mysql/*
[8] Seafile on the Rasp Pi: *https://github.com/jojo243/seafile-docker-arm64*
[9] Docker Compose YAML:*https://download.seafile.com/d/320e8adf90fa43ad8fee/files/?p=/docker/docker-compose.yml*
[10] WebDAV: *https://manual.seafile.com/extension/webdav/*
[11] seaf-cli: *https://help.seafile.com/syncing_client/linux-cli/*
[12] OnlyOffice: *https://www.onlyoffice.com*
[13] Collabora: *https://www.collaboraoffice.com/solutions/collabora-office/*
[14] Office programs integration documentation: *https://manual.seafile.com/*
[15] YouTube series (in German): *https://www.youtube.com/watch?v=uS4c2DPZA6M&t=779s*
[16] Block storage: *https://cloud.netapp.com/blog/cvo-blg-what-is-block-storage-pros-cons-and-comparisons*

■■■

Shop the Shop
shop.linuxnewmedia.com

Become a LibreOffice Expert

LibreOffice Expert
FREE DVD!
LibreOffice Full Version
Dive deep into the world's greatest free office suite
2022/23 Edition
LibreOffice Expert
Edit and Save MS Office Files
Write Your Own LO Macros
Save time and automate common tasks
Create Professional:
Text Documents
Spreadsheets
Presentations
Databases
Replace MS Office and Google Docs!
LibreOffice
Includes full versions for Windows, macOS, and Linux
WWW.LIBREOFFICEEXPERT.COM
LibreOffice 7.3.5
Installation files for Windows, macOS, and Linux
Lots of extras inside! ■ Handbooks ■ Extensions ■ Videos
2022/23
LibreOffice Expert
DVD

Explore the FREE office suite used by busy professionals around the world!

Create Professional:

- Text Documents
- Spreadsheets
- Presentations
- Databases

Whether you work on a Windows PC, a Mac, or a Linux system, you have all you need to get started with LibreOffice today. This single-volume special edition will serve as your guide!

Order online:
shop.linuxnewmedia.com/specials

For Windows, macOS, and Linux users!

FOSSPicks

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

Not content with a Steam Deck full of too much choice, Graham has spent this month installing even more open source software and games on his: The curse of choice.

BY GRAHAM MORRISON

Sound tracker

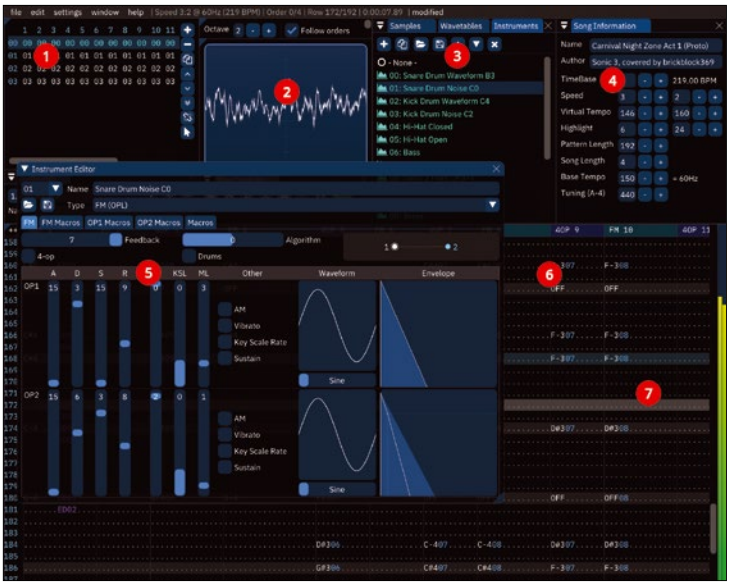
Furnace

Necessity is the mother of invention, but there are few better examples of this in practice than the humble music tracker. A tracker is little more than a scrolling spreadsheet of hexadecimal values spread across several columns, with each column group representing a channel of sound or instrument. Each row in each column is a potential pitch value, with additional columns inside the group used for effects and sound control. With a small shim to handle the audio hardware, this data could be played on 1980s hardware directly from memory and without modification, creating beautiful music for games and demos on early home computers with very little RAM and few CPU resources. These limitations were what forced the invention of trackers, and in so doing, created the incredibly distinctive chiptune music that continues to thrive to this day, in an age of seemingly limitless storage and compute power.

There are a few trackers still being developed for Linux, including the venerable Radium and Schism trackers (and the amazing but commercial Renoise). But it's uncommon to see an entirely new open source project appear, and utterly remarkable to find one that also emulates the sound of every significant sound chip of that early era. And yet, this is what Furnace is and does. Furnace is a tracker that includes sound engines to emulate over 50 different sound chips from the golden chiptune era, complete with their individual parameters, which can all be controlled through the data view. These include FM modules from the Sega Genesis, FM Towns, arcade machines, and many PC sound cards, plus sample playback chips found in the Commodore Amiga, wavetable synthesis used by Konami and Namco arcade machines, beeps from speakers, the ZX Spectrum and Game Boy, plus brilliant recreations of the Commodore 64 SID chip and Mikey in the Atari Lynx.

Each system is added on demand from the File menu, and each will add one or more new column groups to the note editor. The number of groups corresponds to the hardware capabilities of the device you're adding, such as four channels for the Amiga or three for the SID chip. The columns within each group also represent the unique hardware characteristics for that device, such as the waveform, cutoff filter, envelope amounts, or glissando. This is all brilliantly documented for each system in the online reference guide and makes complete sense if you're already familiar with music trackers. There's also an independent instrument editor where all these parameters can be used to create and save an individual sound, which can be switched between in your own music.

The end result of all this is that Furnace becomes a chiptune powerhouse, where you can summon the sound of almost any system and program music with full control of all their capabilities, almost byte for byte. And we've not even mentioned the oscilloscopes, effects, mixers, and even a sample editor that can be opened alongside your composition. The whole ensemble can then be stored in its own format or exported as a scene-acceptable DefleMask module (although you can't import them yet). It's all hugely comprehensive and fun to play with, and completely open source.



1. Sequencer: Despite its low level, songs can still be built from sequences of patterns. **2. Oscilloscope:** View a waveform of the mixed output or for each individual track. **3. Instruments:** These are the heart of Furnace and allow you to emulate almost every popular audio system from the 1980s. **4. Global options:** Set the tempo, pattern, and song length. **5. Instrument editor:** An instrument has its own editor, unique to the capabilities of each sound engine. **6. Pattern editor:** This is where notes are created for each track and instrument. **7. Effects:** Add all kinds of modulation and effects to your tracks with the extra data columns.

