FREE DVD

Openfire
Maximize privacy with your own IM server

THE FUTURE OF VECTOR GRAPHICS

LINUX PRO MAGAZINE
DECEMBER 2019

THE FUTURE OF VECTOR GRAPHICS

Shoebot
Create drawings and animations with Python scripts

Data Security in AWS

Yocto
Build a custom image for your Rasp Pi

One-Armed Bandit
Code a gambling game in Go

Oil Shell
Bash-compatible shell for easy scripting

Mediapurge
Sort and organize your media files

LINUXVOICE

- Launch Android apps on Linux with Anbox
- Chob: Keyword search for packages
- maddog: Keep learning new skills

FOSSPicks
- Zrythm digital workstation
- Drone control with Kirogi

Tutorials
- Foliate
- Pencil2D

Issue 229
Dec 2019
US$ 15.99
CAN$ 18.50

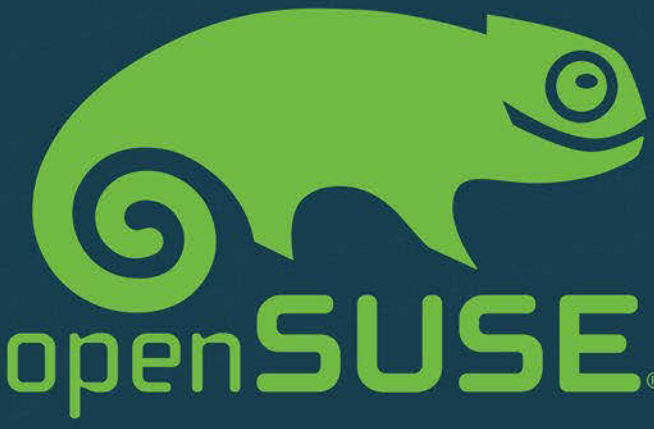
Tired of Waiting for New Software?

Tumbleweed

Get New Software Packages Quick & Easy

Work with New Code

Contributed Back to The Community



DOWN IN THE WELL

Dear Reader,

For the most popular social media platform in the world, Facebook certainly is taking a lot of heat. In 2016, they were an unwitting vehicle for manipulative ads from foreign powers, echoing and amplifying false narratives in the US presidential election, then later in Brexit and other elections around the globe.

In the 2020 election, Facebook is no longer unwitting, and they have reportedly snapped to action with a detailed ad policy. Have they put this urgent problem to rest? Doesn't look like it so far.

For the most part, I try to be apolitical in this space. People certainly try to guess my politics, based on my Linux cred, as well as my haircut, my dearth of business suits, and all the other factors people use to guess your politics. But actually, I don't think bullying and dishonesty (two of the problems previously associated with the Facebook platform) are political issues: They are human issues that everyone should care about.

On my side of the ocean, both sides of the political spectrum seem equally irritated with Facebook. One side accuses Facebook of a radical, West Coast, left-wing bias. The other side accuses Facebook of amoral pursuit of profits at the expense of integrity and civic duty. We were all wondering how this would shake out with the new election and Facebook's new policies. So far, it isn't off to a very good start.

In a letter to a candidate who objected to an allegedly false ad, Facebook has stated that they will not be using their third-party fact checkers to check political ads. According to the letter [1]:

> *…when a politician speaks or makes an ad, we do not send it to third party fact checkers.*
>
> *However, if a politician seeks to share a viral hoax – like a link to an article or a video or photo, that has been previously debunked, we will demote that content, display related information from fact checkers, and reject its inclusion in advertisements. That is different from a politician's own claim or statement – even if the substance of that claim has been debunked elsewhere. If the claim is made directly by a politician on their page, in an ad, or on their website, it is considered direct speech and ineligible for our third-party fact checking program.*

In other words, if you want to circulate a hoax, conspiracy theory, or other lie, just put it in your own words, and Facebook will help you do it. The tone of the company's correspondence continues to imply that their hands are tied by free speech issues, but no such constraint exists today or ever has in the past. Ad platforms have always made choices about what ads they would like to associate with their own reputation. In fact, the ad that provoked the controversy that resulted in Facebook's letter was actually rejected previously by a cable news network for its inaccuracy before Facebook accepted it and gave it over five million views [2].

In other content areas, Facebook reserves the right to reject ads for ethical reasons. For instance, if you traffic in payday loans or diet miracles, Facebook doesn't mind turning down your ad. If you use profanity, Facebook will send you to the door. But if you lie about another living person who is a candidate for office, you're in the clear.

I wouldn't want to be Facebook. I understand they are in a bit of a bind – since they are now accused of being a monopoly, any form of subjective criteria for evaluating political ads will look like they are imposing their will on the electorate. They could, of course, ban *all* political ads, as Josh Constine suggested recently at TechCrunch. That would certainly rescue them from the controversy, but it would cost them lots and lots and lots and lots of *money*, which seems to be what this whole thing is about anyway.

I remember learning in school about the old ladies who supposedly hid their bibles in the well when Thomas Jefferson was elected president in 1800. They were sure Jefferson was going to come take away their bibles, because that's what they were told by Jefferson's opponents. It all seemed so backward and strange at the time I learned it – a window into a bygone world where people could be so manipulated and misled.

It is weird to think that that world is coming back, and companies like Facebook are ready and willing to provide the forum.

Joe Casad,
Editor in Chief

Info

[1] Facebook Letter: *https://twitter.com/donie/status/1181780966806839297/photo/1?ref_src=twsrc%5Etfw%7Ctwcamp%5Etweetembed&ref_url=https%3A%2F%2Fwww.cnbc.com%2F2019%2F10%2F09%2Ffacebook-tells-biden-campaign-it-wont-remove-false-ads-from-politicans.html*

[2] "Facebook's Hands-Off Approach to Free Speech Gets Impeachment Test": *https://www.nytimes.com/2019/10/08/technology/facebook-trump-biden-ad.html*

[3] "Facebook Should Ban Campaign Ads. End the Lies": *https://techcrunch.com/2019/10/13/ban-facebook-campaign-ads/*

LINUX MAGAZINE

WHAT'S INSIDE

Vector graphics applications like Inkscape are a popular option for creating scalable graphics. Could these tools be even better? As is often the case, the science is out in front of the mainstream. We show you some innovations that could revolutionize tomorrow's graphics apps. Also in this month's issue:

- **Go Game Development** – Create a gambling machine using the versatile Go programming language (page 38).
- **Mediapurge** – Organize your media files and eliminate duplicates (page 46).

Turn to MakerSpace if you're wondering how to configure a multiboot Raspberry Pi with BerryBoot. And see our LinuxVoice section for an introduction to Anbox, a cool tool that lets you launch Android apps on Linux.

SERVICE

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

NEWS

08 News
- The Art of PostgreSQL
- Red Hat Announces CentOS Stream
- Linus Torvalds Agrees to Kernel Lockdown
- Richard Stallman Resigns from Free Software Foundation
- Oracle Announces Autonomous Linux
- Attackers Find a New Way to Install Cryptominers
- GitLab 12.3 Brings More Security to DevOps Engineers

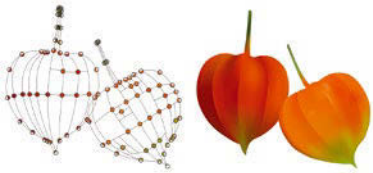
12 Kernel News
- String Handling Routines
- Speeding Up Database Workloads

COVER STORIES

16 Shoebot
A few lines of Python code are all it takes to create drawings, diagrams, illustrations, and animations with Shoebot.

20 Freeing Color
FreieFarbe is an association dedicated to promoting free standards for color graphics.

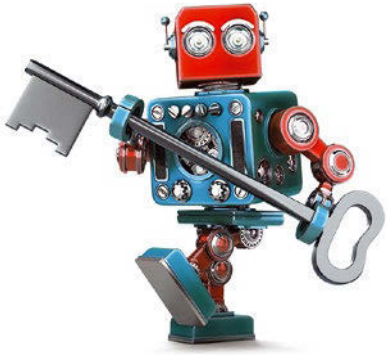
24 Next Steps in Vector Graphics
What are vector graphics and how could we make them better?



IN-DEPTH

28 Openfire
Openfire is an instant messaging and group chat server that lets users communicate with each other using the popular XMPP standard.

34 Data Security in AWS
As a cloud market leader, Amazon Web Services has had to put a great deal of thought into data security. Encryption options and key management play an important role.



IN-DEPTH

38 Go Game Development
A simple simulator for a Las Vegas-style gambling machine lets you practice your Go programming.

42 Command Line – AVR Microcontrollers
Learn about the tools needed to work with AVR microcontrollers, including the avrdude command and the Arduino IDE.

45 Charly – ntpd
Charly Kühnast, sys admin columnist for 15 years, is searching for lost microseconds.

46 Mediapurge
If you have a download folder full of photos and music, Mediapurge can help you sort files and even remove duplicates, but beware of its quirks.

50 Programming Snapshot – Fyne
With the Fyne framework, Go offers an easy-to-use graphical interface for all popular platforms. As a sample application, Mike uses an algorithm to draw arrows onto images.

54 Oil Shell
With its innovative scripting language, Oil, the Bash-compatible Oil shell aims to make life easier for script developers.

MAKERSPACE

56 BerryBoot
BerryBoot turns your Raspberry Pi into a multiboot system, with several operating systems on a single card.

60 Open Hardware – TMT9
This DIY, programmable input device is a compact companion to your keyboard, with nine keys and 16 layers that can be customized for different applications and games.

64 Yocto
Yocto is a tool for creating custom Linux images for embedded devices. We'll show you how to create a customized Linux for your Rasp Pi.

SEE PAGE 6 FOR DETAILS

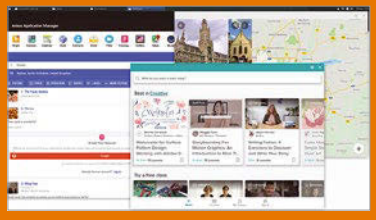
LINUXVOICE

69 Welcome
This month in Linux Voice.

70 Doghouse – Updating Technical Skills
As operating systems and computer languages evolve, programmers need to keep learning new skills.

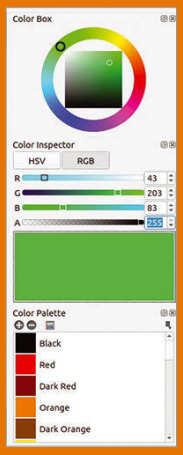
72 Chob
If you are looking for an application in the AppImage, Flatpak, or Snap app stores, Chob lets you perform a keyword-based search from the command line.

76 Anbox Android Apps on Linux
Thanks to Anbox and Snap, you can launch Android apps in a Linux virtual environment.



80 Foliate
You can customize the Foliate ebook reader to suit your needs, with bookmarking, translation, and read aloud features.

84 Pencil2D
Pencil2D, an easy-to-use painting and 2D animation program, lets you create small animations quickly. Despite the simple user interface, you might need a little help getting started.



90 FOSSPicks
This month, Graham explores Zrythm, Mumble 1.3, NoteKit, Kirogi, monolith, pastel, Nu Shell, PacVim, Stunt Car Racer Remake, and more!

On the DVD

CentOS 8

CentOS is a community-based distro based on source code from Red Hat Enterprise Linux. The latest edition comes with enhanced virtualization, as well as better support for LUKS2 disk encryption and an improved Image Builder tool. The nftables framework replaces iptables as the default packet filter, and the new release also includes the innovative extended Berkeley Packet Filter (eBPF) feature as a technology preview.

Arch Linux

Many seasoned Linux users consider Arch Linux the ultimate hacker distro. Arch starts with a minimalist "keep it simple" approach and lets users build the system around their own needs. The popular pacman package manager and other native tools combine to provide a unique Linux environment with a cult following among Linux developers and other power users.

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.

Additional Resources

[1] CentOS Project: *https://centos.org/*
[2] CentOS Wiki: *https://wiki.centos.org/*
[3] CentOS Documentation: *https://docs.centos.org/en-US/docs/*
[4] Arch Linux: *https://www.archlinux.org/*
[5] Arch Wiki: *https://wiki.archlinux.org/*
[6] Arch Forum: *https://bbs.archlinux.org/*

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

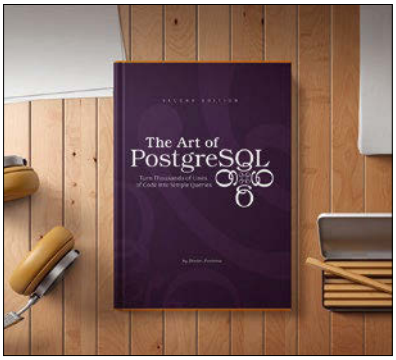
08
- The Art of PostgreSQL
- Red Hat Announces CentOS Stream

09
- Linus Torvalds Agrees to Kernel Lockdown
- Richard Stallman Resigns from Free Software Foundation
- Oracle Announces Autonomous Linux
- More Online

10
- Attackers Find a New Way to Install Cryptominers
- GitLab 12.3 Brings More Security to DevOps Engineers

The Art of PostgreSQL

The Art of PostgreSQL by Dimitri Fontaine is now available at *https://theartofpostgresql.com*. The book has several digital editions as well as a paperback edition.



As the new edition of the previously released *Mastering PostgreSQL in Application Development*, *The Art of PostgreSQL* updates content and comes with a new chapter about PostgreSQL extensions, such as hstore, pg_trgm, intarray, earthdistance, ip4r, and HyperLogLog, one of Craig Kerstiens' all time favorite extensions.

The Art of PostgreSQL will help you turn thousands of lines of code into simple queries.

Red Hat Announces CentOS Stream

Red Hat has announced a new Linux distribution called CentOS Stream for better synergy among RHEL (Red Hat Enterprise Linux), Fedora, and CentOS (*https://lists.centos.org/pipermail/centos-announce/2019-September/023449.html*).

CentOS is a clone of RHEL (minus Red Hat branding), which is compiled from the source code that Red Hat releases publicly. CentOS is funded by Red Hat but is a purely community driven project, though most lead developers of CentOS are employed by Red Hat.

CentOS Stream will sit somewhere between Fedora and RHEL to provide a place for developers who want to get their packages in RHEL. So far Fedora was used as a fast moving upstream project for RHEL. Red Hat forks code from Fedora to build the next version of RHEL. However, most enterprise-centric users were on CentOS and not Fedora, and there was not a direct path for those users to target RHEL, as CentOS was downstream of RHEL. With CentOS stream, developers can start playing with what to expect next in RHEL, and they can also submit patches.

"In practice, CentOS Stream will contain the code being developed for the next minor RHEL release. This development model will allow the community to discuss, suggest, and contribute features and fixes into RHEL more quickly," said Karanbir Singh, project leader of CentOS.

Linus Torvalds Agrees to Kernel Lockdown

Linus Torvalds has finally agreed to implement a lockdown feature for the Linux kernel (*https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=aefcf2f4b58155d27340ba5f9ddbe9513da8286d*). The feature was proposed several years ago but was rejected by Torvalds.

The upcoming release of Linux, version 5.4, will include this feature as a Linux Security Module (LSM). It will have two lockdown modes: "integrity" and "confidentiality."

Torvalds explained that, "If set to integrity, kernel features that allow userland to modify the running kernel are disabled. If set to confidentiality, kernel features that allow userland to extract confidential information from the kernel are also disabled."

According to ZDNet (*https://www.zdnet.com/article/linux-to-get-kernel-lockdown-feature/*), the new feature's primary function will be to strengthen the divide between userland processes and kernel code – even the root user will have limited access.

© rawpixel, 123RF.com

The feature will be disabled by default as it could lead to unexpected behaviors. Many Linux distributions, including Ubuntu and Red Hat have already implemented their own lockdown features using additional modules.

Richard Stallman Resigns from Free Software Foundation

The outspoken founder of Free Software Foundation (FSF) and the GNU project, Richard Stallman, has resigned from his post as President of the FSF. Stallman came under fire for his comments in defense of the late Marvin Minsky, co-founder of MIT's Artificial Intelligence lab, who was implicated in the Jeffrey Epstein sex trafficking scandal. Although Stallman did not defend Epstein, his comments regarding Minsky's involvement were regarded as insensitive at best and came across as an inappropriate logical exercise in the face of growing concern over Minsky's actions (*https://medium.com/@selamjie/remove-richard-stallman-fec6ec210794*).

Reaction from the Free Software community was swift. The Free Software Conservancy (SFC) said in a blog post (*https://sfconservancy.org/news/2019/sep/16/rms-does-not-speak-for-us/*), "The fight for diversity, equality, and inclusion is the fight for software freedom; our movement will only be successful if it includes everyone. With these as our values and goals, we are appalled at recent statements made by the President and founder of the Free Software Foundation, Richard Stallman, in his recent email to the MIT CSAIL mailing list."

The SCF then called for Stallman to step down from positions of leadership in the free software movement. A few hours later Stallman announced that he has resigned from his positions at MIT and FSF.

Oracle Announces Autonomous Linux

At the Oracle OpenWorld event, Larry Ellison announced Oracle Autonomous Linux, (*https://www.youtube.com/watch?v=1fbM6mP9Kl0&t=15s*) an addition to Oracle Linux that lifts the burden of managing and maintaining the operating system off the shoulders of operators and sysadmins.

Oracle Autonomous Linux can provision,

MORE ONLINE

Linux Magazine
www.linux-magazine.com

Linux Administration Focus
http://www.linux-magazine.com/tags/view/administration

Jump Box Security • Ken Hess
While Linux can be made very secure, you can increase the security of your entire network with jump boxes.

Rootkit Sleuth • Ken Hess
Linux can be infected by rootkit malware that is hidden and hard to detect. The chkrootkit program can help find rootkit infections.

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

High-Performance Python – GPUs
Jeff Layton
Python finally has interoperable tools for programming the GPU – with or without CUDA.

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

Build a Honeypot with Real-World Alerts
Joseph McManus
The honeyBot tools create a honeypot that activates an alert in the real world.

Get Started with OpenShift • Chris Binnie
We help you get started with OpenShift and OKD using Minishift.

Orion Software Development Environment
Bernhard Bablok
The open source Orion project is a browser-based tool integration platform for web development.

patch, tune, and scale the system automatically. Oracle is offering the feature for free to Oracle Linux customers.

In an exclusive interview to TFiR, Wim Coekaerts, Senior Vice President, Software Development at Oracle, explained that Autonomous Linux is built on top of Oracle Linux. The genesis of it is Oracle Autonomous database, which takes away mundane tasks from admins to keep the database secure and do upgrades for users. However, the OS running underneath the database was its Achilles heel, and an unpatched and poorly tuned system may have adverse effects on the database itself. So Oracle borrowed the ideas from an autonomous database and brought them to Oracle Linux.

"As a developer or sysadmin, you push a button and we provision the VM on bare-metal for you and we do the online patching so you don't have to worry about downtime," said Coekaerts.

Attackers Find a New Way to Install Cryptominers

This year in June, F5 researchers found a new malware campaign exploiting a Jenkins dynamic routing vulnerability to install a cryptominer (*https://www.f5.com/labs/articles/threat-intelligence/attackers-use-new–sophisticated-ways-to-install-cryptominers*).

F5 explained that the vulnerability bypasses specific access control lists and leverages the Groovy plugin metaprogramming to download and remotely execute a malicious cryptominer.

© Tomasz Pacyna, 123RF.com

The cryptominer consumes valuable computing resources, raising bills and leading to slower performance. In the case of enterprise applications, it could means hundreds and thousands of dollars in bills and lost revenues due to the performance hit.

F5 suggests the following steps to protect users: Implement web fraud protection to detect customers logging into your applications with infected clients designed to engage in fraud. Notify your clients of the malware you detected on their system while logging into your application (which can result in them being blocked from carrying out a transaction), so they can take steps to clean their systems; and provide security awareness training to employees and clients.

GitLab 12.3 Brings More Security to DevOps Engineers

With the release of version 12.3, GitLab has added a new security focused feature called Web Application Firewall for Kubernetes Ingress (*https://about.gitlab.com/blog/2019/09/22/gitlab-12-3-released/*).

"In GitLab 12.3 we are shipping our first iteration of a Web Application Firewall built into the GitLab SDLC platform. Its focus is on monitoring and reporting of security concerns related to your Kubernetes clusters," said GitLab in a press announcement.

The feature helps sysadmins determine if HTTP or HTTPS traffic contains malicious code.

GitLab plans to expand the scope of the feature by adding firewall rules to help reduce risk in earlier stages of application development.

According to the release, "Future releases will expand the WAF capabilities to block malicious traffic, create and manage firewall rules, and inform earlier stages of development to take action to further reduce risk."

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!

Subscribe now!

shop.linuxnewmedia.com/subs

GET IT NOW!
FAST DELIVERY WITH OUR PDF EDITION

Zack's Kernel News

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

By Zack Brown

String Handling Routines

In the midst of a discussion on patch handling in the kernel development process, it came up that apparently a lot of patches are scripted constructs that are hard to synchronize between the various maintainers. So, someone asked for an example of such a patch, and someone else offered a recent set of changes from Julia Lawall for the `stracpy()` code.

A wrapper around `strscpy()`, `stracpy()` is a replacement for `strncpy()`, which is a function for copying strings. It was at this point that Linus Torvalds, essentially off-topic, weighed in on the string handling issue:

"I'm not actually convinced about stracpy() and friends.

"It seems to be yet another badly thought out string interface, and there are now so many of them that no human being can keep track of them.

"The 'badly thought out' part is that it (like the original strlcpy garbage from BSD) thinks that there is only one size that matters – the destination.

"Yes, we fixed part of the 'source is also limited' with strscpy(). It didn't fix the problem with different size limits, but at least it fixed the fundamentally broken assumption that the source has no size limit at all.

"Honestly, I really really REALLY don't want yet another broken string handling function, when we still have a lot of the old strlcpy() stuff in the tree from previous broken garbage.

*"The fact is, when you copy strings, both the destination *AND* the source may have size limits. They may be the same. Or they may not be.*

*"This is particularly noticeable in the 'str*_pad()' versions. It's simply absolutely and purely wrong. I will note that we currently have not a single user or strscpy_pad() in the whole kernel outside of the testing code.*

*"And yes, we actually *do* have real and present cases of 'source and destination have different sizes'. They aren't common, but they do exist.*

So I'm putting my foot down on yet another broken string copy interface from people who do not understand this fundamental issue."

Joe Perches replied, "I think you are mistaken about the stracpy limits as the only limit is not the source size but the dest. Why should the source be size limited?" Joe asked if Linus had actually looked at the `stracpy()` code. How could it be a problem, since it was only a wrapper around another function?

Linus replied:

"Yes, Joe, I have.

"What part of 'there are now so many of them that no human being can keep track of them' didn't you see as a problem?

"How many broken string functions are we going to do, adding yet another one when you notice that the _last_ one wasn't great?

"We never seem to remove the broken ones. We just add yet another one, and have a never-ending jumble of random letters."

And in response to Joe indicating that he didn't feel the source size needed to be limited, Linus replied with some detailed analysis, including samples of broken code he found in the current kernel tree:

*"You just proved my point. You don't understand that sources can also be limited, and the limit on a source can be *smaller* than the limit of a destination.*

*"Did we learn *NOTHING* from the complete and utter disaster that was strlcpy()?*

"Do you not understand why strlcpy() was unacceptably bad, and why the people who converted strncpy() to it introduced real bugs?

"The fact is, it's not just the destination that has a size limit. The source often has one too.

"And no, the source is not always guaranteed to be NUL-terminated, nor is the source buffer guaranteed to be larger than the destination buffer.

*"Now, if you *know* that the source is smaller than the destination size, you can do:*

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**.

```
len = strnlen(src, srclen);
memcpy(dst, len);
dst[len] = 0;
```

"and that's not wrong, but that works only when
(a) you actually do the above
(b) you have no data races on src (or you at least only require that 'dst' is NUL-terminated, not that 'len' is necessarily the correct length of the result
(c) you actually know as the programmer that yes, the source is definitely smaller than the destination.
"and honestly, people don't get _any_ of that right.

*"For one thing, the buffer sizes of the source and destination may be two different things and some #define. It's hard to tell that one is always smaller than the other (or that they are always the same size). So then to get it right in the *general* case, you may need to do something like*

```
if (srcsize < dstsize) {
  .. do the above ..
} else {
  strlcpy(dst,src,dstsize);
}
```

"do you see the reason?

"Do you see why strlcpy() is only safe to do when the allocation size of the source buffer is at least as big as the allocation size of the destination buffer?

"For example, I just grepped for some patterns, and I can find code like this in the kernel

```
name_len = strnlen(fileName, PATH_MAX);
name_len++;  /* trailing null */
strncpy(pSMB->fileName, fileName,
  name_len);
```

"where pretty much everything is wrong. The comment is fundamentally wrong, and even spells "nul" wrong. Christ.

"Here's another one:

```
/* will be less than a page size */
len = strnlen(link,
  ocfs2_fast_symlink_chars(
  inode->i_sb));
kaddr = kmap_atomic(page);
memcpy(kaddr, link, len + 1);
```

"and notice how this time at least the comment is (hopefully) correct. But the code is wrong once again, because it doesn't actually do the correct pattern I showed above, it does a "memcpy(len + 1)" instead. Bzzt. WRONG!

*"What I think the code *wants* to do is*

```
kaddr = kmap_atomic(page);
copy_string(
  // destination and destination
  // size limit
  kaddr, PAGE_SIZE,
  // source and source size limit
  link, ocfs2_fast_symlink_chars(
    inode->i_sb)
);
```

"ie the destination has one size, and the source has another size, and the source may or may not be NUL-terminated.

"And then the programmer wouldn't have needed the comment, and wouldn't have needed to make sure that yes, ocfs2_fast_symlink_chars() is guaranteed to be less than PAGE_SIZE.

"Again, the code we actually _have_ in the kernel is not sensible. It doesn't actually nul-terminate the destination, despite clearly _trying_ to (note that "len + 1" in the memcpy).

"Now, it's possible that it doesn't care about properly nul-terminating things. And it's possible that the source is always nul-terminated to begin with unless the filesystem is corrupted. But the code clearly _tries_ to do something, and fails.

"Because copying a string is complicated, particularly when the allocations for source and destination may be entirely different.

"On a happier note, I actually found a correct code case too. Our 'kstrndup()' function seems to actually be at a first glance entirely bug-free, and actually takes a limited source string size, and gives you back a nul-terminated destination string of a different size. Of course, that's a simple case, because the size of the destination is something that that function actually controls, so getting it right is actually somewhat simpler."

The main discussion (about patch handling in the kernel development process) continued, but I wanted to bring up this side issue. It's clearly not an urgent thing, because Linus grepped around in the kernel tree and found lots of broken and buggy code, but he

didn't ask anyone to fix it. His concern is broader. He doesn't like the proliferation of special-case string handling routines that he feels should all be replaced by something clear, simple, and correct.

When Linus comes out with this sort of opinion, it's generally followed by various developers trying to give him what he requested. So probably in the next six months, we'll see an effort to make a proper string handling function and get rid of all the rest. I expect it'll look a lot like the struggle to create a good revision control system, except at the end he probably won't have to roll this one himself.

Speeding Up Database Workloads

Recently, Khalid Aziz from Oracle wanted to reclaim unused memory as fast as possible, to be available for other processes. In particular, he said, memory reclamation and compaction were normally triggered when the system was already running low on available RAM, which meant that running processes may have already begun to experience allocation delays.

Memory compaction is distinct from compression. In compression, you try to take up less space by recognizing repeating patterns in a given block of memory. In compaction, you just try to group allocated memory blocks together, in order to have larger available blocks of free space.

Ordinarily I'd expect this sort of issue to come from someone in the embedded systems industry, but Khalid is in the database universe, where computing resources are often tight as well. He pointed out that "In real life, forced direct reclamation has been seen to cause sudden spike in time it takes to populate a new database or an extraordinary unpredictable latency in launching a new server on cloud platform. These events create SLA [Service Level Agreement] violations which are expensive for businesses."

He wanted to implement a kernel feature to track memory trends over time, in order to predict when reclamation and compaction would become necessary, so that a running system could take steps to free up memory before it started to represent numbers of dollars lost.

The question was, how to actually capture memory usage data on a running system, such that the data collection itself didn't slow everything down. Essentially, even at the risk of lower accuracy, Khalid wanted to identify opportune moments to grab relevant data in order to graph free pages over time. He said, "Capturing these data points and computing trend line for pages of order 0-MAX_ORDER allows us to not only foresee free pages exhaustion point but also severe fragmentation points in future."

Khalid posted two patches to implement this. However, Michal Hocko objected, saying that this sort of analysis could be done just as easily from user space. Free memory pages could be tracked from anywhere, and he saw no need to load up the kernel with any of this logic.

Khalid replied, "the initial prototype to assess the feasibility of this approach was written in userspace for a very limited application. We wrote the initial prototype to monitor fragmentation and used /sys/devices/system/node/node*/compact to trigger compaction."

He went on, "The primary reason to implement this logic in the kernel is to make the kernel self-tuning. The more knobs we have externally, the more complex it becomes to tune the kernel externally. If we can make the kernel self-tuning, we can actually eliminate external knobs and simplify kernel admin. In spite of availability of tuning knobs and large number of tuning guides for databases and cloud platforms, allocation stalls is a routinely occurring problem on customer deployments. A best fit line algorithm shows immeasurable impact on system performance yet provides measurable improvement and room for further refinement."

But Michal pointed out that there were actually many approaches to measuring resource pressures. If Khalid's approach would be valuable in Oracle's particular use case, that didn't necessarily mean it would be beneficial for other users. Why have what seemed to be a specialized solution, in the official kernel tree?

Khalid didn't argue that particular point, but he said that in this case, the feature could really only be effective as an in-kernel mechanism. From inside the kernel, it would be possible for the kernel to tune itself very rapidly and deli-

cately in response to the changing environment. If done in user space, those same subtle responses would not be available. Delays in accessing the data would lower their value.

But he explained that his code was intended to be generally useful, not just for Oracle workloads. He said, "It is true different workloads will have different requirements and that is what I am attempting to address here. Instead of tweaking the knobs statically based upon one workload requirements, I am looking at the trend of memory consumption instead. A best fit line showing recent trend can be quite indicative of what the workload is doing in terms of memory."

Michal wasn't convinced. And he also pointed out that the kernel did already have self-tuning mechanisms for these things. Rather than implementing new self-tuning features, he said, "let's talk about what is missing in the existing tuning we already provide. I do agree that compaction needs some love but I am under impression that min_free_kbytes and watermark_*_factor should give a decent abstraction to control the background reclaim. If that is not the case then I am really interested on examples because I might be easily missing something there."

Khalid remarked, "Something so fundamental to kernel memory management as making free pages available when they are needed really should be taken care of in the kernel itself. Moving it to userspace just means the kernel is hobbled unless one installs and tunes a userspace package correctly."

In terms of what is missing in existing internal memory-tuning mechanisms, Khalid said, "last week an email crossed my mailbox where an order 4 allocation failed on a server that has 768 GB memory and had 355,000 free pages of order 2 and lower available at the time. That allocation failure brought down an important service and was a significant disruption."

His goal was to allow the kernel to address these situations itself, without requiring system administrators to tweak configurations for each new workload.

Michal still didn't like it. He said, "existing autotuning works mostly ok for a vast variety of workloads. A more clever tuning is possible and people are doing that already. Especially for cases when the machine is heavily overcommitted." And he pointed out that in order for Khalid's patches to be accepted into the source tree, they would have to first be tested on a wide variety of workloads, in order to prove they did not pose a risk to existing users. Short of that, Michal would remain skeptical. So, given the difficulties of actually doing that level of testing, Michal said, "I would really focus on discussing whether we have sufficient APIs to tune the kernel to do the right thing when needed. That requires to identify gaps in that area."

There was a little more back-and-forth, but the discussion essentially petered out at that point. Khalid was disappointed, because, in terms of testing different workloads, he said, "I have seen DB and cloud server workloads suffer under default behavior of reclaim/compaction. It manifests itself as prolonged delays in populating new database and in launching new cloud applications."

But he acknowledged, "It is fair to ask for the predictive algorithm to be proven before pulling something like this in kernel. I will implement this same algorithm in userspace and use existing knobs to tune kernel dynamically. Running that with large number of workloads will provide data on how often does this help."

There are a lot of areas of kernel development that developers find frustrating. Security is one, because you never know when some obscure aspect of the kernel will turn out to have a security implication for some new feature you've just spent weeks implementing. There's just no knowing until you post your patch and someone says, "wait, there's a problem."

It's similar here. Any time you try to adjust the way memory is handled, or how the kernel switches between running processes, or anything else that is meant to speed up a particular workload, you're going to run into the problem that your fix needs to work not just on your workload, but on everyone else's too – in spite of the fact that you don't know "everyone else" and have no way to test your patch on all the systems of the world.

But over time, strangely enough, these difficult features do come into being, and they tend to become cleaner, more secure, and more generalizable. ■■■

Create drawings and animations with Python scripts

Program It!