Project Website

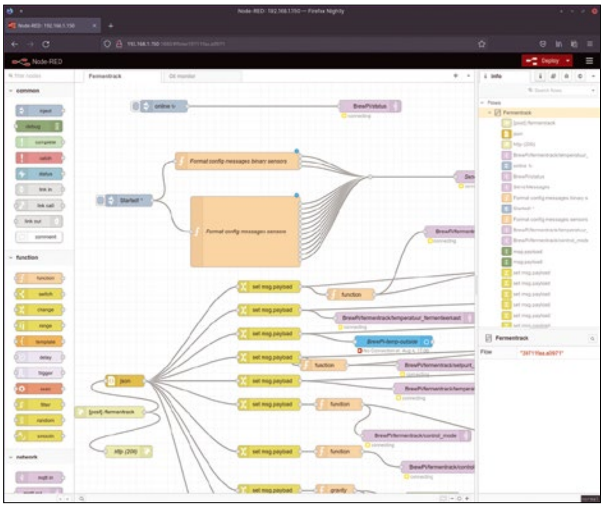
https://github.com/tildearrow/furnace

Event IDE

Node-RED 3

Node-RED is a brilliant project and one that's had significant coverage in these pages, most recently when version 2.2.0 appeared. Version 3.0 is another major upgrade and consolidates its brilliance at being the glue for your various data sources while remaining easy to use and supremely powerful. In this release, for example, the addition of a right-click context menu lets you access all the most common features from where your cursor is, which is exactly where you need this kind of menu. This is important because it wasn't always clear what you could do and where in previous versions, and you would often need to switch between editing and adding nodes in the flow and then go back to the inspector on the right to configure the elements you wished to use.

Another great feature is the effortless upgrade process that takes you from 2.x to 3.x era. There are several ways to do this, but the well-documented install and upgrade script worked faultlessly on a Raspberry Pi. With a single restart of the service, we had old flows running just as well on the new version, complete with lots of shiny new potential. You can now add junction nodes to better see and route your data paths, for instance, which is especially convenient when you have lots of destinations for a single source, because the nodes hugely reduce the clutter. Instead of multiple lines from a single place, you can now route a single line to a point where you add the junction and route the ongoing connections from there. You will also find that new nodes have sensible default names, and while debugging their full path is revealed via a new tooltip. All of this makes Node-RED a real pleasure to use and one of the best ways to build bespoke messaging and data flows for your own hardware.



The new junction can be used by selecting multiple connections and selecting *Insert | Junction* from the equally new context menu.

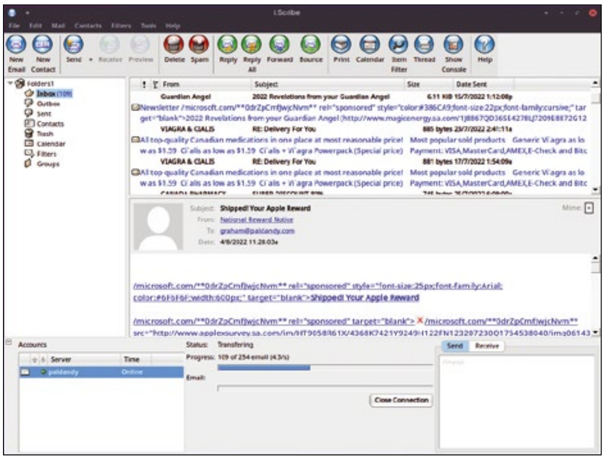
Project Website
https://nodered.org

Email client

Scribe

Recently, Mozilla's Thunderbird email application has been trying to reboot its popularity with a promising modern redesign and a partnership-plus-rebranding of the open source Android client, K-9 Mail. This is great news for Thunderbird and new potential users, but it also helps to remind us that desktop email clients are still a thing, and that they can be a great alternative to web-based clients. Scribe is another great alternative. Although it's not as full-featured or as capable as Thunderbird, it is less than 10MB to download and can be run from its unarchived directory. It's also fast to load, easy to navigate, and convenient even on low-powered devices with very little storage. There are also versions for macOS and Windows, which means you could theoretically run executables for each off a USB stick when traveling and not knowing what kind of computer you might have access to.

Despite its size, Scribe is also very capable, at least for day-to-day email work. It supports POP3, SMTP, and IMAP, with or without SSL. It can therefore talk to every major email vendor and even includes built-in spam filtering. You first need to mark about 100 spam messages as spam before the Bayesian filter can start to accurately predict whether new messages follow the same pattern, but it works well if you already have a large folder or previously filtered spam messages – which you will if you're hosting your own email server. The application design is reminiscent of Java or Gnome Evolution circa 2012, with a folder list on the left, messages on the right, and account selection below. The message list shows a preview of an email itself and will display the full body in the pane below when a message is selected. It can feel a little dated, but that's also what makes email such a survivor in a world of unlimited online distractions and clickbait, and for that, we're grateful.



Scribe has been around since 1999 and is a small, fully functional email client that can run from almost anywhere.

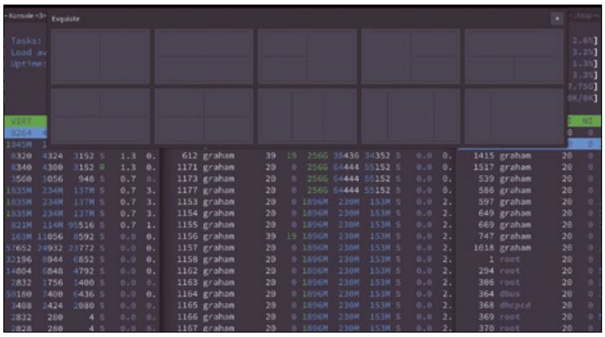
Project Website
https://memecode.com/scribe/

Plasma layout manager

Exquisite

One of the best things about the KDE Plasma desktop is that, while it's a great experience in its own right, it's also a brilliant desktop construction set. This means you can customize and augment the default experience in almost limitless ways, using either the inbuilt settings provided by KDE or by creating your own scripts to add more programmatic functionality. This is best seen in the variety of scripts that change the window's arrangement behavior. We've covered a couple before, including KWin Tiling (to turn the desktop into a tiling window manager with its own rules) and Kröhnkite (which performs a similar role with dynamic layouts). Exquisite is another one, developed to mimic the good bits of Windows 11 window management. It's a great option for users who may want to experiment with different styles of windows management but may lack the confidence to go all-in on something that requires complicated keyboard shortcuts.

Exquisite is installed via either the *Get New Scripts* button in the KWin Scripts configuration panel or by importing the tar archive manually. It's triggered by pressing a specific keyboard shortcut that can be reconfigured from KDE's global shortcuts panel and defaults to Ctrl+Alt+D. What makes Exquisite a good introduction to window management is that pressing this combination will show a pop-up of various layouts, and you can choose a location on one of these to move the currently selected window to. There are 10 layouts to choose between, and they can all be modified, and even created with simple scripts, to better suit your needs and screen layouts. The panel stays on the screen and easily lets you choose other windows to move, and it can be configured to hide itself when the screen is full or after a period of time. It's a great way to get into tiling and much easier to use than remembering many keyboard shortcuts.



Exquisite, an easy-to-use window arranger for KDE Plasma desktops, is a great introduction to tiling in general.

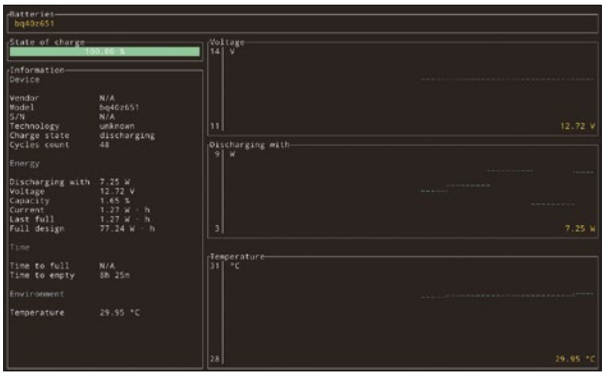
Project Website
https://github.com/qewer33/Exquisite

Battery monitor

battop

One surprising thing the Steam Deck has done for mobile Linux is make battery life a first class issue again. As brilliant as the device is, its 15W graphics hardware can easily drain its humble 40Wh battery in less than two hours. While the Steam Deck does include some excellent performance tweaking and analysis overlays, these aren't available in desktop mode. This is where `battop` can help, and also help any other Linux running on a battery, because it tells you everything you need to know about your batteries. It runs from your terminal, connects directly to the battery management subsystem, and renders everything it can divine from your hardware to a brilliantly functional terminal-based user interface.

The main display is split into six sections spread across two columns on the left and right. The large text-based pane on the left lists your battery's capabilities and consumption details. These include the battery's vendor ID, alongside consumption, voltage, capacity, time to full, time to empty, and the ambient temperature. Above this is a histogram showing the percentage of battery left. On the right are three histograms charting the temperature, voltage draw, and consumption per watt hours of the battery so you can see consumption over time. The minimum and maximum values are dependent on the value fluctuation but are labeled to show their limits, and each chart is annotated with the current value for each. All of this works just like a CPU usage chart and is really useful when you want to see the impact that recent actions may have had on battery performance. As the `battop` website says, it's the battery equivalent of something like `gotop` or `htop` for process and memory management.



```
Batteries
 bq40z651

State of charge
 100.00 %

Information
Device

Vendor            N/A
Model             bq40z651
S/N               N/A
Technology        unknown
Charge state      discharging
Cycles count      48

Energy

Discharging with  7.25 W
Voltage           12.72 V
Capacity          1.65 %
Current           1.27 W · h
Last full         1.27 W · h
Full design       77.24 W · h

Time

Time to full      N/A
Time to empty     8h 25m

Environment

Temperature       29.95 °C
```

Unlike Apple hardware, access to Linux battery information is unrestricted and accessible to anyone with `battop`.

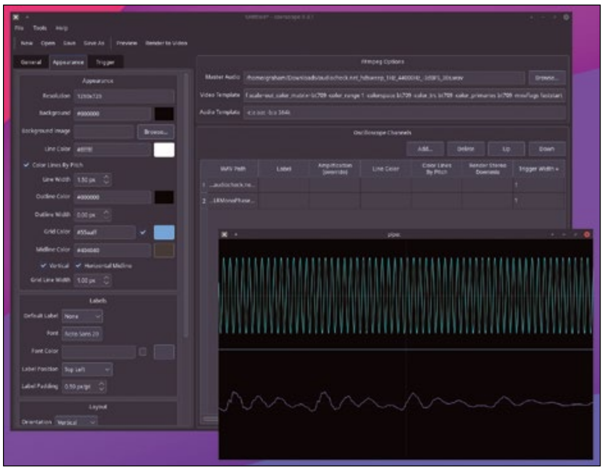
Project Website
https://github.com/svartalf/rust-battop

Music visualizer

corrscope

The Internet is full of videos containing just music with some simple visualization to help provide some feedback while you're listening. They're easy to create and useful if you want to share your music or create an easily accessible playlist, but it's notoriously difficult to create a meaningful visualization. The biggest problem is generating something useful that can be synchronized to the sound. One approach is to render the output from a music player visualizer, such as projectM, which we've previously looked at here. Another approach is to use corrscope, which takes one or more audio files and generates an animated oscilloscope view of those inputs. These show the changing frequencies in the audio, and they can be notoriously difficult to sync to the actual sound.

Corrscope solves the sync problem in exactly the same way a real oscilloscope does, triggering synchronizing points from the upwards or downwards slopes of a waveform. These triggers can be adjusted just as they can on the real hardware too, although the default values worked with everything we tried. The waveforms themselves can be customized by changing the line color, the line width, the background grid, and either virtual or horizontal layout. Most importantly, you can generate waveforms for more than one input and split these into totally separate oscilloscopes. This works well for stereo tracks but works best with chiptune music. This is because each channel can be separated by the composition software and each channel usually consists of simple waveforms. These look excellent when they're pulsing away together in a grid of many mini oscilloscopes. Any changes you make in the GUI can be previewed immediately by clicking on *Preview*. This fires up FFmpeg, the only dependency, and shows you a low quality version of the eventual output. If you like what you see, this high quality version is rendered at your chosen resolution and can be shared however you prefer.



With corrscope, you can construct oscilloscope visualizations of your multi-channel audio files, which are brilliant for video.

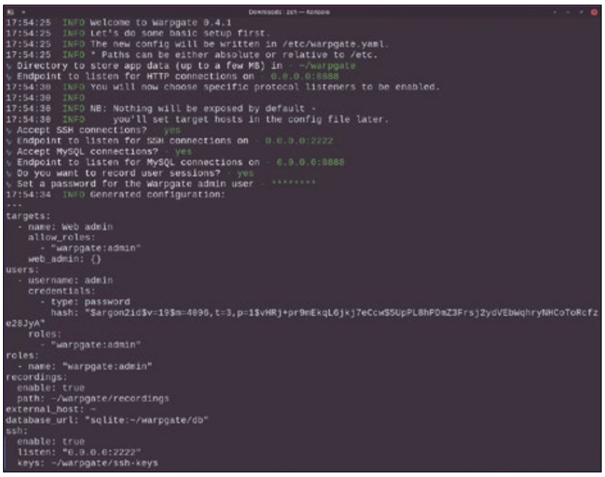
Project Website
https://github.com/corrscope/

SSH bastion

Warpgate

A bastion host is a server that lives just outside your network, either in a DMZ or beyond the safety of your firewall, from where it can bridge or forward certain services from your network to a remote host. Its location makes it prone to attack, but bastion servers are typically hardened and audited to ensure nothing can breach their simplified services. Warpgate is a simple bastion service provided as a single binary executable you should build yourself and run on your own hardened server. It can authenticate and forward specifically permitted SSH, HTTPS, and MySQL connections, which can also be fully logged and recorded for future playback or auditing. This is an essential part of making sure a service hasn't been compromised, and it's what differentiates Warpgate from setting up your own SSH proxy, for example.