A few lines of Python code are all it takes to create drawings, diagrams, illustrations, and animations with Shoebot. *By Tim Schürmann*

Shoebot developers cryptically refer to their innovative tool as the "Python graphics robot" [1]. However, Shoebot's functionality is extremely simple: Using uncomplicated Python commands, programmers can scribble lines, rectangles, circles, and other geometric figures on a virtual canvas. The result is a vector graphic that Shoebot can paint to the screen and also save as an SVG, PDF, Postscript, or PNG file. After generating the graphics, the computer artist can even animate the graphics and change them in real time with the help of live variables.

Many Faces

By default, Shoebot starts an editor, in which you can type commands directly. Shoebot shows the resulting graphic at the touch of a button. The editor is particularly useful for quickly achieving initial results, starting experiments, and building prototypes. Shoebot is also useful as a teaching aid. In mathematics lessons, for example, Shoebot can easily visualize fractals.

If you are getting started with software development, Shoebot lets you learn how to program and use Python in a playful way. Artists can generate procedural or random graphics, and designers can generate visualizations or design concepts.

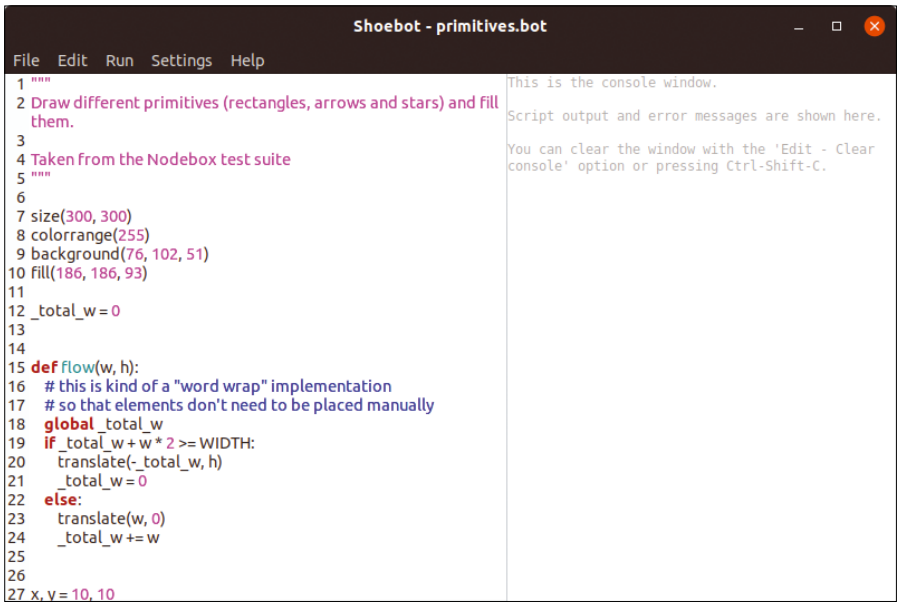
Alternatively, Shoebot works as a command-line program that reads and evaluates a script with Shoebot commands. You can also embed Shoebot commands in shell scripts to automatically generate vector graphics or drop data into a PDF diagram.

Python programmers who include Shoebot as a Python module can quickly design the prototype for their GUI. Shoebot can also be used as a plugin for tools that you can control via Python scripts – including Inkscape. But there is one downer: Shoebot still requires Python 2.7, and the support for Python 2.7 will soon expire.

Shoebot is closely oriented on NodeBox [2], which works in a similar way. The documentation even references the NodeBox docs at times. Other role models include Shoes [3] and DrawBot [4], from which Shoebot derives its name. The source code [5] is licensed under the GPLv3.

DIY Store

To run Shoebot, first install Python 2.7, the Python tools, and `pkg-config`. In addition, you need some libraries: GtkView v3 and Cairo, along with their developer packages.




```
"""
Draw different primitives (rectangles, arrows and stars) and fill them.

Taken from the Nodebox test suite
"""

size(300, 300)
colorrange(255)
background(76, 102, 51)
fill(186, 186, 93)

_total_w = 0


def flow(w, h):
    # this is kind of a "word wrap" implementation
    # so that elements don't need to be placed manually
    global _total_w
    if _total_w + w * 2 >= WIDTH:
        translate(-_total_w, h)
        _total_w = 0
    else:
        translate(w, 0)
        _total_w += w


x, y = 10, 10
```

Figure 1: The editor displaying an existing sample script from the `examples/basic` directory. The script draws several basic geometric shapes in one window.

On Ubuntu 19.04, the following command retrieves everything you need:

```
sudo apt install python python-setup-tools ➥
pkg-config libcairo2-dev libgtk-sourceview-3.0-dev
```

Then download the source code archive from the Shoebot website [1] and unpack it. The following commands in the Shoebot directory resolve further necessary dependencies and install the software globally on the system:

```
./install/install_dependencies.sh
sudo python setup.py install
```

The first command only works on systems with Red Hat, Fedora, Debian, Ubuntu, Linux Mint, or a flavor of SUSE. For other distributions, users need to manually install the packages for GCC and Gir RSVG, the developer package for Python 2.7, and the Pycairo, GObject, Cairo GObject, and Pillow Python modules. You also need the GTK + 3, Cairo, and libjpeg libraries, along with their developer packages.

If the call to `setup.py` fails, or if Shoebot does not launch, one of the many packages needed by Shoebot is probably missing. You'll see what could be a large number of error messages, with the culprits either right at the outset or close to the end.

Painting by Numbers

Calling `shoebot` launches the editor from Figure 1. *File* | *Open* loads an existing Shoebot script, of which several exist in the `examples` subdirectory. Shoebot assigns a separate window to each script. In the window, the source code appears on the left – the editor also offers syntax highlighting. *Run* | *Run Script* then starts the script. All text output ends up on the right side of the window; the drawing appears in a separate, new window. A message also appears in the event of a typo or other error (Figure 2).

An example of a simple Shoebot script is shown in Listing 1. Each command is assigned some required parameters in brackets. For example, a rectangle needs details of its position and size. Comments start with a hashtag (`#`).

Listing 1 first uses `size()` to set the size of the drawing area to 300x300 pixels; it then colors the `background()` a soft bright red. The intensity of the red, green, and blue color components normally needs to be stated as a floating-point value between `0` and `1.0`. For example, a call of `background(1.0, 0, 0)` would paint the background bright red. However, many developers are used to specifying values between 0 and 255. Listing 1 therefore switches `colorrange(255)` to this notation.

Translation

`rect()` then draws a rectangle. The first two values indicate the position of the upper left corner. By default, the drawing area numbers all pixels from top left to bottom right. The coordinates `0.0` are thus in the top left corner of the drawing area.

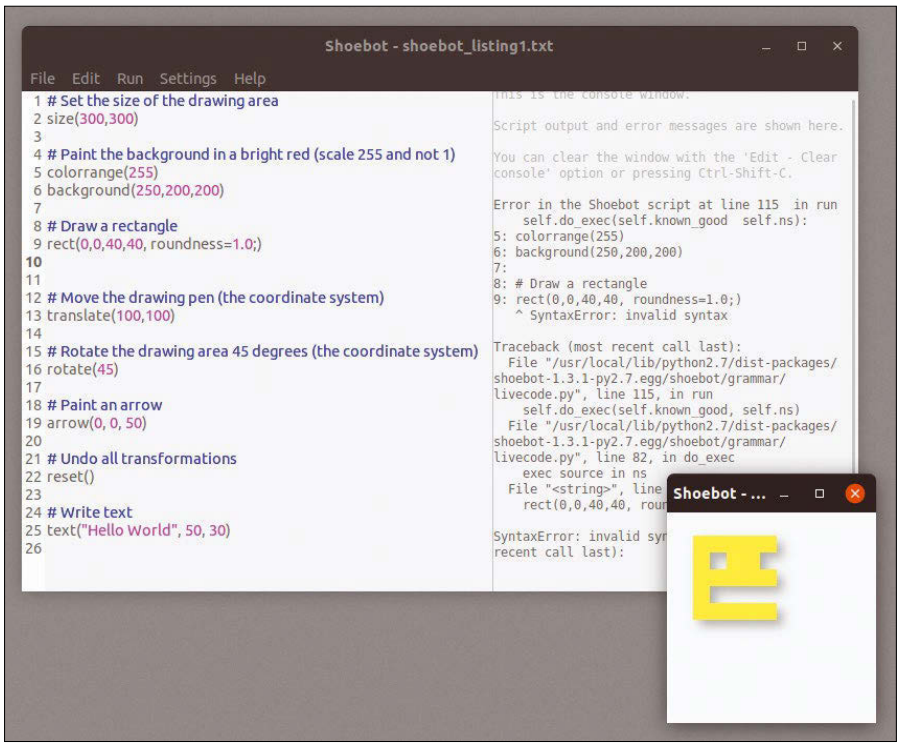


Figure 2: Text output is written on the right side of the main window, and the drawing appears in a new window.

Listing 1: Simple Drawing Example

```
01 # Set the size of the drawing area
02 size(300,300)
03
04 # Paint the background in a bright red (scale 255 and not 1)
05 colorrange(255)
06 background(250,200,200)
07
08 # Draw a rectangle
09 rect(0,0,40,40, roundness=1.0)
10
11 # Move the drawing pen (the coordinate system)
12 translate(100,100)
13
14 # Rotate the drawing area 45 degrees (the coordinate system)
15 rotate(45)
16
17 # Paint an arrow
18 arrow(0, 0, 50)
19
20 # Undo all transformations
21 reset()
22
23 # Write text
24 text("Hello World", 50, 30)
```

The rectangle from Listing 1 is also 40 pixels wide and 40 pixels tall. The value following the optional `roundness=` additionally determines the extent to which the script rounds the corners. The maximum value is `1.0`, which creates rounded corners, as shown in Figure 3.

`translate()` moves the coordinate system of the drawing area to the specified position; `rotate()` rotates it. In Listing 1, Shoebot first moves 100 pixels to the right, then 100 pixels down, then rotates the drawing area through 45 degrees, and at this position, paints an `arrow()` with a width of 50 pixels. Thanks to the rotation, the arrow points to the top right. The final `reset()` reverts all transformations; position `0,0` is therefore in the upper left corner once again. `text()` finally writes `Hello World` at the given position.

All Shoebot scripts are also Python programs. Developers can therefore use the complete language range of Python 2.7 and draw several circles on the screen in a loop. In this way, you can create different patterns quickly.

Librarian

In addition to the instructions from Listing 1, Shoebot only supports a few other primitive drawing commands [6], which essentially paint geometric figures on the screen. Thanks to additional libraries [7], Shoebot can make work easier for the user. Some Shoebot developers have simply adopted these libraries from NodeBox; others have written them themselves. For example, the commands from the *photobot* library manipulate photos, and *sbaudio* visualizes audio files. And developers can draw graphs with *graph*.

Listing 2 shows a simple example that creates a graph with three nodes and two edges (Figure 4), which `g.solve()` then automatically arranges to fit the canvas and `g.draw()` finally draws.

More Movement

You can also use Shoebot to create animations and process mouse and keyboard input. Developers will need some basic Python knowledge. Listing 3 shows an example of a simple animation that slowly draws a line from top left to bottom right. `speed()` defines the number of frames per second. Shoebot calls the newly defined `setup()` function automatically before it starts the animation. Whenever Shoebot needs to redraw the display, it calls `draw()`. In Listing 3, `draw()` draws a line with a length of one point.

The current status of the mouse and keyboard are assigned to special variables. For example, `MOUSEX` and `MOUSEY` are the current x- and y-position of the mouse pointer. `mousedown` is `True` while the user is pressing the mouse button; `key` contains the currently pressed key. The `text(key, 50, 30)` function would display the letter of the last key pressed.

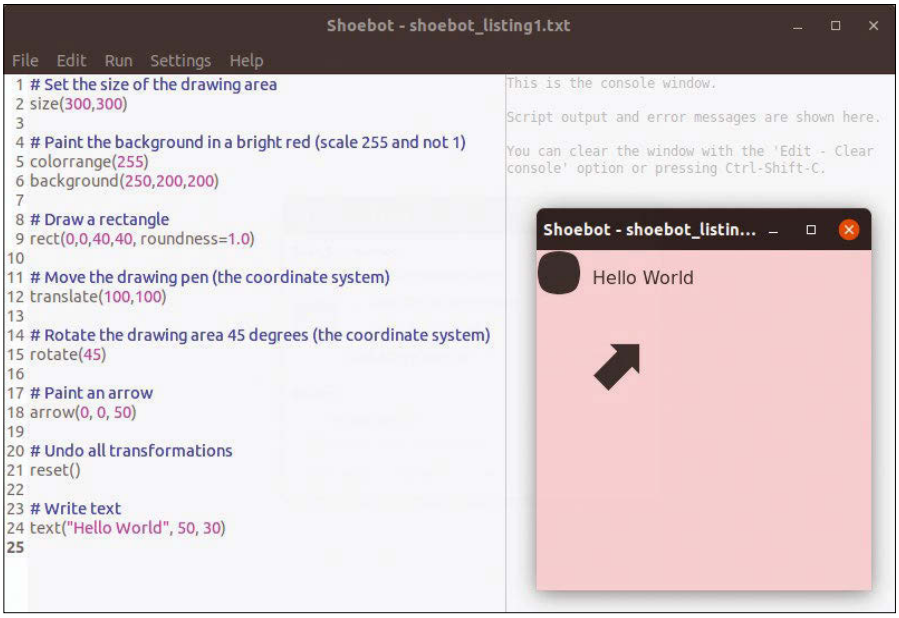


Figure 3: Listing 1 generates this result. The corners of the rectangle are rounded to the max.

Live Is Live

Like in the example from Listing 3, Shoebot users occasionally need variables whose contents they then feed to the

Listing 2: Graph Example

```
01 graph = ximport("graph")
02 map = graph.create(distance=0.6)
03 map.add_node("Duesseldorf")
04 map.add_node("Cologne")
05 map.add_node("Bonn")
06 map.add_edge("Duesseldorf", "Cologne")
07 map.add_edge("Cologne", "Bonn")
08 map.solve()
09 map.draw()
```

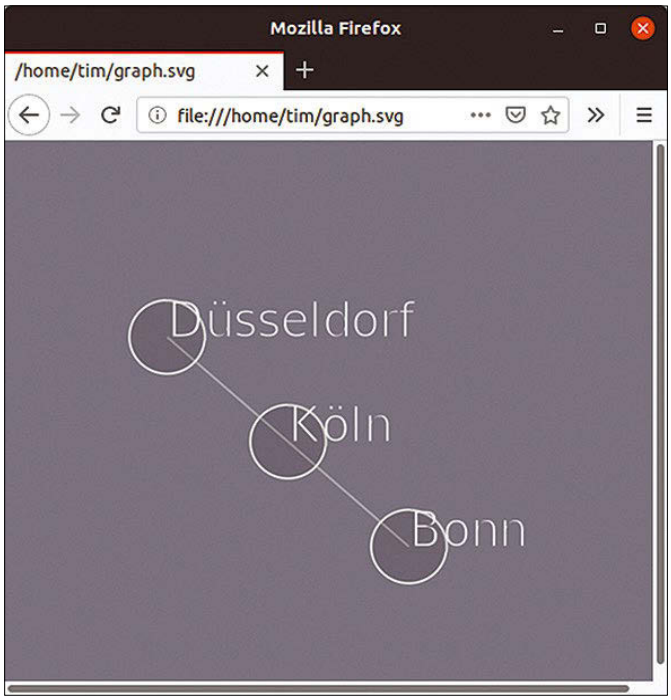


Figure 4: Section of the graph exported to SVG format showing three cities.

drawing functions. Shoebot supports live variables that developers can change while the Shoebot script is running. These variables are especially useful in animations. A live variable is defined with the following syntax:

```
var('position', NUMBER, 15., 0., 30.)
```

This example creates the new `position` live variable, which stores numbers (`NUMBER`). By default, the `position` variable contains a value of 15, but basically any number between 0 and 30 is allowed. If you enable the *Run* | *Show Variable* option in the Shoebot editor, another window appears with a slider that changes the value of `position` in real time (Figure 5).

At My Command

Even if the Shoebot scripts are Python programs, they are ultimately stored in files with a `.bot` extension. Scripts of this kind are processed by the command-line version of Shoebot. You only have to feed the script to the `sbot` tool:

Listing 3: Animation Example

```
01 size(400,400)
02 speed(25)
03 stroke(0.2)
04 strokewidth(1)
05
06 def setup():
07     global x,y
08     x=0
09     y=0
10
11 def draw():
12     global x,y
13     x = x + 1
14     y = y + 1
15     line(x,y, 1,1)
```

```
sbot primitives.bot
```

Shoebot draws the results in a new window. The optional `-f` parameter

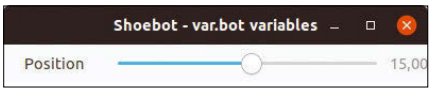


Figure 5: Use the slider to change the position variable in real time.

activates full-screen mode, which is enabled in the editor by selecting the *Run* | *Full screen* menu item. Alternatively, `sbot` can redirect the output to a file:

```
sbot primitives.bot -o primitives.png
```

In this case, a PNG file is created. `sbot` guesses the desired file by reference to the file extension. `.png`, `.svg`, `.pdf`, and `.ps` are currently supported.

Integration Helper

If you need to pass in the value of live variables in JSON notation, for example, use:

```
sbot --vars='{ "position": 21 }' script_name.bot
```

Developers can use Python code in Shoebot scripts and also include Shoebot as a Python module. This just takes two lines; the output ends up in the `test.png` file:

```
import shoebot
bot = shoebot.create_bot(outputfile="test.png")
```

The `bot` object now supports the well-known Shoebot commands as methods. For example, `bot.rect(10,10,40,40)` would paint a rectangle on the canvas. Finally, `bot.finish()` writes the results to the specified file.

Conclusions

Shoebot offers a very interesting approach. With just a few lines of Python code, developers can quickly create drawings and even animations. The instruction set, however, is essentially limited to a few simple geometric forms. The additional libraries only help slightly. The Turtle module from Python's standard library is thus a direct competitor.

Shoebot has its drawbacks. The documentation is still quite sparse, with several links that point to black holes. Additionally, the tool uses the legacy Python 2.7.

With Shoebot, even Python beginners can quickly achieve results. You can create animations quickly and easily change them thanks to the live variables. If you want to program uncomplicated graphics, take a look at Shoebot. ■■■

Info

[1] Shoebot: *https://shoebot.github.io/shoebot/*
[2] NodeBox: *https://www.nodebox.net/code/index.php/Home*
[3] Shoes: *http://shoesrb.com*
[4] DrawBot: *http://www.drawbot.com*
[5] Shoebot on GitHub: *https://github.com/shoebot/shoebot*
[6] Shoebot Command Reference: *https://shoebot.readthedocs.io/en/latest/commands.html*
[7] Shoebot Libraries: *https://shoebot.readthedocs.io/en/latest/libraries.html*

FreieFarbe and the quest for free color communication

Color my Freedom

FreieFarbe is an association dedicated to promoting free standards for color graphics. *By Ulrich Bantle*

Colors and color palettes existed for many years before the arrival of modern computers, and the graphics industry developed methods for handling color that seem out of date by today's standards. Different organizations and vendors often have their own color palettes. In some cases, the formula necessary for creating a color is guarded as a trade secret. Some colors are even trademarked. An organization called freieFarbe (Free Color) has been working to modernize color specifications. FreieFarbe advocates open and free color communication. Their goal is to promote the use of mathematically defined color models and ISO standards for color specification and selection.

FreieFarbe was founded in Oldenburg, Germany in 2016. The founding members are German and Swiss professionals who use color in their work. The association now has around 50 members. The association's ecosystem includes commercial providers, as well as free projects such as Scribus and Gimp.

The Problem

The freieFarbe website gives an example based on the RAL color palette, a standard used for specifying colors for varnish, powder coatings, and plastics in Europe. An architect or designer can specify a RAL color, such as RAL 6011, but the options for recreating this color graphically are actually quite limited. The RAL color palette is not available on the computer without specialized software. Even if you were able to recreate the hue on your computer

using computer-friendly RGB colors, you wouldn't be able to print it, because printer ink and printing devices don't map conveniently to the RAL color space, and the RGB spectrum itself is device-dependent, so it might not print the way it looks on your screen.

Professionals who work with colors often find themselves fighting their way out of a maze. Different color samples are used for printing than for paints, wall coverings, plastic films, and textiles. And the multitude of color systems are neither mutually compatible nor calculable. FreieFarbe believes that all colors should be specified mathematically using open formulas that are freely available to all users.

Holger Everding, the chair of freieFarbe e.V. [1], sees the association's mission as promoting open and free color communication. Many people who use colors for creative work view the fact that the color models in the commercial color collections are not based on mathematical formulas as one of the major problems. The models are not predictable and, in the opinion of the association members, this lack of definition is intentional – to avoid endangering the business models of these providers.

Color Marks

Trademark law imposes further restrictions on a free choice of colors. FreieFarbe lists 95 registered color marks in Germany alone. Examples include the well-known Telekom magenta and Milka purple. But organizations such as Nivea, UPS, Langenscheidt, and the Sparkasse chain of banks also protect their color marks. No competitor can use these colors for competing products.

For Holger Everding, this kind of commercial protectionism is heading in the wrong direction. "The world is colorful," Everding says, and colors are also there to express thoughts and feelings.

For the members of the association, this state of affairs is difficult to understand. "The only thing that really works with these commercial systems is their marketing," Everding adds.

FreieFarbe's aim is to point out alternatives to this mode of working. Their credo is: We see the future in freely available mathematical color models such as CIELAB or RGB. The advantages of free colors are huge, and the perspectives unbelievable. Holger Everding sees a real market niche in the development of predictable color models.

Calculated color systems such as RGB or CIELAB cannot be protected, and no one can prevent their propagation. It is also the association's aim to provide users with tips on how these systems can be deployed in all fields of application, and on how they can be leveraged to create colors that are fit for any purpose.

The organization relies on open standards that are integrated in modern computers. The computer is thus the ideal tool for working with color. FreieFarbe believes the computer can set colors free.

Scribus and Gimp

The association publishes templates and tools for popular open source programs. Gregory Pittman, a member of the Scribus development team, also wrote a script that can be used to create color value text tables and documents with color fields in Scribus [2]. The program then generates DIN A4 pages with 49 color patches each. You'll find the script on the amply-stocked freieFarbe download page [3].

Using the Scribus and Gimp programs, you can enter the color values directly, Everding explains. The Open Color Systems Collection (OCSC) 2.0, which is available for download for various programs, also contains color systems for installation in the free graphics programs Gimp, Calibra, Inkscape, and Krita.

The association does not offer its own program for calculating color models. Having their own tool is a desirable objective, according to Everding, but also one that the members have not yet been able to implement. The association does offer existing tools and aids under a Creative Commons license.

Open Color Communication

FreieFarbe obtained a research contract from the German Institute for Standardization (DIN) that led to DIN SPEC 16699 Open Color Communication [4]. A DIN SPEC is not a DIN standard, but in some cases, it can serve as the basis for a standard.

The DIN SPEC is currently available for download. The association has also published a reference implementation, the CIELAB HLC Color Atlas (Figure 1). The term HLC (Figure 2) consists of the letters H for Hue, L for Lightness, and C for Chroma.

The atlas includes a freely available PDF version that can be used for soft proofs on screen, for example. The core product, however, is a printed edition [5].

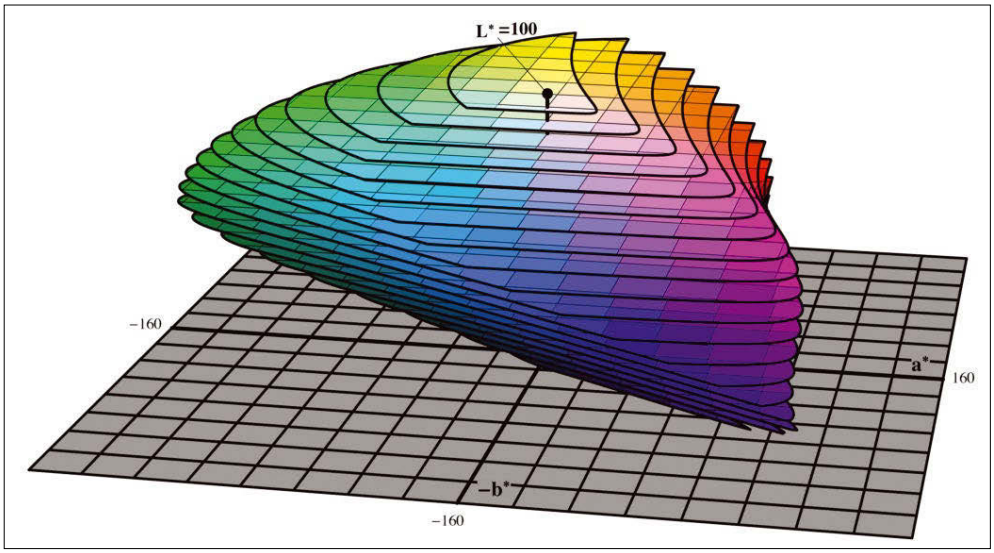


Figure 1: The CIELAB color space with additional brightness levels. © Professor Bernhard Hill, Aachen

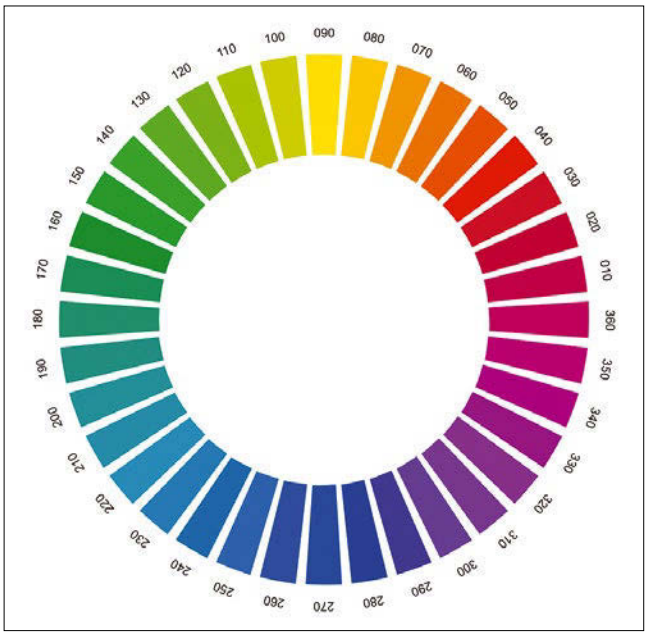


Figure 2: The HLC color circle in 10-degree segments.

The CIELAB color space presented by the Commission Internationale de l'Eclairage in 1976 [6] forms the basis for freieFarbe's atlas. The non-protectable and royalty-free CIELAB not only describes specific hues but also clearly defines all visible colors.

The CIELAB gamut includes the gamuts for both the RGB and CMYK color models, thus allowing for easy conversion, and because CIELAB can describe any visible color, it is a good neutral format for referencing colors to users of other color palettes. CIELAB is also device independent, which means the color you print is more likely to match the color that appears on your monitor.

Conclusion

FreieFarbe has only been around for three years, and the organization is just getting started with the international effort for open standards and better color communication. As they continue to refine their technical vision, more of their attention will turn to advocacy. It will no doubt take years to untangle the maze of incompatible industrial color palettes, but freieFarbe is laying the groundwork for a better solution that could serve as a blueprint for tomorrow's graphic designers. ■■■

Info

[1] FreieFarbe: *https://www.freiefarbe.de/en/*

[2] Scribus: *https://www.scribus.net/*

[3] Script for Scribus: *https://www.freiefarbe.de/en/thema-farbe/software/*

[4] DIN SPEC 16699: *https://www.beuth.de/en/technical-rule/din-spec-16699/295721446*

[5] CIELAB HLC Color Atlas: *https://www.freiefarbe.de/en/thema-farbe/hlc-colour-atlas/*

[6] CIELAB Color Space: *https://en.wikipedia.org/wiki/CIELAB_color_space*

Photorealistic images with vector graphics

Visual Math

What are vector graphics and how could we make them better? *By Pieter Barendrecht*

Have you ever wondered why you can zoom in on a piece of text in your web browser, PDF viewer, or word processor, and it still retains the smooth look it had at the original scale? In comparison, zooming in on a web image generally yields a pixelated or ragged-looking result. What gives text this resolution-independent property, and is it possible to extend this property to graphics in general?

Understanding Vector Graphics

A discussion of computer graphics should begin with the observation that there are two types of graphics: raster graphics and vector graphics. You're probably quite familiar with raster graphics — rectangular grids composed of individually colored pixels. Zooming in on a pixel-based raster image eventually results in either a blocky picture or a somewhat smoother result that often looks a bit blurred. This blurring or distortion is caused by an interpolation algorithm that generates new pixel data for the magnified version based on the values of neighboring pixels. On the other hand, vector graphics relies on mathematical descriptions of curves. These descriptions are resolution-independent, which means that vector graphics can scale without losing quality. The glyphs of a font that you look at on your screen are examples of vector graphics in daily life. If you're interested in taking a closer look at the curves defining these characters, install FontForge [1] (or alternatively, Birdfont [2]), and open a font you're interested in (commonly stored in `/usr/share/fonts`).

For a deeper look at vector graphics, it is very useful to install a dedicated vector graphics editor — the open source Inkscape is an excellent choice. Start by adding a circle, a (rounded) star, and a single character from your favorite font. Set the fill of all objects to

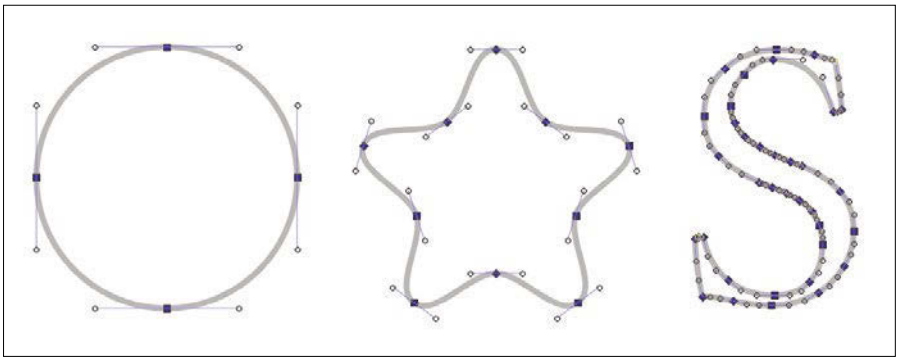
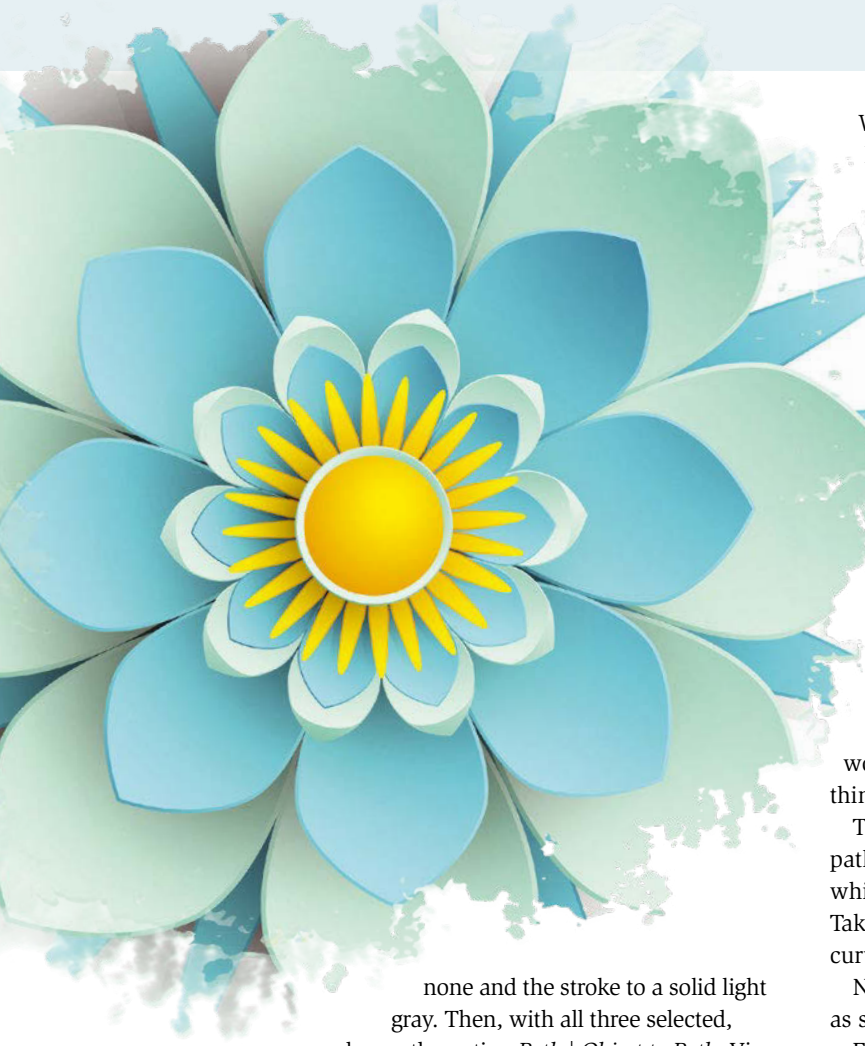


Figure 1: Nodes and handles define a vector graphics object.



none and the stroke to a solid light gray. Then, with all three selected, choose the option *Path* | *Object to Path*. Viewing the resulting paths in edit mode (F2) with everything selected should yield a result similar to the images in Figure 1.