To set things up, the `warpgate` command takes a single `setup` argument that will step you through its configuration. This includes the location to store the credentials data, the SSH listening port, and the port for the admin web interface. Before the service can be useful, however, you need to add authentication for either the SSH or HTTP targets. This starts with retrieving the SSH keys from the web interface or command line and adding them to your local `authorized_hosts` file, just as you might with any other SSH server. A target address now needs to be added to the Warpgate configuration file simply by listing its address, port, and optional username. It's a similar process for adding MySQL and HTTP targets. Correct configuration can be checked through both the command line and the web interface, and you can now connect to your remote host by connecting to Warpgate. The connection is forwarded automatically. While there is some text output to show the target is working, it is otherwise transparent to the client.



```
17:54:25  INFO Welcome to Warpgate 0.4.1
17:54:25  INFO Let's do some basic setup first.
17:54:25  INFO The new config will be written in /etc/warpgate.yaml.
17:54:25  INFO * Paths can be either absolute or relative to /etc.
✔ Directory to store app data (up to a few MB) in · ~/warpgate
✔ Endpoint to listen for HTTP connections on · 0.0.0.0:8888
17:54:30  INFO You will now choose specific protocol listeners to be enabled.
17:54:30  INFO
17:54:30  INFO NB: Nothing will be exposed by default -
17:54:30  INFO     you'll set target hosts in the config file later.
✔ Accept SSH connections? · yes
✔ Endpoint to listen for SSH connections on · 0.0.0.0:2222
✔ Accept MySQL connections? · yes
✔ Endpoint to listen for MySQL connections on · 0.0.0.0:8888
✔ Do you want to record user sessions? · yes
✔ Set a password for the Warpgate admin user · ********
17:54:34  INFO Generated configuration:
---
targets:
  - name: Web admin
    allow_roles:
      - "warpgate:admin"
    web_admin: {}
users:
  - username: admin
    credentials:
      - type: password
        hash: "$argon2id$v=19$m=4096,t=3,p=1$vHRj+pr9mEkqL6jkj7eCcw$5UpPL8hPDmZ3Frsj2ydVEbWqhryNHCoToRcfz
e28JyA"
    roles:
      - "warpgate:admin"
roles:
  - name: "warpgate:admin"
recordings:
  enable: true
  path: ~/warpgate/recordings
external_host: ~
database_url: "sqlite:~/warpgate/db"
ssh:
  enable: true
  listen: "0.0.0.0:2222"
  keys: ~/warpgate/ssh-keys
```

Warpgate sits in the danger zone beyond your network so that local clients are better protected from the dangers of system-wide access to the Internet.

Project Website
https://github.com/warp-tech

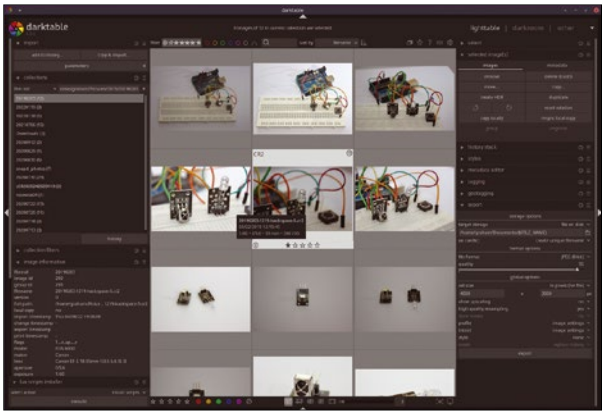
Photo editor

darktable

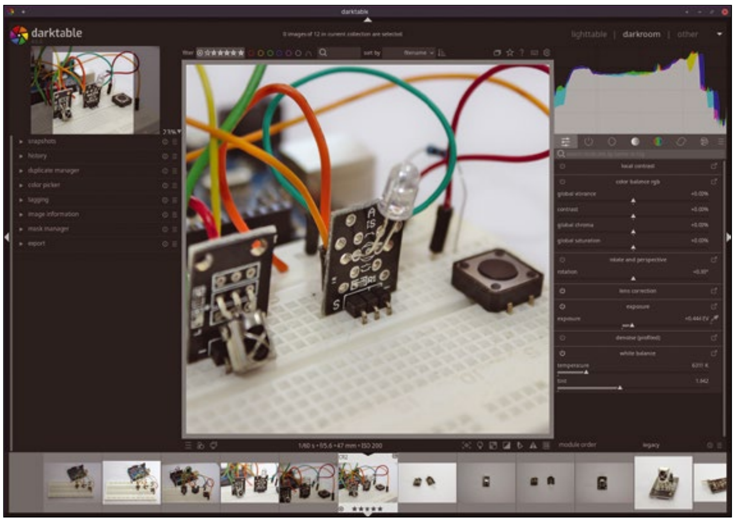
We often mention the triumvirate of open source RAW photo editors: digiKam, RawTherapee, and darktable. However, it's the last one we find ourselves using the most. Darktable always used to chase Adobe Lightroom for features and capabilities, but since we last looked at darktable properly a few years ago, we think it has finally surpassed its inspiration and become a powerful photo manipulation and management platform for serious photographers in its own right. We used darktable for our photography in Linux Voice, and it saved us from a complete lack of skill and ability on many occasions, especially when it came to under- or over-exposed photos where darktable can magically turn a black smudge into something we could print. This is all thanks to the various RAW image formats darktable supports, because these files will often contain far more proprietary detail from the camera sensors than can be encapsulated in a JPEG, for example, and every camera is different.

Coming from Gimp or Krita, darktable can be a little intimidating to get your head around. There are two main view configurations which you can switch between by selecting either *lightable* or *darkroom* mode in the top right corner. These are analogous to old predigital processes, where you'd look at negatives through light to see which ones you wanted to process, before doing the process in a dark (red-lit) room. In *lighttable* mode, you can organize your photos into collections, give them a star rating, add tags, and colorize their categories, all for easy storage and retrieval. In *darkroom* mode, you can edit the photos by enabling one or more processing modules that can do anything from crop an image to compensate for specific lens distortion. The results can be miraculous and utterly professional.

Version 4 is a major update and includes over 1,600 separate changes even from the previous 3.8 release, let alone the countless updates darktable has received since we last looked at it properly with the release of version 2.2.0 in 2016. The most important new features all deal with color. You can now create a target exposure or color profile, for example, and apply these to a collection of images for consistency. Fix one photo, and you can automatically match the same fix across a set. The filmic module, introduced in 3.8 to mimic the high dynamic range color response of classic film and derived from a Blender module, is now far more flexible and allows for more saturated colors. Similarly, highlights can now be reconstructed with a new "guided Laplacian" method, deriving more intelligent and natural results from regions close to the highlight. All of these might sound deeply theoretical, and unlikely to be useful to smartphone users, for instance, but this isn't true. Many phones can still save their photos as RAW images, which can all be processed and managed with darktable. It's ideal if you ever need to print something out, create a photo frame of an image, or share something for publication. And darktable is ideal if, like us, you never properly light a scene or change the exposure settings on your camera to compensate for poor or bright lighting. Simply trust in darktable to fix things.



To be able to support new cameras and their RAW images, darktable is asking users to contribute their own photos under a CC license.



The user interface has been completely rewritten for this release to improve the look and consistency.

Project Website

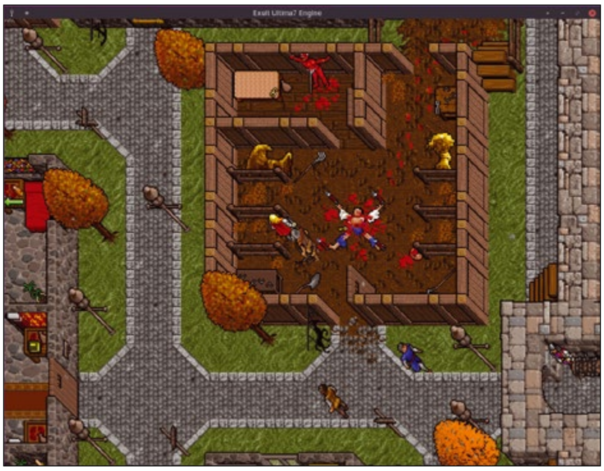
https://www.darktable.org/

Role-playing engine

Exult 1.8

Doom notwithstanding, Ultima VII was one of the most brilliant and influential games of the early 1990s. It was a role-playing game with a top-down isometric graphic style that is still being copied today and whose graphics engine is part of the legacy behind Ultima Online, 25 years after its launch. But most importantly, the game itself was one of the first open world adventures where you could go anywhere and do anything. As you took your time to travel from one town to the next, Britannia's denizens would go about their days and seasons as they might in 15th-century England, tending to fields and cattle before going down to the local tavern for a beer and a meal. Meanwhile, your character – the avatar – starts on a quest to solve a grizzly murder that only ends after trekking across miles of landscape and an entire expansion called the Forge of Virtue.

Exult is a long lived project that runs Ultima VII and its expansions on modern hardware. It's available for almost any platform, including Android, and is easy to build yourself if needed. Like many such projects, it enables anyone with the original story files to run an old MS DOS game to be pulled into the modern era, complete with sounds, upscaled graphics, and modern controllers, if you choose. Version 1.8 of Exult, which has been released to coincide with the 30-year anniversary of the original, is still adding features, including work on the game editor, now ported to GTK+ 3; added background effects; and a proper annotation in the ending sequence. This last fix is likely to have endured because playing Ultima is so absorbing you never want to actually finish the game, which is still brilliant all these years later.



While Exult was built specifically to play Ultima VII, it has since expanded into a map editor and is now even capable of creating independent Ultima-like games.

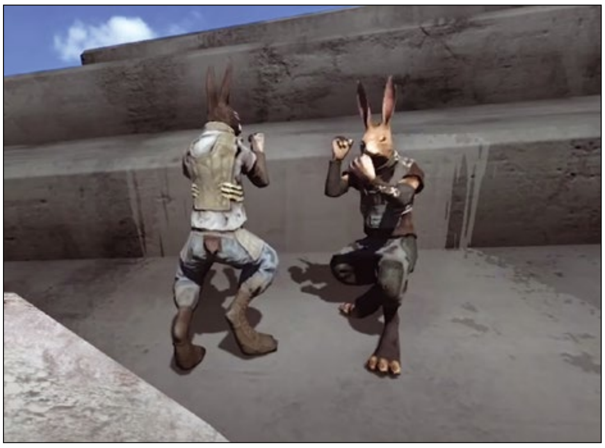
Project Website
http://exult.sourceforge.net

Martial arts

Overgrowth

Overgrowth is one of those rare pieces of software that has already lived an independent life as a commercial and proprietary product before getting a new lease on life as an open source project. It's a game that has been available on Steam and GOG for the best part of a decade (and still is) and one with a keen user base who love its rabbit-themed wild martial arts physics brawling. Wolfire Games, its developer, really needs to be commended for this approach, as well as always providing a native Linux version. It even has a history of doing the same thing: Its earlier rabbit-themed title, Lugaru, is also open source.

Although both titles feature rabbits, neither focus on the traditionally cute side of their in-game anthropomorphic avatars. Instead, they're more sinister, and more like something from *Donnie Darko* than Donald Duck. Your mission is to save your fellow rabbits by fighting various other animals littered (pun intended) across various landscapes and levels. You do this with twitchy Matrix-style kung-fu which you learn by jamming the game controller in any direction it still functions. In this way, your rabbit can hop across hectares of landscape, dealing bloodthirsty blows to your adversaries in true fully bunnysploitation style. It's a lot of fun and doesn't require too much thought or strategy, although the gore might be too much for younger players. The graphics were dated even when the game was released, but this adds to its charm. This is perhaps something that can now be addressed because, most importantly, the game is now open source. This is thanks to the developer moving on to a new project and not wanting their old one to slowly decompose. If only some other developers could do the same.



Overgrowth is by the same developer behind the similarly styled Lugaru, which can also be replayed here as a second campaign.

Project Website
https://github.com/WolfireGames/overgrowth/

Accessing iTunes XML metadata with Python

Unlock Your iTunes Data

Read and manipulate data from your iTunes XML in order to share music metadata between applications and across devices.