The squares (referred to as nodes) and round handles are all it takes to define these composite curves. A single curve segment consists of two of each: a node at each endpoint of the curve segment and a handle connected to each node to define the tangent — that is, the direction — of the curve at those points. These curves are known as (cubic) Bézier curves, and they were discovered in the French car industry in the early sixties by Paul de Casteljau and Pierre Bézier, who at the time were working at Citroën and Renault, respectively. Bézier curves are the fundamental building block of vector graphics. (Press *B* in Inkscape and click/drag on the canvas to create a new Bézier curve.)

When a curve consists of multiple segments, nodes usually have handles on both sides. If a node and its two handles lie on a single line, the connection between the two segments is smooth (or *tangent-continuous* in mathematical language), which is often desired in a design. Most nodes in Figure 1 are smooth nodes, with the exception of some nodes defining the outline of the character S from the Linux Libertine font, which are sharp nodes.

Bézier curves facilitate the design of intricate 2D shapes. You can design these shapes manually or obtain them automatically — in Inkscape, a raster graphic can be traced using the option *Path* | *Trace Bitmap*.

The fill of paths can be set to a single color, though that probably won't result in something quite lifelike. A linear or radial gradient brings the result a bit closer to life but is still rather limiting.

For something better, consider a rectangular curve network, where each cell is defined by four Bézier curves. Something like the 2x2 curve network in Figure 2 will do for now.

The curves can be moved around freely, though the nodes at path intersections are connected to each other (something which currently cannot be done in Inkscape in edit mode). Take, for example, the central node, which is connected to four curve segments, or in other words, is a corner of four cells.

Next, assign a color to each node, which you could visualize as shown in Figure 3.

Finally, interpolate the assigned colors over the interior of the cells. You can interpolate in a variety of ways — using cubic interpolation (the same type used by the Bézier curves, but now in both directions) is a reasonable choice resulting in a smooth color propagation (Figure 4).

The overall concept is known as a gradient mesh, and it was added to Adobe Illustrator as a new vector graphics primitive in 1998 following the introduction of PostScript 3. Soon after, it also became available in CorelDRAW, and nowadays it is available in Inkscape as well. The gradient mesh feature facilitates the design of lifelike graphics that are fully scalable.

Refinements?

After using the gradient mesh tool for a while, you'll probably become aware of certain shortcomings. For one, if you want to

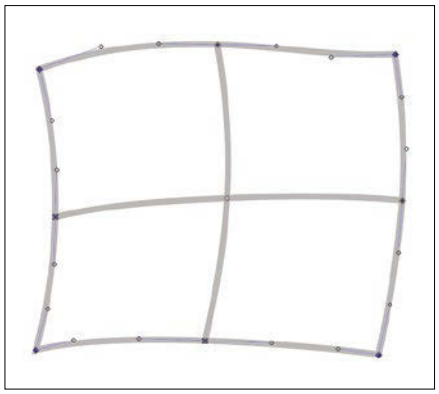


Figure 2: A 2x2 curve network.

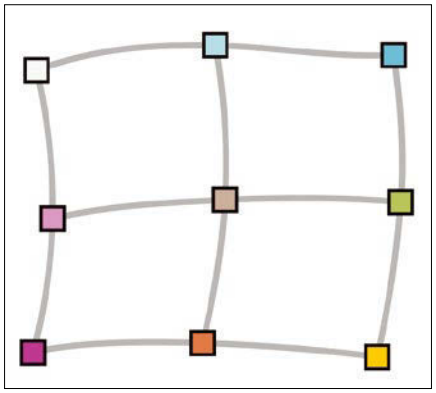


Figure 3: Assign a color to each node…

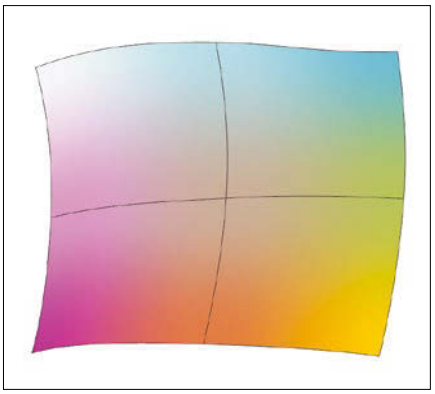


Figure 4: … and then interpolate the node colors to fill in the cells.

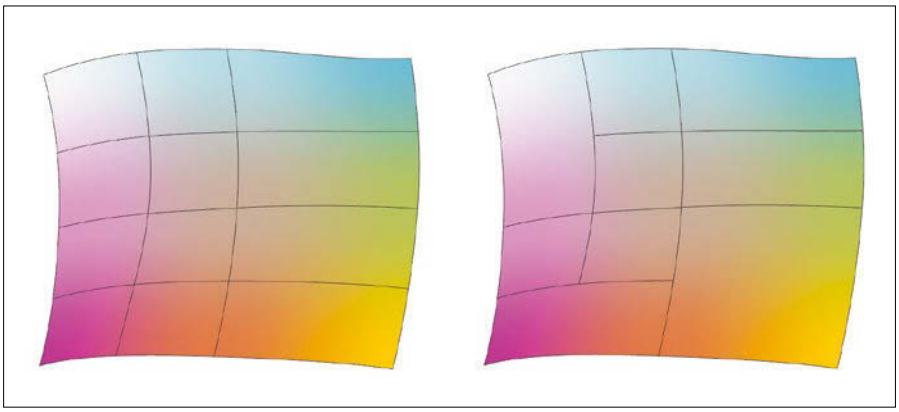


Figure 5: Global refinement (left) often adds superfluous cells. Local refinement (right) allows you to add cells only where desired.

add more detail to your shape, you'll likely need more cells. But because the gradient mesh is inherently stored as a rectangular curve network, you'll end up with an entire new row and/or column of additional cells. These unneeded cells can quickly clutter the workspace and slow down your machine. Instead, a local approach would be much better, though it requires a more sophisticated data structure. The example in Figure 5 shows the difference between a global (standard/traditional) and a local (improved) refinement of a gradient mesh.

Secondly, the network's rectangular nature itself is rather limiting — modeling an arbitrary object as a deformed curved rectangle is not exactly intuitive. Although several gradient meshes can be placed on top of each other to partially remedy this, this approach is not very user-friendly. Instead, letting go of the rectangular nature of the curve network altogether provides much more freedom. Although maintaining a smooth color propagation in such a flexible setting poses a mathematical challenge, it is certainly possible.

Another obstacle is the assignment of sharp transitions in color, which requires a minor modification of the data structure. The examples in Figure 6, 7, 8 and 9 showcase the three improvements I just described.

As of the time of this article, vector graphics editors only support the traditional version of the gradient mesh (the examples in Figures 6-8 were generated using my own software, which was developed as part of my PhD). We hope that graphics developers will add these advanced features to the leading open source vector graphics editors in the near future. Nevertheless, digital artists have created some quite sophisticated digital artwork using gradient meshes — a quick search on DeviantArt [3] shows how realistic you can get with common vector graphics tools.

Other Improvements

Just as shapes composed of curves can be traced automatically, raster graphics can be automatically vectorized into gradient meshes by external tools. This fact makes it possible to create resolution-independent versions of existing art, which can then be displayed and printed at any size without losing quality. An added benefit is that a vector representation can be much smaller than a raster representation of a medium to large image — in other words, vectorization could also be used from the perspective of file compression.

In addition to the gradient mesh, there is a second primitive known as diffusion curves that aims for the same goal — scalable photorealistic images. The necessary steps are as follows. First, the outline and inner detail of a shape are designed using Bézier curves. Then, a color

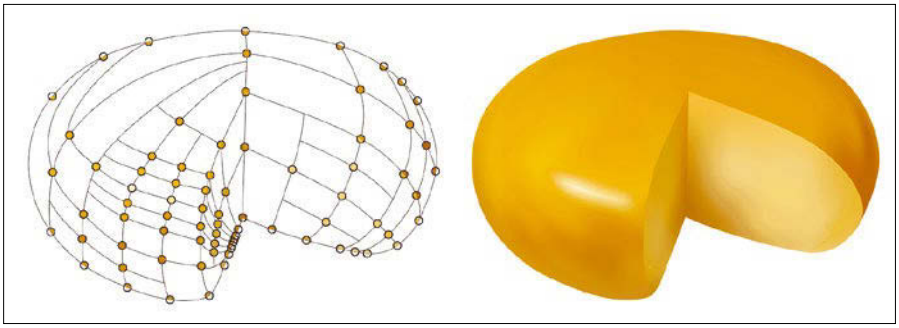


Figure 6: A wheel of Gouda cheese modeled as a gradient mesh supporting local refinement and sharp color transitions.

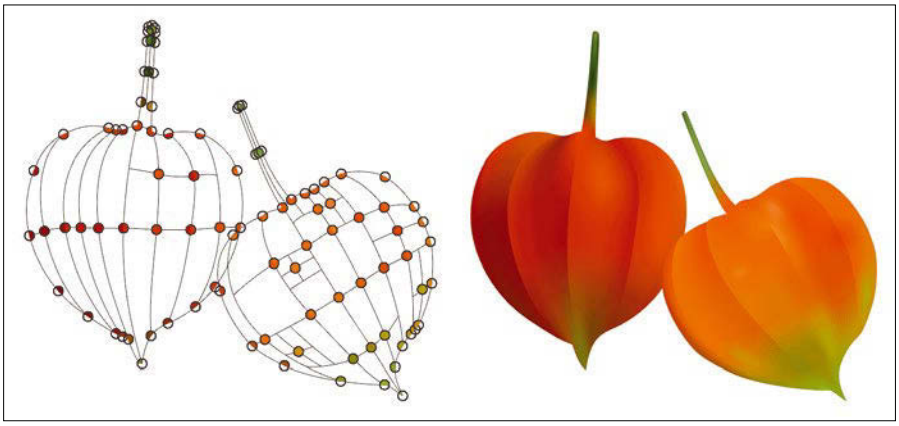


Figure 7: Husks of the Chinese lantern plant (Physalis alkekengi) modeled using all three improvements.

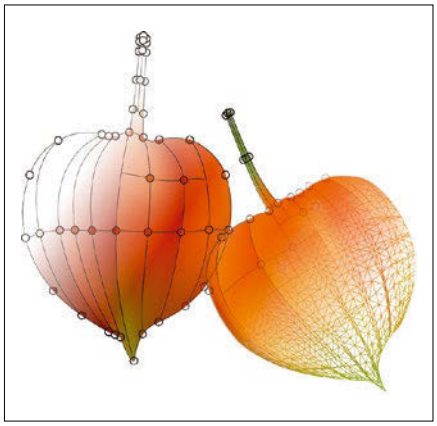


Figure 8: Different visual stages of an improved gradient mesh.

gradient can be assigned to each curve. Finally, these assigned color profiles are then diffused throughout the interior, producing smooth results. In addition to curves, individual color points can be added that also diffuse their color around them. This is evidently much closer to the workflow of an artist, and therefore more intuitive to use than gradient meshes. Although the concept has been around for over a decade, it got introduced as a primitive in Adobe Illustrator only recently under the name freeform gradient — you will find some demonstrations and how-to on YouTube.

Conclusion

Is it truly possible to extend the resolution-independent property of text to general images? Well, that depends on the image. Everything can be vectorized, but an image with a lot of local detail might currently still result in a gradient mesh composed of a large number of cells. Vectorization through diffusion curves is also becoming possible, although this technique suffers from similar issues. In academia, it is a topic of active research.

Ultimately, scalable photorealistic images will only catch on if the concept is supported through open standards such as the Scalable Vector Graphics (SVG) standard [4]. A proposal to add gradient meshes to a future version of SVG has been around for a while, but this concept largely depends on support from web browsers. Surprisingly, there seems to be very little interest among the major browser vendors in incorporating something so potentially revolutionary. The need to raise awareness about this promising technology is one of my reasons for writing this article. I hope the amazing artwork created with the current generation of vector graphics tools will bring wider support for these advanced techniques.

For a more technical description of the techniques described in this article, see the article I co-wrote with Martijn Luinstra, Jonathan Hogervorst, and Jiří Kosinka for the journal *The Visual Computer*, which is available online [5]. ■■■

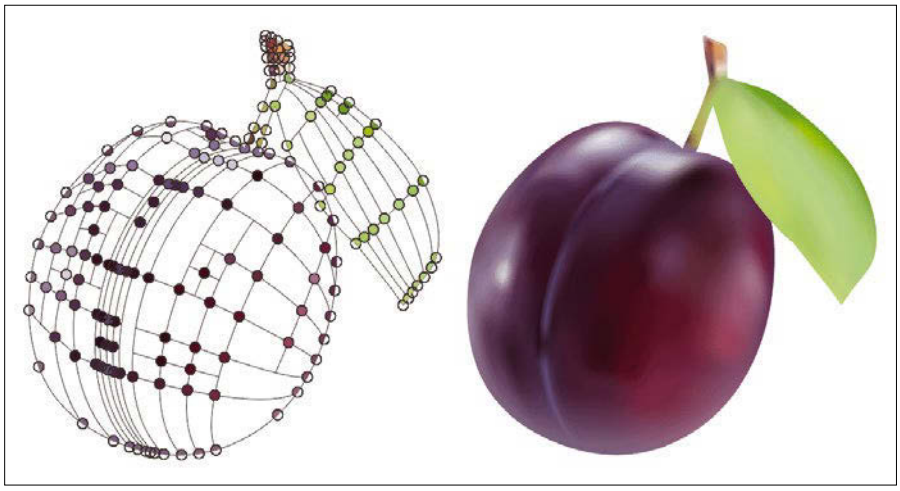


Figure 9: A plum modeled using all three improvements. Local refinement is particularly effective for modeling specular highlights.

Info

[1] FontForge: *https://fontforge.github.io/*

[2] Birdfont: *https://birdfont.org/*

[3] DeviantArt: *https://www.deviantart.com/*

[4] Scalable Vector Graphics Format: *https://en.wikipedia.org/wiki/Scalable_Vector_Graphics*

[5] "Locally Refinable Gradient Meshes Supporting Branching and Sharp Colour Transitions" by Pieter J. Barendrecht, Martijn Luinstra, Jonathan Hogervorst, and Jiří Kosinka, *The Visual Computer*, issue 6-8, vol. 34, June 2018, pp. 949-960, *https://link.springer.com/article/10.1007/s00371-018-1547-1*

Deploy a real-time collaboration server

Small Talk

Openfire is an instant messaging and group chat server that lets users communicate via the popular XMPP standard. *By Mayank Sharma*

Used responsibly, instant messaging (IM) offers the benefit of instant communication and collaboration on the corporate intranet. However, many companies, fearing IM's adverse affect on productivity, tweak their corporate firewalls to block all ports ferrying IM traffic. A better approach is to control the IM server by bringing it in-house. The Java-based, cross-platform Openfire server [1] makes it easy to host your own IM server.

The Extensible Messaging and Presence Protocol (XMPP) is one of the most popular protocols for powering an IM server. There are several XMPP-based IM servers available, but Openfire is one of the easiest to deploy and manage. Openfire implements many of the XMPP protocol's commonly used features [2] and scales well.

The Openfire server can be deployed on Windows, Mac OS X, and on various Linux distros, such as Debian, CentOS, Ubuntu, and Fedora [3]. To get started with Openfire, first make sure your server has the Java Runtime Environment (JRE) installed. If you are deploying Openfire on an RPM-based system such as CentOS, you're all set since the Openfire RPM package bundles the JRE.

On a DEB-based system such as Debian or Ubuntu, you need to install the JRE from the distro's repository with:

```
sudo apt install openjdk-8-jdk
```

When it's done, you can install the server with:

```
sudo dpkg -i openfire_4.4.2_all.deb
```

After it's installed, you will find Openfire under the `/opt/openfire` directory. You can control Openfire just like any other service with:

```
systemctl {start | stop
| restart | status} openfire
```

The first time you run Openfire, you need to configure it by pointing your web browser to port 9090 on your server. For example, if the Openfire server is on a machine with the IP address *192.168.0.14*, head to *http://192.168.0.14:9090* to configure the server. A five-step configuration wizard will ask you basic questions, such as the database details for storing user information and a directory server for fetching authentication information.

After selecting the language in the first step, the configuration wizard will ask you to tweak basic server settings, such as the domain name (Figure 1). You should also change the default ports for accessing the administration console for security purposes. Furthermore, you should also encrypt the server's system properties by selecting one of the two encryption algorithms offered (Blowfish and AES) and specifying a key.

In the next step, you're asked to select a database (Figure 2) for storing information, such as user profiles and offline messages. If you are deploying Openfire on a small network, you can choose to use the built-in HSQLDB database, which doesn't require you to set up an external database server. However, if you'll be servicing hundreds of users concurrently, you should select the option to connect to an external database, such as MySQL.

Similarly, in the following step, you are asked to select a mechanism for fetching user authentication information. Unless you have a directory server already managing users, you should use the default option, which entrusts user management to Openfire. In the last step, you are asked for the administrator's email address along with the password for accessing the admin interface on subsequent visits. Your Openfire server is now ready to accept connections and facilitate communications inside your network.

Until you have gained experience with Openfire, you should use its built-in HSQLDB database instead of an external one. Similarly, when deploying Openfire in a test environment, it's more convenient to entrust user management to Openfire instead of your directory server.

Using the Server

Your server is now up and running. You can connect to it via any IM client that supports the XMPP protocol, such as

Author
Mayank Sharma is a technology writer. You can read his scribblings in various geeky magazines on both sides of the pond.

Pidgin, Gajim, Kopete, or Psi. For best results, I'll use the Spark IM client, which is designed by the Openfire developers specifically to be used with their IM server. Like the server, the Spark client also is written in Java. While its RPM version includes the JRE, the DEB version does not, so make sure you pull in the JRE (if you haven't already) before installing Spark.

Once installed, all you need to connect to the Openfire server is the server's location and your authentication information. Remember, however, that the new Spark versions are based on the latest Smack library, which doesn't allow you to connect if you are using IP addresses instead of domain names. You can temporarily work around this problem by disabling hostname verification in Spark [4].

The first order of business is to add some users, since your server currently doesn't have any users besides the default admin user. You can add users either via the web-based administration console or from within the client. If you prefer the first route, log into the server's administration console by heading to *http://192.168.0.14:9090*. This time however, instead of the configuration wizard, you'll be greeted by a login page. Enter the username (admin) and the password you specified in the last step of the configuration wizard to access Openfire's admin interface (Figure 3).

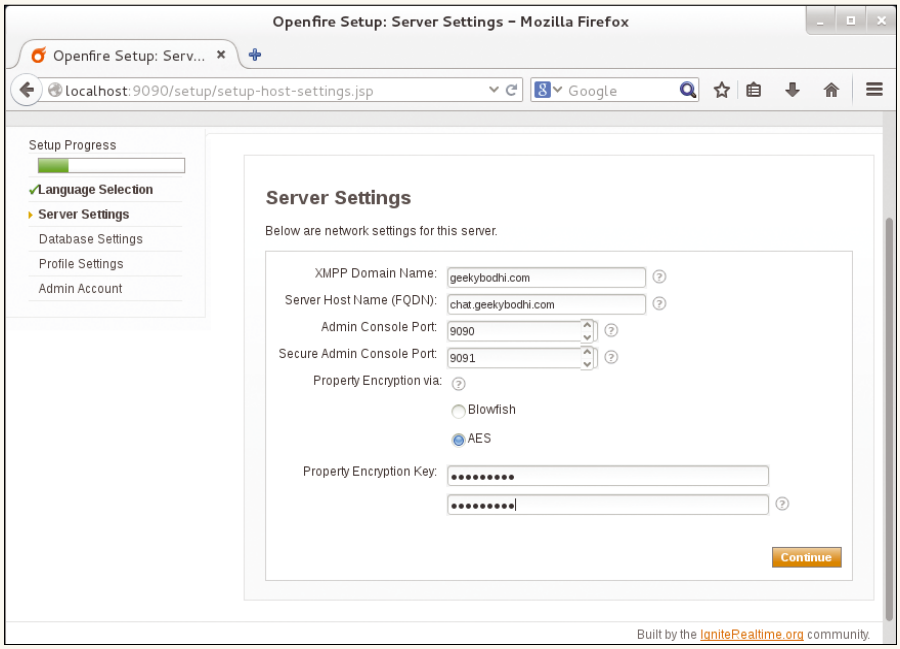


Figure 1: Click on the *?* icon for help with the individual fields.

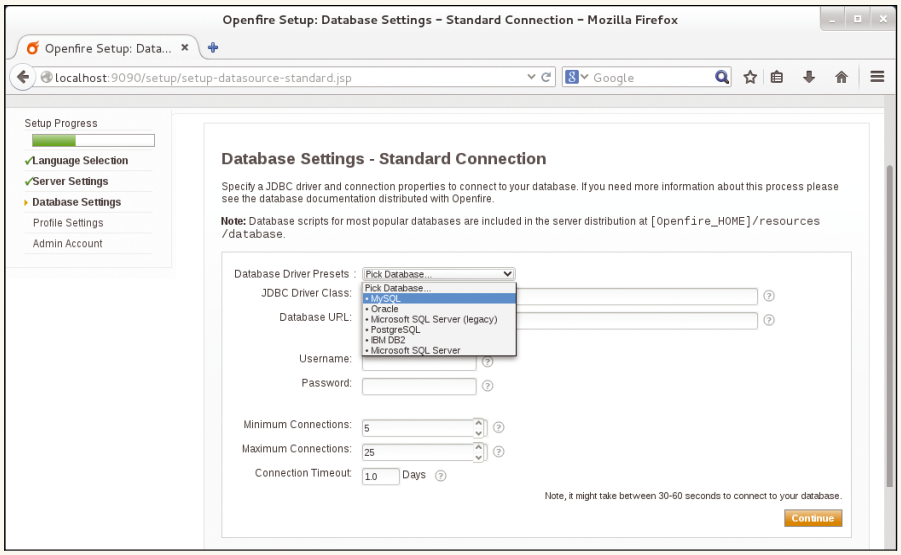


Figure 2: Depending on your network and use, you can also control the number of connections to the database.

To add a user, head to the *Users/Group* tab in the admin console, click on *Create New User*, and fill out the user's details. Users can also register themselves with the Openfire server from their clients. Click the *Accounts* button on the Spark login page and simply fill out the login details in the pop-up window.

Explore the Console

Openfire has a simple and straightforward administration console. It's divided into several tabs with each tab housing multiple configuration options. From within these, you can tweak every aspect of the server and integrate it with existing network services. The other aspect of running a server is being able to monitor its activity. The interface is also designed to provide you with visual feedback at a glance to help you keep tabs on the server and gauge its performance.

When you log into the server, you're presented with basic details about the server including which Openfire version you are running, the platform you are running it on, how long it has been running, the Java version powering the server, how much memory it is consuming, and so on.

On this page, you can also see which ports Openfire uses and for what purpose. Like other aspects of the server, these ports are also configurable. For instance, you can change the default secured (9091) and unsecured (9090) ports for accessing the server administration console by clicking the *Edit Properties* button in the Server Ports section.

You can also choose to change your time zone from a drop-down menu under *Language and Time*. Don't be too adventurous with this option, as the logs are time stamped based on the time zone you've selected here. If you select a time zone that's different from your location, you'll have trouble understanding the logs when troubleshooting a problem.

Speaking of logs, the administration console has a log viewer that you can use to view the server logs. Under the *Logs* option, you can get a snapshot of the Error, Warn, Info, and Debug logs. All of these logs, except Debug, are enabled by default. Enable the Debug log if the server is misbehaving (e.g., after you alter a system property or add a new component).

The Openfire server manages several tasks and resources. For all the components it has to maintain, the server keeps

track of their maximum permissible size, their current size, usage as a percentage, and the resource's effectiveness. The effectiveness, as explained in the interface itself, measures how well your cache is working. If the effectiveness is low, this usually means that the cache is too small. Openfire flags such caches. You can view these caches under *Cache Summary* (Figure 4). To clear any individual cache, select the checkbox in the rightmost column for the cache you want to empty, then scroll down, and click the *Clear Selected* button.

Plugins

You can extend Openfire and infuse new functionality by adding plugins. The default Openfire installation ships with over 20 plugins that have been thoroughly tested. There are plugins that add useful features such as the ability to broadcast messages to all users and to filter messages based on their content. There also are plugins that help hook up Openfire to existing services on the network, such as the Asterisk private branch exchange or an email server.

To enable a plugin, head to the *Plugins* tab and click on *Available Plugins* to see a brief description of each plugin, as well as buttons to view each plugin's readme file and changelog (Figure 5). To install a plugin, click on the green + button next to it. Some plugins worth exploring are Broadcast, which sends messages to all users at once; the Email listener, which connects to your mail server and alerts users of incoming emails; and the Packet filter, which keeps conversations polite.

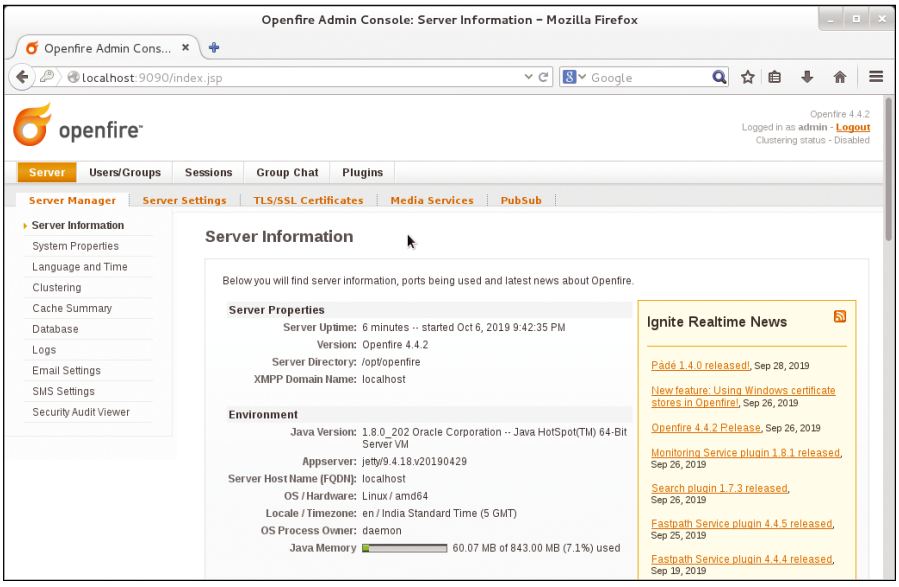


Figure 3: Openfire's dashboard has a tabbed interface that's logically arranged and intuitive to navigate.

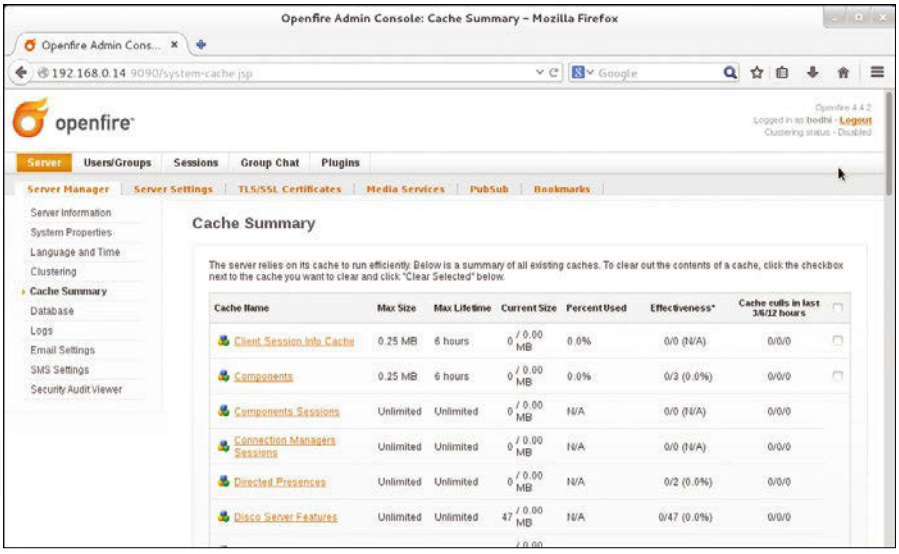


Figure 4: You can click on any cache to view its contents.

Share Rosters

Although Openfire has tons of useful features, I find roster sharing to be particularly useful for my deployments. Roster sharing allows you to populate your users' rosters in advance. In XMPP parlance, a roster is a friends list. With this feature, all your users can have their coworkers in their contact list as soon as they log in. This will work as long as you have a categorized list of groups and users within these groups, irrespective of whether this information comes from a directory server or has been created manually using the admin interface.

To allow members of each group to see each other, go to the *Users/Groups* tab in the admin interface. Switch to the *Groups* tab and click on any one of the listed groups. Scroll down to the Contact List (Roster) Sharing section and toggle the *Enable contact list group sharing* option and enter the name of the group in the text box (Figure 6).

By default, members of a group are added in the contact list of other members of the same group. However, besides members of their own groups, certain users should be in everyone's roster, such as the IT department. For this, toggle the *All users* radio button under the *Share group with* sub-setting, which will give all users in every department access to users from IT in their contact list. Save the settings, and repeat the procedure for all the groups on your server.

Ready for Rollout

After you've tested Openfire and tried it in a pilot project, you'll want to put it into production. When you deploy the Openfire server on your production network, it's a good idea to hook it up to an external database instead of the built-in one, which doesn't offer the same level of performance and flexibility as an external database such as MySQL.

To connect Openfire to a MySQL database server, first create an empty database on that server, for instance with something like:

```
mysqladmin create openfire_db
```

Then, from under the Openfire installation (`/opt/openfire/resources/database/`),

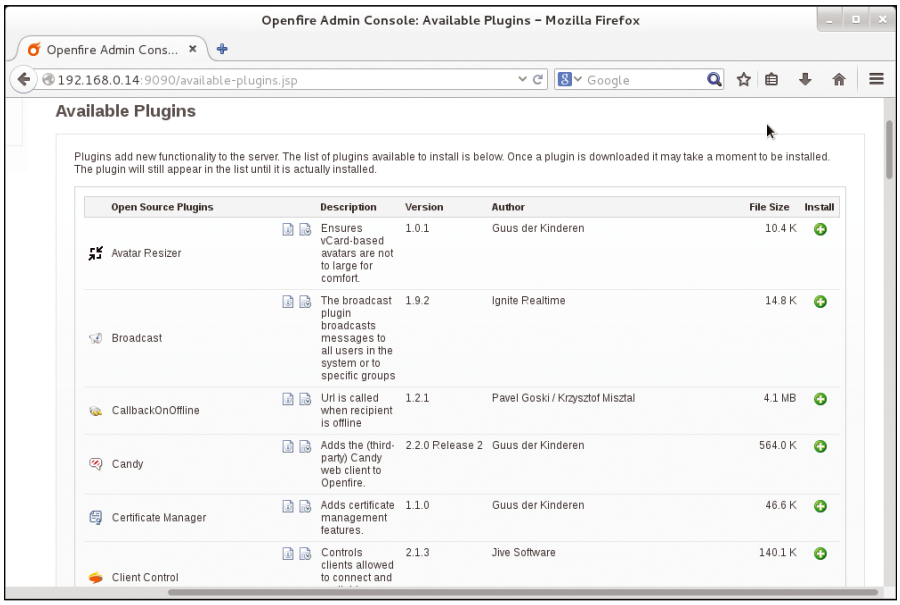


Figure 5: Besides the official plugins, you can find some third-party plugins for Openfire as well.

copy the database schema (`openfire_mysql.sql`), and use it to populate the database with something like:

```
cat openfire_mysql.sql ➥
| mysql openfire_db;
```

Openfire ships with database schemas for several databases, such as PostgreSQL, IBM's Db2, Microsoft SQL Server, Oracle, and more.

You'll also want to connect the Openfire server to a directory server such as OpenLDAP, if you have one running on your network [5]. You need to know the hostname or IP address and the port of the machine on which the directory server is running, its base DN (look for it in your OpenLDAP configuration file), and the authentication information. In the Openfire admin interface, toggle the *Directory Server* radio button in the *Profile Settings* step in the configuration wizard. Select the type of directory server you are running and enter and test the settings. If Openfire is able to connect to your directory server, you'll be able to pick out the elements from the directory server that you want to use to populate the users' IM profiles.

The easiest way to make changes to the Openfire server's database or profile settings is to rerun the setup wizard. For this you'll have to disable the administration console, by editing the `/opt/openfire/conf/openfire.xml` file and changing the `<setup>true</setup>` entry to `<setup>false</setup>`. Save the file and restart the server, which will now greet you with the setup wizard instead of the administration console login page.

Openfire behaves nicely with most network components if you have the right connection settings. Don't forget to configure your firewall to handle Openfire traffic. In addition to the admin console that runs on port 9090, Openfire uses some other ports to facilitate communication. In your firewall, make sure to forward traffic on port 5222, which clients use to connect to Openfire; port 7777, for file transfers; and any other port specified in your Openfire configuration.