BY JOHN COFIELD

The Extensible Markup Language (XML)[1] is a widely used markup language and text file format for storing and exchanging arbitrary data. A wide variety of applications, including the Apple iTunes application, use the XML format for exchanging or storing library data. In this article, I will show you how to use Python to read and manipulate data from an iTunes [2] XML library file.

iTunes Library File

iTunes divides its content into two separate categories: media and metadata. Media (music and video) are saved in separate files from metadata. Throughout the rest of this article, I will focus on music information – artist, song title, album title, track number, genre, release year – and refer to it as metadata. The iTunes metadata files have either a `.itl` or `.xml` extension (Figure 1).

Why Choose XML?

If you are a music lover like me, there may be times when you want to exchange music metadata between a variety of applications on a variety of digital devices. In my case, I have imported iTunes metadata into database applications to run SQL queries. Some applications use proprietary formats to get optimum performance and efficiency but limit access with proprietary products. The iTunes `.itl` format is an example. On the other hand, there are open formats that place more of an emphasis on wide availability and ease of development. XML is in the open format category.

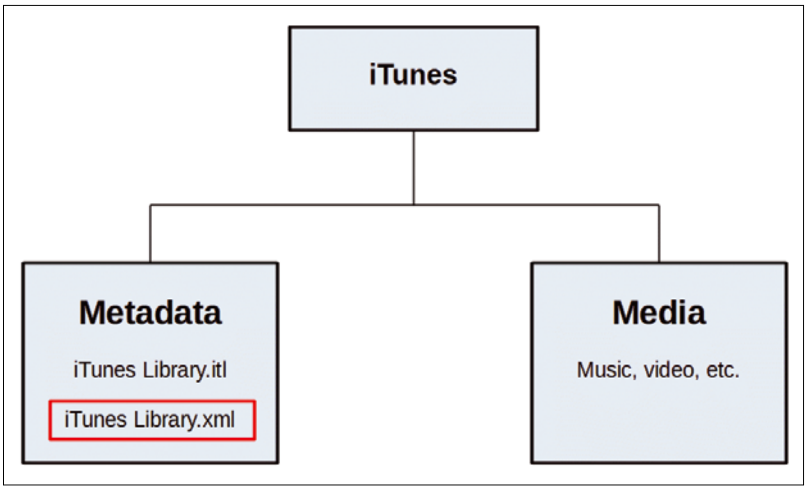


Figure 1: The iTunes XML metadata file hierarchy.

Apple has historically used XML files to export library and playlist metadata to external applications. While the native format for the iTunes library is a binary file with a `.itl` extension, users can configure iTunes to automatically save an XML copy of the library via Advanced preferences or manually via the *File* | *Export* menu. If you prefer to export a subset of the library, the *File* | *Export* menu allows you to save playlists to an XML file.

Advantages of the XML format are that it is both human-readable and machine-readable. The benefit of having a human-readable file is being able to easily inspect and diagnose the data with a simple text editor. Machine readability makes it easy for applications to access the metadata via an application procedural interface (API). Many applications that we use every day – text editors, word processors, spreadsheet editors, presentation tools, and graphics editors – either use XML as the native format or to exchange data with other applications.

Python XML Module

The Python standard library has structured markup tools that include modules for processing XML. In this article, I will describe how metadata in iTunes XML files is organized and how to use the Python `xml.etree.ElementTree` [3] module to navigate and access that data. In order to extract metadata from an iTunes XML playlist or library file, it is useful to understand how the XML file is structured. Songs within a library or playlist file are organized as a list of tracks identified by XML tag `<key>Tracks</key>`. Metadata for each track consists of a dictionary of key-value pairs.

Listing 1: DTD in DOCTYPE Declaration

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
```

The XML language allows for data types to be defined in a Document Type Definition (DTD) that can be declared either inline inside an XML document or as an external reference. Apple traditionally uses property list files for their application configuration, thus the term “property list” (`plist`) in the declaration (Listing 1). Because metadata is structured as elements in an XML tree, the `xml.etree.ElementTree` module, which stores tree elements hierarchically, is an appropriate way to get to those elements. Elements are defined in the external public DTD link specified in the `DOCTYPE` declaration on the second line of the exported music file.

The data types defined in this DTD are: array, data, date, dict, real, integer, string, true, and false. XML elements are also declared in the DTD file. An element is a logical document component that begins with a start tag and ends with a matching end tag. Each metadata element contains a string that is either a key or a value in the music dictionary. Metadata for song tracks is organized as key-value pairs as illustrated in Listing 2, where a `<key>` tag with a text string is followed by a `<string>` or `<integer>` tag with a text string.

In Table 1, you will see Python code snippets to show you how to use the listed methods and attributes to access metadata in XML elements.

Build a Tree

The `parse()` method takes input from an XML file object and builds a hierarchical tree as a collection of elements. Each element may have multiple child elements or no child elements. The root element is instantiated by the `getroot()` method (Listing 3).

Track Dictionary

Track metadata is a hybrid of nested lists and dictionaries with key-value pairs. The dictionary containing the metadata is nested two levels below the root. The `find()` method in the two statements shown in Listing 4 finds the next level descendant with a `<dict>` tag.

The second-generation dictionary contains a list of all track dictionaries in the library or playlist. After locating the track dictionaries, you can work on the child elements that contain the metadata for each track.

Element Tags and Text

To get the track metadata, you need to find the track metadata tags and then retrieve the text from those tags. The statement in Listing 5 uses the `findall()` method to find all child elements of each track dictionary beneath `dict_gen2` and creates a list object named `tracklist` that contains a list of dictionaries.

Next extract from `tracklist` nested lists of child elements that have track metadata. Not all `<dict>` tags will contain the metadata that you want, so you can use the `Artist` text string of the `<key>` tag to identify the desired dictionaries. See Figure 2 and Listing 6. The element text is accessed with the `element.text` attribute.

The `itunes_music` list object is a list of all song tracks contained in the input XML file. At this point, you may want to inspect the resulting metadata in the `itunes_music` list. Each `<key>` tag is followed by the corresponding `<string>` or `<integer>` tag. The code in Listing 7 assumes that the `tagtrue()` function was previously defined (for brevity, code not shown here), tests for `<key>` tag text that matches the desired

Listing 2: XML Track Metadata Example

```
<key>Name</key>
<string>Justice's Grove</string>
<key>Artist</key>
<string>Stanley Clarke</string>
<key>Album</key>
<string>East River Drive</string>
<key>Genre</key><C>
<string>Jazz</string>
<key>Track Number</key>
<integer>1</integer>
<key>Year</key>
<integer>1993</integer>
```

Listing 3: Build Tree from XML File

```
tree = ET.parse(xmlinfile)
root = tree.getroot()
```

Listing 4: First and Second Generation Dictionaries

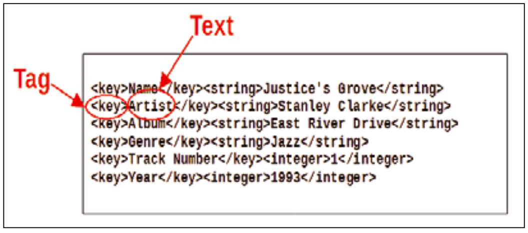
```
dict_gen1 = root.find('dict')
dict_gen2 = dict_gen1.find('dict')
```

Table 1: Accessing Metadata in XML Elements

Code Snippet	Action
`parse()`	Sources and parses an external XML file or file object
`getroot()`	Returns the root element for the tree
`find()`	Returns an instance of the first specified sub-element
`findall()`	Returns a list containing all matching elements in document order
`tag`	Specifies the element name as defined in the DTD
`text`	Specifies the characters between the start and end tag

Listing 5: Tracklist Metadata Dictionaries

```
tracklist=list(dict_gen2.findall('dict'))
```



```
<key>Name</key><string>Justice's Grove</string>
<key>Artist</key><string>Stanley Clarke</string>
<key>Album</key><string>East River Drive</string>
<key>Genre</key><string>Jazz</string>
<key>Track Number</key><integer>1</integer>
<key>Year</key><integer>1993</integer>
```

Figure 2: Tag and text.

metadata strings (Figure 3), and then prints the key and value pair text strings to your screen.

After confirming the metadata, you can then utilize it as you need. Because the list is organized as a two-dimensional array, I will represent it in table form as it would appear in a spreadsheet or database table (Figure 3). Note that the `i` loop (or outer loop variable) represents rows, and the `j` loop (or inner loop variable) represents columns.

Summary

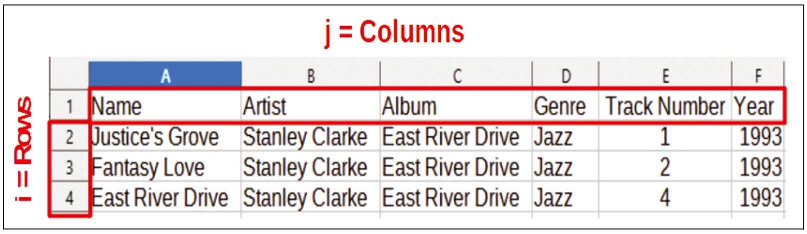
While I don't cover SQL APIs in this article, my ultimate goal in this exercise is to create and update an SQL music database with this metadata. You may choose to use it for other purposes, such as analytics, sharing with other applications, or rendering in HTML for website display. Because XML is a widely used industry standard, there is an abundance of libraries available in almost every programming language. In this case, I've chosen Python. Whatever your goal, it is useful to have a solid understanding of how iTunes music metadata is defined and organized so you can decide how to get the metadata that you need. ■■■

Listing 6: Create Metadata List for All Tracks

```
itunes_music = []
for item in tracklist:
    x = list(item)
    for i in range( len(x) ):
        if x[i].text == "Artist":
            itunes_music.append( list(item) )
```

Listing 7: Display Metadata