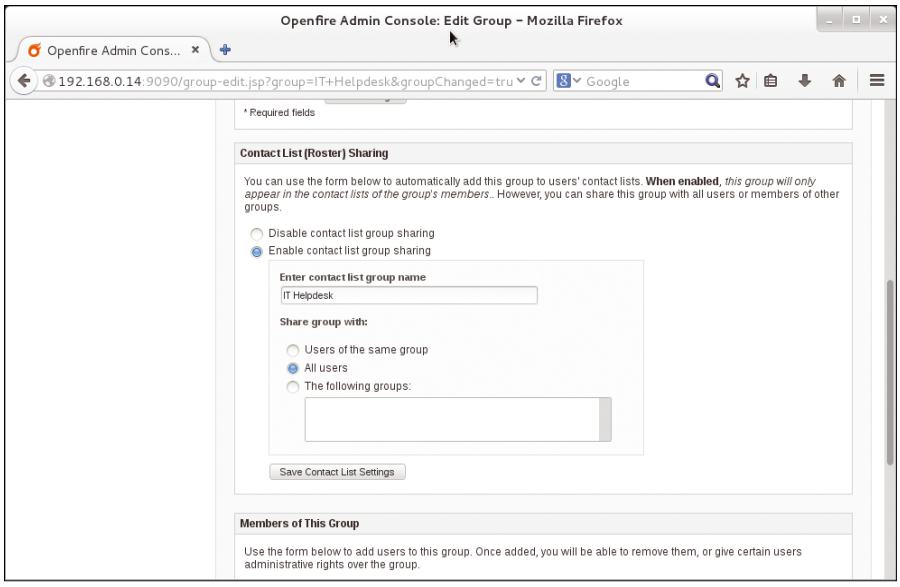


Figure 6: You can share a roster with a specified group of users.

Lots to Explore

Although I have covered a lot of ground, Openfire is capable of much more. For instance, you can create and manage chatrooms, run Openfire as a distributed load-balancing server across multiple physical installations, and connect users from two Openfire servers from different locations.

You can also use the Asterisk-IM plugin to link Openfire into your Asterisk VoIP gateway for unified communication. Besides enabling users to communicate with each other within your organization, you can also use the Candy plugin to hook Openfire to your website and use it to run an online help desk.

As with any server software, a shopping list of your exact IM needs should help you select the most useful software. However, Openfire is so diverse and malleable that you can use it in pretty much any situation to enhance communication. ■■■

Info

[1] Openfire: *https://www.igniterealtime.org/projects/openfire/*

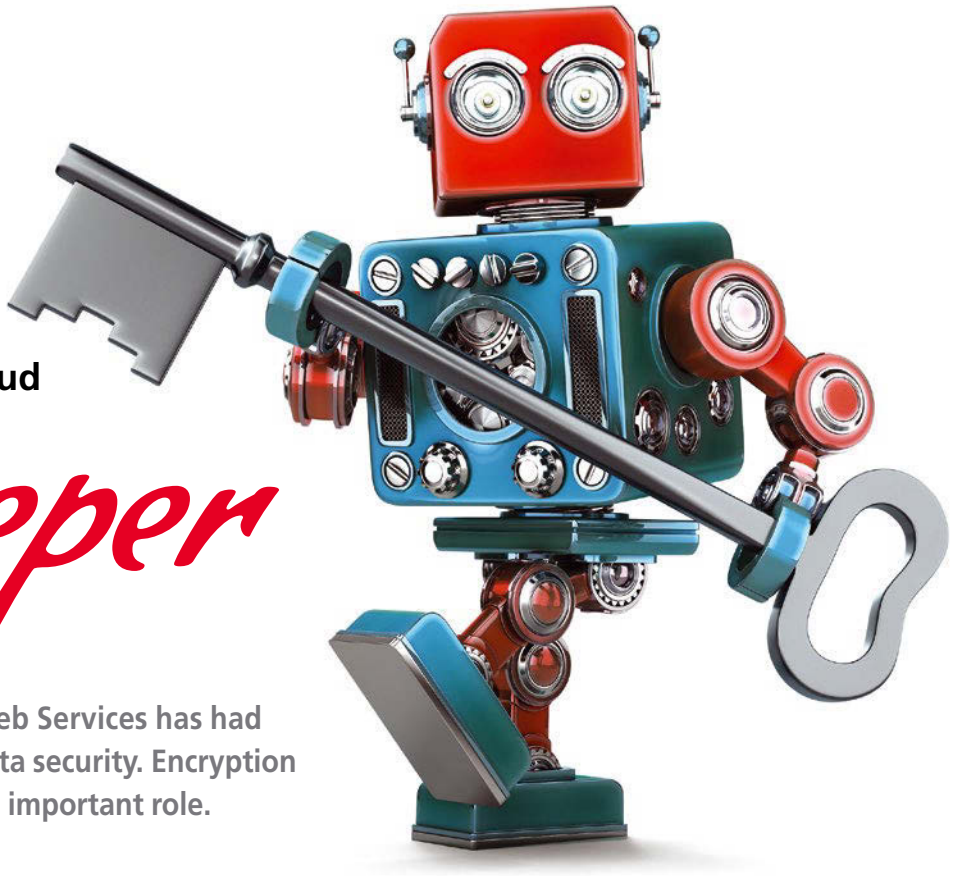
[2] Openfire protocol support: *http://download.igniterealtime.org/openfire/docs/latest/documentation/protocol-support.html*

[3] Openfire downloads: *https://www.igniterealtime.org/downloads/index.jsp*

[4] Spark login issues: *https://discourse.igniterealtime.org/t/login-issues-since-spark-2-8-0/41881*

[5] Openfire LDAP Guide: *http://download.igniterealtime.org/openfire/docs/latest/documentation/ldap-guide.html*


FREE DVD!
ALL THE SOFTWARE YOU NEED!
openSUSE
THOUSANDS OF FREE TOOLS IN EASY REACH!
SWITCH TO LINUX NOW!
2019 Edition
GETTING STARTED WITH
LINUX
Learn how to set up a Linux system to:
Listen to Music
Surf the Web
Play Games
and Much More!
Process Photos
MORE Powerful,
MORE Secure,
MORE Fun!
JOIN THE
LINUX REVOLUTION!
LINUX
WWW.LINUX-MAGAZINE.COM
300+ BEST BASH COMMANDS
SAVE 15%
on Linux Certification
See details inside
LINUX SHELL
HANDBOOK
2019 Edition
SUPERCHARGE
YOUR LINUX SKILLS
Travel Light
with fast and graceful keyboard commands
Power at Your Fingertips
Manipulate text strings
Pipe and redirect output
Monitor processes
Manage users and groups
Create easy automation scripts
WWW.LINUX-MAGAZINE.COM
FREE DVD!
LibreOffice installation files for Windows, Linux, and macOS
GET PRODUCTIVE
OFFICE TOOLS FOR EVERYONE!
DISCOVER
LibreOffice
Free Office Suite
Find out why 75 million users have stopped paying for office software!
Runs on Windows, Linux, and macOS!
Create your own:
Word processing docs
Spreadsheets
Presentations
Databases
Use open file formats that other programs can read and import!
Also inside:
Extensions
Templates
Macros
LINUX
101 COOL LINUX HACKS
101 COOL LINUX HACKS
Inspirational tricks and shortcuts for Linux geeks
Repair your bootloader
Cure the Caps Lock disease
Tricks with terminal output
Disable your webcam and mic
Run C one-liners in the shell
Undelete lost files
Ignore case in file names
PHONING IN
Sync your phone with a Linux desktop
QUICK SWITCH
Change to a second distro using chroot
LINUX
WWW.LINUX-MAGAZINE.COM



Data Security in the AWS Cloud

Key Keeper

As a cloud market leader, Amazon Web Services has had to put a great deal of thought into data security. Encryption options and key management play an important role.

By Konstantin Agouros

You've probably seen T-shirts emblazoned with "There is no cloud; it's just someone else's computer." This skepticism results from the management policy of quickly outsourcing as many IT services as possible, with the sole focus on efficiency and cost savings. As a result, data security becomes a secondary feature that the shrinking IT department must somehow guarantee.

Admins who simply run their applications in the cloud run the financially significant risk of violating the General Data Protection Regulation (GDPR), for example, if they store unprotected personal data on servers outside the European Union. However, the online bank N26, which runs entirely in Amazon Web Services (AWS), has passed an audit by the German regulator BaFin (in this respect), showing that it is feasible to operate cloud services compliant with strict rules.

In addition to the choice of the runtime environment (configured as the "region" on AWS and other cloud providers), there are several options for encrypting data for cloud storage. At the last AWS Summit in Berlin, the CTO of AWS, Werner Vogels wore a T-shirt that advocated "Encrypt Everything." If encryption is the answer, then who has access to the keys and where are they kept?

Author

Konstantin Agouros works as Head of Open Source and AWS Projects at Matrix Technology AG, where he and his team advise customers on open source and cloud topics. His new book *Software Defined Networking: Practice with Controllers and OpenFlow* has been published by de Gruyter.

Who Can Do What?

The first question for data security in the cloud concerns read and write permissions. This issue raises its head whenever you deploy any type of IT service and starts with user management. Weaving a complex structure of authorizations that define which user can access which data, servers, and other resources can be a Sisyphean task, with changes occurring constantly in IT operations.

The sheer number of possible permissions from which admins can assemble roles and services are far greater in a cloud like AWS. Finding the permissions you need for a particular cloud service to work without allowing too much is never going to be trivial. The complexity of the task can drive admins to distraction, prompting them to press *Allow everything* and thus release confidential customer data in an openly accessible Amazon Simple Storage Service (S3) bucket (Amazon's object store). Although this is inexcusable, it is something that you can at least empathize with from personal experience.

Data protection to and from the cloud, and on internal transfer paths between services, is another consideration. Many admins will suggest enabling TLS. But in practice, the success of the project often depends on where the certificates originate.

While a multitude of AWS services are affected by access controls, I have limited this article to two basic AWS services: the S3 object store and the Elastic Compute Cloud (EC2) virtual machine (VM) service. Additionally, I will look at AWS key management, as well as a few aspects of Identity and Access Management (IAM), which distributes users and their rights.

Trinity

The confidentiality, integrity, and availability (CIA) triad plays an important role in determining data security. Confidentiality (C) means that only authorized users see the data content. On a public web page, the group of permissions will often be *All*.

Integrity (I) means that only authorized users can modify the data. Where applicable, this means that some of the authorized users are only able to change a certain dataset within defined value ranges. A bank employee, for example, can only transfer money to accounts per customer request, instead of at will.

Availability (A) pertains to how data is maintained and stored. If all the important corporate data is on a single hard disk without a backup, and the disk bites the dust, then the data is no longer available.

Protection from Whom?

When it comes to protection against unauthorized read (C) and write (I) access to the data in the cloud, admins need to determine who has access to which data. There is public access via the Internet, plus a small group of users with different authorization levels (i.e., order processing does not need access to human resources' salary tables).

Since the whole thing runs on a third-party infrastructure, you also need to consider protection from the cloud provider's employees, as well as access controls for the in-house administrators who manage the systems. This is particularly relevant for personal data, such as salary tables.

Availability is something that AWS customers can typically assume to be a given. With S3, for example, the user would have to actively disable high availability to voluntarily suffer from data loss in the event of a crash. In addition, the object store supports versioning so that the customer can revert to older versions in the event of problems.

Users, Roles, and Rights

In the AWS cloud, the simplest hierarchy level is that of accounts containing users who are assigned authorizations within the account, such as the ability to create and start VMs or databases. The *IAM* configuration area is used to manage users or admins.

AWS recommends setting up accounts with sub-accounts. This allows the AWS customer to impose company-wide policies so that even an admin with full rights for a sub-account cannot violate the organizational rules.

When generating an account, AWS also creates the superuser's credentials for this account. By clicking on the user list, the admin will find this superuser. The user has full access to all functions offered by AWS (the exception would be a sub-account with an organizational policy). If the admin creates a second user here, the user is only granted explicitly assigned rights. When creating a new user, you are first prompted for the user name and details of how this user will log on, via CLI/API and/or the Web Console.

Next, the admin assigns rights by selecting from existing user groups (for example *S3 Admins*, *Networkadmins*, etc.), assigning roles (*S3 Admin*, *Networkadmin*), or as individual assignments at policy level. If you really want to make your life complicated, you can also define an arbitrary combination of these rights for the new user.

To avoid selecting overly liberal permissions by mistake, a permissions boundary can be defined within which, say, the security administrator responsible for AWS restricts permissions in a policy. If a conflict then arises between this limitation and the assigned rights, the limitation wins.

Rights to Resources

AWS controls access through policies. A policy consists of a set of statements, each granting one or more rights to a resource (with wild cards) to a role or user. Optional conditions are possible. Listing 1 shows a section of a policy in JSON format [1].

The `Sid` field contains a name for the permission, but it is optional. `Effect` allows or denies access. The `Action` field contains a list of the API access instances at issue. In the example, these are all listing and downloading operations in the S3 API. The `Resource` field contains a list of targets for the operations, formulated as Amazon Resource Names (ARNs). The example shows a bucket named `confidential-data` and its contents. If you do not include the last field `Condition`, then the rule would be universal.

The condition in line 14 ensures multifactor authentication of the user for this rule to apply. Depending on the logic to be mapped, the admin either writes individual policies in this form and combines them or bundles several statements into a single policy.

The Right to Interact

Policies are not only used for user access controls, but also to govern the interaction between AWS entities. A lambda function wanting to send a Simple Notification Service (SNS) [2], for example, needs a role that contains a policy with the corresponding rights in the SNS area.

Policies determine which operations are allowed on which objects. The admin assigns them to users or functions. In regard to the CIA triad, a policy controls who can access the data within the created AWS objects. This does not consider the confidentiality and integrity objectives in relation to the AWS administrators.

S3

S3 [3], one of the oldest services in AWS, is divided into buckets at the top level. A bucket contains folders and ob-

Listing 1: JSON Policy Definition

```
01 {
02   "Version": "2012-10-17",
03   {
04       "Sid": "s3zugriff",
05       "Effect": "Allow",
06       "Action": [
07         "s3:List*",
08         "s3:Get*"
09       ],
10       "Resource": [
11         "arn:aws:s3:::confidential-data",
12         "arn:aws:s3:::confidential-data/*"
13       ],
14       "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
15     }
16   ]
17 }
```

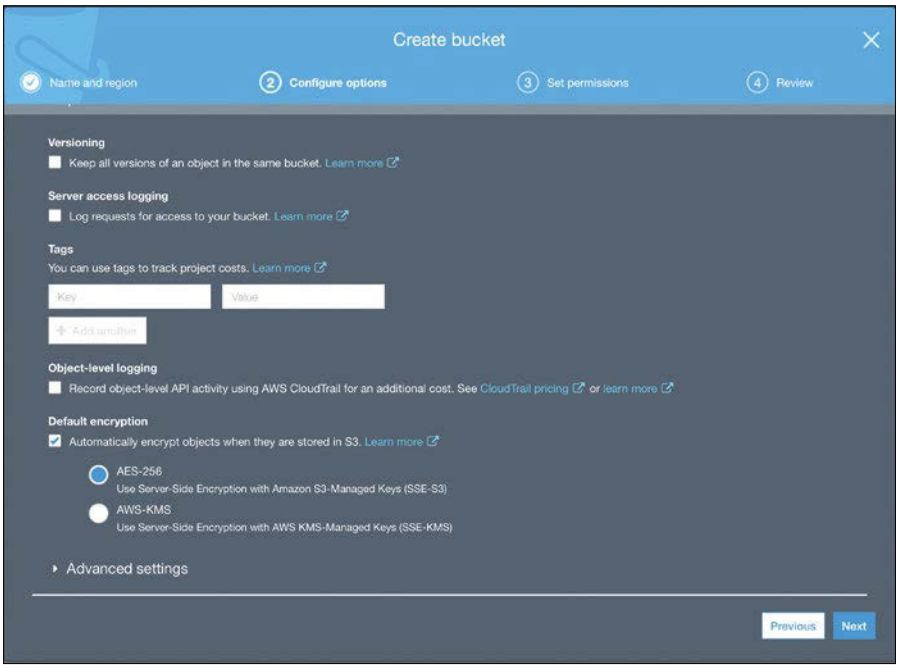


Figure 1: Creating an S3 bucket.

jects/files. S3 buckets are also the easiest way to launch a static website in AWS. You drop the files that make up the website into a bucket and then make them available via HTTP(S) with a few clicks. In the past, this occasionally went wrong, because confidential data was accidentally left in the clear on the Internet.

Figure 1 shows the settings for creating a bucket. If encryption is enabled, the user can choose whether the AWS system will use automatically generated data or the data stored in the Key Management Service (KMS) [4]. Encryption applies to the objects in the bucket. By default, public access is also blocked (Figure 2).