```
for i in range(len(itunes_music)):
    for j in range(len(itunes_music[i])):
        if tagtrue(itunes_music[i][j].text):
            print(itunes_music[i][j].text\
                , itunes_music[i][j+1].text)
```



	A	B	C	D	E	F
1	Name	Artist	Album	Genre	Track Number	Year
2	Justice's Grove	Stanley Clarke	East River Drive	Jazz	1	1993
3	Fantasy Love	Stanley Clarke	East River Drive	Jazz	2	1993
4	East River Drive	Stanley Clarke	East River Drive	Jazz	4	1993

Figure 3: Loop index rows and columns.

Info

[1] XML: *https://www.w3.org/standards/xml/schema*

[2] iTunes Package Music Specification 5.3.14, PDF: *https://help.apple.com/itc/musicspec/en.lproj/static.html*

[3] xml.etree.ElementTree: *https://docs.python.org/3/library/xml.etree.elementtree.html*

The Author

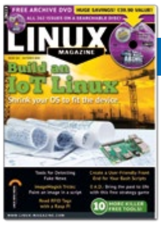
John Cofield is a retired software marketing manager in Northern California. His training is in electrical engineering, and he has worked at multiple Silicon Valley semiconductor and software companies. His nontechnical interests include jazz music ranging from modal to fusion.

■■■

LINUX NEWSSTAND

Order online:
https://bit.ly/Linux-Newsstand

Linux Magazine is your guide to the world of Linux. Monthly issues are packed with advanced technical articles and tutorials you won't find anywhere else. Explore our full catalog of back issues for specific topics or to complete your collection.



#263/October 2022

Build an IoT Linux

The most amazing thing about Linux is its flexibility. Linux systems run on the biggest computers in the world – and on many of the diminutive devices that populate your home environment. If you've always wondered how developers adapt Linux to run on tiny tech, you'll appreciate this month's stories on Buildroot and the Yocto project.

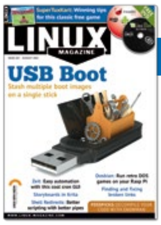
On the DVD: Linux Magazine Archive issues 1-262

#262/September 2022

Beyond 5G

Behind the scenes, the cellular phone network has always been the preserve of highly specialized and proprietary equipment, but some recent innovations could be changing that. This month we explore the Open RAN specification, which could one day allow more of the mobile phone network to operate on off-the-shelf hardware.

On the DVD: openSUSE Leap 15.4 and MX Linux 21.1



#261/August 2022

USB Boot

Live boot was such an exciting idea 15 years ago – just carry a CD with you and boot from anywhere. But old-style boot CDs had some limitations. Today's USB boot tools solve those problems plus offer a feature that no one even thought about back then: access to several boot images on a single stick.

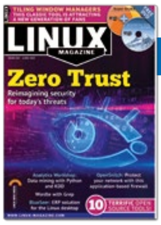
On the DVD: Linux Mint MATE 20.3 and FreeBSD 13.1

#260/July 2022

Privacy

If you are really serious about privacy, you'll need to lean on more than your browser's no tracking button. Those who need anonymity the most depend on the Tor network – a global project offering safe surfing even in surveillance states. We also look at Portmaster, an application firewall with some useful privacy features.

On the DVD: Ubuntu 22.04 and Fedora Workstation 36



#259/June 2022

Zero Trust

Twenty Years ago, everyone thought a gateway firewall was all you needed to stay safe from intruders, but recent history has told a different story. Today, the best advice is: Don't trust anyone. Your internal network could be just as dangerous as the Internet.

On the DVD: Zorin OS 16.1 Core and Super GRUB2 Disk

#258/May 2022

Clean IT

Most people know you can save energy by changing to more efficient light bulbs, but did you know you can save energy with more efficient software? This month we examine the ongoing efforts to bring sustainability to the IT industry.

On the DVD: Manjaro 21.2 Qonos and DragonFly BSD 6.2.1

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 *https://www.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 *info@linux-magazine.com*.

NOTICE

Be sure to check the event website before booking any travel, as many events are being canceled or converted to virtual events due to the effects of COVID-19.

SC22

Date: November 13-18, 2022

Location: Dallas, Texas

Website: *https://sc22.supercomputing.org/*

Come see your friends and colleagues in Dallas and explore incredible learning experiences at one of the largest HPC conferences in the world. If you can't attend in person, SC offers a robust digital experience with most sessions synchronously live-streamed and made available on-demand 24 hours after each session occurs.

Black Hat: Infosec on the Edge

Date: November 15-18, 2022

Location: Riyadh, Saudi Arabia

Website: *https://blackhatmea.com/*

Immerse yourself in the world of cybersecurity. Join 30,000+ infosec professionals at the largest cybersecurity gathering in the region. The Black Hat Middle East and Africa series will feature an unparalleled executive summit and a range of high-profile briefings, arsenal, trainings, and a gaming tournament. Learn more and get registered now!

Events

KubeCon + CloudNativeCon North America 2022	Oct. 24-28	Detroit, Michigan	https://events.linuxfoundation.org/
SREcon22 Europe/ Middle East/Africa	Oct. 25-27	Amsterdam, Netherlands	https://www.usenix.org/conference/srecon22emea
CyberDefenceCon 2022	Oct. 27-28	Orlando, Florida	https://cyberdefenseconferences.com/
All Things Open 2022	Oct. 30 – Nov. 2	Raleigh, North Carolina	https://2021.allthingsopen.org/save-the-date-2022/
SeaGL GNU/Linux Conference	Nov. 4-5	Virtual	https://seagl.org/
SC22 (Supercomputing 2022)	Nov. 13-18	Dallas, Texas	https://sc22.supercomputing.org/
Open Source Monitoring Conference	Nov. 14-16	Nurember, Germany	https://osmc.de/
Black Hat: Infosec on the Edge	Nov. 15-17	Riyadh, Saudi Arabia	https://athack.com/
Open Source Summit Japan	Dec. 5-6	Yokohama, Japan + Virtual	https://events.linuxfoundation.org/
Open Compliance Summit	Dec. 7	Yokohama, Japan + Virtual	https://events.linuxfoundation.org/
SREcon22 Asia/Pacific	Dec. 7-9	Sydney, Australia	https://www.usenix.org/conference/srecon22apac
FOSDEM	Feb. 4-5	Brussels, Belgium	https://fosdem.org/2023/
DeveloperWeek	Feb. 15-23	San Francisco, California + Virtual	https://www.developerweek.com/
FAST'23	Feb. 20-23	Santa Clara, California	https://www.usenix.org/conference/fast23
SCaLE 20x	Mar. 9-12	Pasadena, California	https://www.socallinuxexpo.org/blog/scale-20x
CloudFest	Mar. 21-23	Europa-Park, Germany	https://www.cloudfest.com/
KubeCon + CloudNativeCon Europe 2023	Apr. 17-21	Amsterdam, Netherlands	https://events.linuxfoundation.org/

Images © Alex White, 123RF.com

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

Bernhard Bablok	62
Erik Bärwaldt	54
Zack Brown	12
Bruce Byfield	6, 34, 44
Joe Casad	3
John Cofield	92
Mark Crutch	71
Prof. Dr. Jörg Frochte	16
Jon "maddog" Hall	72
Vincent Mealing	71
Martin Mohr	68
Graham Morrison	86
Dr. Gerhard Paaß	22
Mike Schilli	48
Ferdinand Thommes	38, 80
Daniel Tibi	74
Garry Tuohy	26
Jack Wallen	8

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com
Copy Editors
Amy Pettle, Aubrey Vaughn
News Editors
Jack Wallen, Amber Ankorholz
Editor Emerita Nomadica
Rita L Sooby
Managing Editor
Lori White
Localization & Translation
Ian Travis
Layout
Dena Friesen, Lori White
Cover Design
Dena Friesen
Cover Image
© Kheng Ho Toh, 123RF.com
Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7679420
Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA
Publisher
Brian Osborn
Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the disc provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2022 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media USA, LLC, unless otherwise stated in writing.

Linux is a trademark of Linus Torvalds.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by Zeitfracht GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Published monthly as Linux Pro Magazine (Print ISSN: 1752-9050, Online ISSN: 2832-1413) for the USA and Canada and Linux Magazine (Print ISSN: 1471-5678, Online ISSN: 2833-3950) for Europe and other territories by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed.

Approximate
On Sale Date

UK / Europe	Nov 05
USA / Canada	Dec 02
Australia	Jan 02

Please note: On sale dates are approximate and may be delayed because of logistical issues.

Issue 265 / December 2022

Quantum Computing

Fact or science fiction? Real or just a fascinating mental exercise? The story of quantum computing is just beginning to unfold. Tune in next month for a closer look.

Preview Newsletter

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: ***https://bit.ly/Linux-Update***

Image © Pop Nukoonrat, 123RF.com

Register by October 14 for Big Discounts!

You. Dallas. SC22.

NOVEMBER 13–18

One of the world's largest gatherings of HPC researchers and professionals, SC22 is an incredible week of learning new skills, forging new friendships, solving complex problems, and viewing next-gen technology.

A new, custom-built Digital Experience is available if you prefer to attend virtually.

The International Conference for High Performance Computing, Networking, Storage, and Analysis

SC22
Dallas, TX | hpc accelerates.

Register Today!

sc22.supercomputing.org

Sponsored by: IEEE COMPUTER SOCIETY TCHPC acm sighpc

HETZNER

LOCATED IN THE USA

CLOUD SERVER

STARTING AT

$4.45

monthly | incl. IPv4

HETZNER CLOUD SERVER CPX11

- ✓ AMD EPYC™ 2nd Gen
- ✓ 2 vCPU
- ✓ 2 GB RAM
- ✓ 40 GB NVMe SSD
- ✓ 20 TB traffic inclusive
- ✓ Intuitive Cloud Console
- ✓ Located in Germany, Finland or USA

EPYC
AMD

HIGH QUALITY - UNBEATABLE PRICES

DEPLOY YOUR
HETZNER CLOUD
IN UNDER
10 SECONDS!

MANAGE YOUR CLOUD QUICK AND EASY WITH FEATURES LIKE
LOAD BALANCER, FIREWALLS, ONE CLICK APPS AND MANY MORE!

GET YOUR CLOUD NOW

CLOUD.HETZNER.COM

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.