The admin can also control what kind of encryption applies at the folder level (Figure 3). The last stage is the individual object (a file, for example). The user can set encryption here (if using) along with the encryption method.

Alternatively, you can encrypt the objects locally in S3 before uploading them, so that there is no AWS key capable of decrypting them.

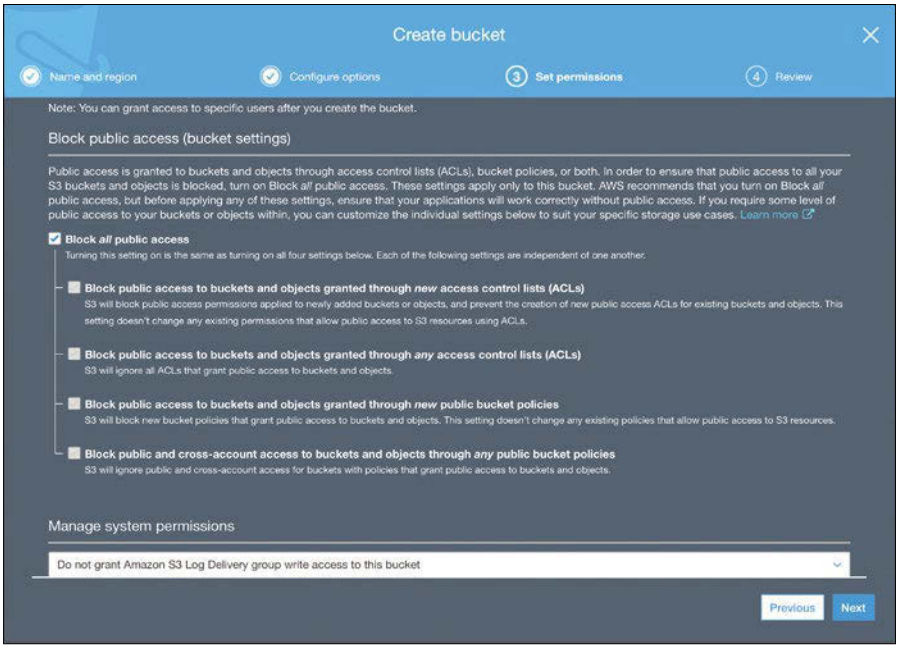


Figure 2: Blocking public access in bucket settings.

KMS

AWS by default uses keys that are automatically generated on the fly to quickly encrypt data. However, the confidentiality and integrity objectives pertaining to the Amazon admins are still not in place.

If you want more control over the keys, you need to use KMS [4]. In the KMS console, AWS offers a list of implicitly generated keys for encrypting databases. For customer-managed keys that are generated by the admin in the console and then have a policy applied to them, a policy is attached to a key – anyone who uses the key is allowed to do what the policy states.

When you create your own key in KMS using KMS to generate the key, the dialog prompts you to define who can manage the key and who can use it for encryption and decryption. Based on this, the user guide calculates a policy, which it attaches to the key.

Instead of having AWS generate a key, an admin can store an externally generated key. In this case, the console requires the admin to select an algorithm to pack the key before uploading it and a token to unpack the key again (see [5] for an example).

Finally, Amazon offers a CloudHSM cluster [6]. If you choose this option, you are binding several hardware security modules (HSMs) that contain a tamper-proof key; this achieves the highest level of control possible in the cloud.

All requests for encryption are answered by the HSMs; their hardware makes sure that nobody reads the keys. Designed as a cluster, CloudHSM is highly available (the A in CIA) – which is important for people who secure large sections of their infrastructure with HSM.

Due to interchangeable components and API compatibility, taking the big step towards CloudHSM in an architecture does not have to happen at the very beginning of development. In this configuration, the remaining trust topic is that the KMS clients are under the control of AWS (i.e., they store decrypted data in RAM).

VMs with Encrypted Hard Disks

Amazon's EC2 VM service uses S3 for its virtual hard disks, making encryption of the hard disks possible. Linux adminis-

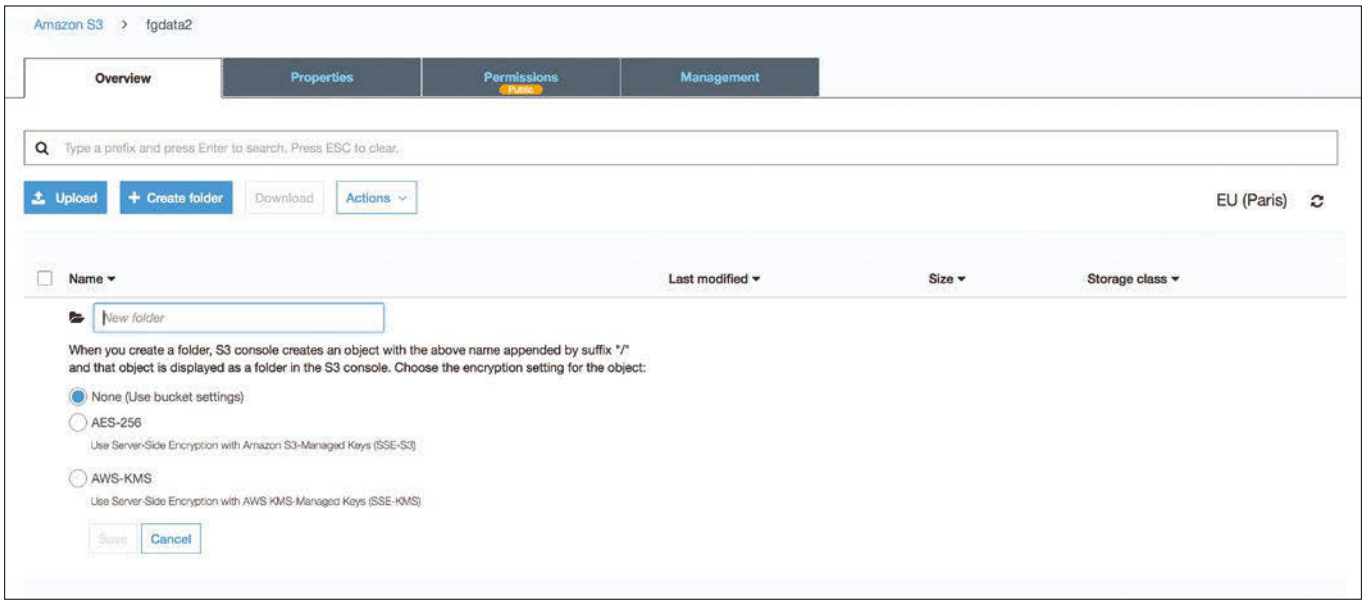


Figure 3: The folder encryption options in S3.

trators have been aware of this kind of protection for a long time at the level of encrypted partitions or volumes. This either means that the admin has to enter a password when booting the machine or that the password has to be stored in the bootloader. In the latter case, the data remains unprotected in the event of theft.

The AWS Cloud uses KMS behind the scenes. When an admin creates a new VM, the memory management menu offers the option of encrypting the hard disk (Figure 4). The user keys generated in the selected AWS region appear in the selection list.

When the VM starts, the hypervisor retrieves the data for decryption. Admins who failed to secure access permissions to the key when it was created are allowed to attach the hard disk to a VM, but are unable to read the data, similar to any other encrypted volume.

Since AWS unfortunately does not provide a console for a Linux VM, the ability to work with Linux on-board tools does not exist for the root volume. If the confidential data resides on a separate partition from the VM, the admin boots the VM in the normal way, manually mounts the volume, and enters the password; this means that the key does not reside in AWS.

Conclusions

Even AWS cannot protect private keys against every form of threat on third-party servers. Utilizing the CloudHSM services moves admins towards an acceptable level of protection for their corporate data. Regardless, users have to have a certain amount of trust in Amazon or – where possible – adapt the cloud architecture to avoid storing sensitive data.

KMS makes using encryption relatively simple, which hopefully mitigates some admins' tendency to avoid encryption altogether. ■■■

Info

[1] AWS policies: *https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html*

[2] SNS: *https://docs.aws.amazon.com/sns/latest/dg/welcome.html*

[3] S3: *https://docs.aws.amazon.com/s3/?id*

[4] KMS: *https://docs.aws.amazon.com/kms/?id*

[5] Creating and importing keys: *https://docs.aws.amazon.com/kms/latest/developerguide/importing-keys-encrypt-key-material.html*

[6] CloudHSM: *https://docs.aws.amazon.com/cloudhsm/latest/userguide/introduction.html*

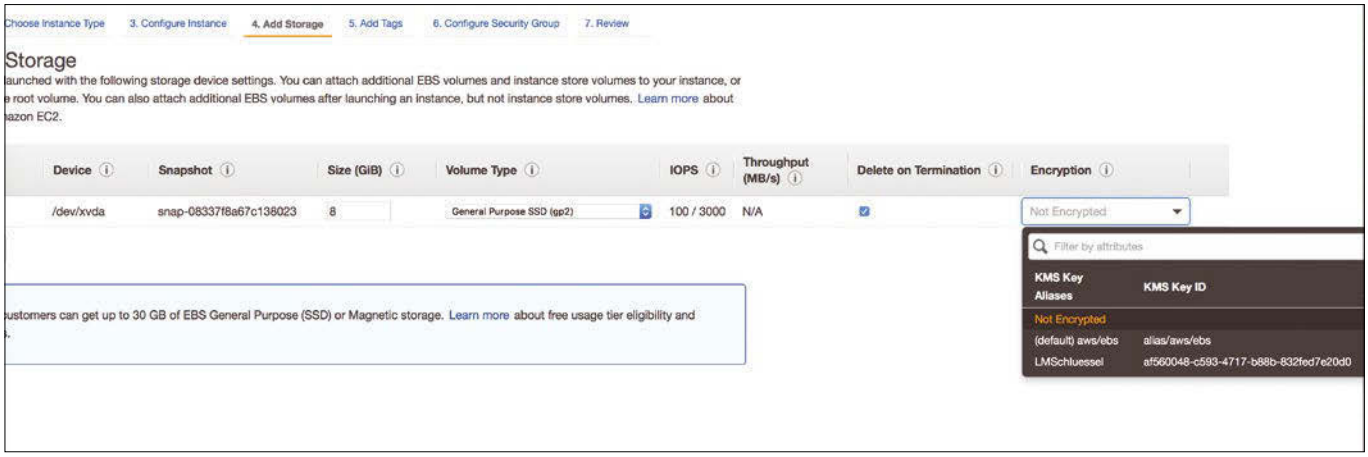


Figure 4: Selecting an encryption option in AWS EC2 when creating a VM.

The GoBandit slot machine simulation game

Pull!

A simple simulator for a Las Vegas-style gambling machine lets you practice your Go programming.

By Rick Timmis

GoBandit is an open source game that simulates a slot machine, sometimes called a fruit machine (Figure 1). In traditional parlance, these machines were known as one-armed bandits, perhaps because they had a single lever – and usually took your money.

GoBandit uses the Simple DirectMedia Layer 2 (SDL2) library, which provides access to OpenGL and Direct3D and works across platforms. In theory, then, you should be able to build for Linux, macOS, and Windows, although in this article I focus on Linux.

Structure

Before diving into code, I'll take a few moments to explore the structure of GoBandit, which comprises the custom defined types Tiles, Board, Score, and Control. Each of these types has a number of associated functions. Control and Tiles types implement the Controller and TileSet interfaces.

The `main.go` file initializes the main structures and then passes pointer references to the Controller interface, which implements the Control type, data structure, and game play logic.

Implementing an interface enables the separation of the Control type from the implementation code. The beauty of this approach is that `main.go` now becomes pluggable. I can instantiate any control code that I choose to write, provided it implements the functions defined in the Controller interface (i.e., `spin()`, `nudge()`, `hold()`).

The Tiles type and the TileSet interface let you set up the tiles that make up a fruit machine, but this could easily be a card game with 52 cards in a set or a clone of some of the popular apps that fit into this genre. Doing so is as easy as adding a new `/resources/tile_image/` folder with a set of tile images.

Installing Go

Before I get too deep into the internals, you should get a copy of the code and get the game up and running. The following instructions are appropriate for Debian and Ubuntu – other distributions will need to follow the instructions for installing the SDL2 dependencies [1].

To begin, open a terminal and enter:

```
sudo apt-get update
sudo apt install libsdl2{
  ,-image,-mixer,-ttf,-gfx}-dev
```

The code in this example needs Golang. Ubuntu users can use the snap package manager, which provides a contained environment accessible to the local user. I also provide instructions for installation with Apt that is suitable for system installation on both Debian and Ubuntu distributions. To install Golang, enter one of the following commands for snap or Apt:

```
snap install golang git-ubuntu
sudo apt install golang git
```

The next step is to check out a copy of the GoBandit code [2]. With Go now installed, simply enter:

```
go get github.com/ricktimmis/gobandit
```

The final step is to use `go get` to retrieve the various dependencies:

```
go get -v github.com/veandco/go-sdl2/
  {sdl,img,mix,ttf}
```

With Go installed, you should find a new `/go` folder in your home directory, which is the standard Go path (`$GOPATH`) and usually contains the `/bin`, `/pkg`, and `/src`

Photo by anna-samoylova on Unsplashcom

Figure 1: GoBandit fruit machine simulator.

folders, Go will create `/bin` and `/pkg` on the fly when required. The `/src` folder contains a `/github.com` folder, within which resides a `/ricktimmis` folder and, inside of it, the `/gobandit` folder. This structure provides a consistent format for the management of source code and dependencies. When written as `/go/src/github.com/ricktimmis/gobandit`, it makes sense and is easy to copy from the `github.com` stub and paste into a browser to get directly to the code on GitHub or other hosting service.

The interesting files in the project each come with a `_test.go` test suite, and you should be able to run the test from the command line:

```
$ go test
```

To get the game up and running, enter:

```
$ go build -O gobandit.bin -i ../gobandit
./gobandit.bin
```

Next, I'll look at the interesting types.

Tile Type

The `tiles.go` struct defines a new `Tile` type and data structure to hold image sets and values; `tiles.go` defines and meets the requirements for the TileSet interface. A `Tile` loads the data model meta information from the configuration and expects image assets as PNG files. A TileSet provides the faces and values that will be displayed in the rows and columns of a Board and are declared by the filename with the structure `<facename_value>.png`.

This abstraction enables `TileSets` to be created as pluggable components to the GoBandit game. Thus, a fruit machine game might hold a TileSet of six fruits, or a playing card game could hold a TileSet of 52 playing cards. A standard set of functions – `Count()`, `Shuffle()`, `Next()`, `GetFace()`, `GetValue()`, `GetImage()`, `GetTile()` – make up the interface's requirements. These functions are attached to the `Tile` struct by convention of their written signature. I won't go into the details of function signatures within Go in this article, because the Go language documentation on this topic [3] is excellent.

To bind a function to a struct, refer a pointer to the struct as the first element in the function signature:

```
func (t *Tile) GetValue() int {
  return t.value[t.currface]
}
```

The `func` keywords (e.g., `(t *Tile)`, where the asterisk denotes a pointer to a `Tile` struct) say "associate the following function signature as a method of the struct pointed to by `t`." Next, the function `GetValue()` is defined, which then lets you call that method from the struct:

```
myTile := new(Tile)
tilevalue := myTile.GetValue()
```

This fairly simple construct is important to understand, because it is a very common idiom throughout the GoBandit codebase.

Board Type

The simple `board.go` struct holds information about the game board. A Board is initialized in the `Init(c,r,t)` function, where `c` and `r` are the number of columns and rows, respectively, and `t` is a TileSet interface. At the time of writing, the board is limited to eight columns and eight rows. However, this is purely for visual aesthetics in the context of using GoBandit as a fruit/reel game. The board's initialization function also generates a data array into which `TileSets` are loaded. The key thing to recognize here is that the `board.go` file is the place to modify the game's overall board behavior.

Score Type

The `score.go` struct provides an evaluation component for calculating rules against the board. The simplest usage is to initialize a variable with a function and then call the `evaluate()` method, injecting the function through the variable. Score also defines the Scorer interface, which declares two standard functions that fulfill that interface contract: `Init()` and `evaluate()`. The interesting programmatic aspect of the scorer is the use of function passing, which is a nice little trick in Go, whereby you can prepare a function that holds the scoring rule(s) you want to apply. In this implementation, I keep it fairly simple, although when you pull the code, you'll see an open issue for extending this aspect of the codebase. I also use a deferred function to catch failures if the ruleset function fails.

Deferring the error trap is a nice example of idiomatic Go.

Control Type

Here, I provide a Control type and a Controller interface. The `control.go` struct contains most of the GoBandit logic; essentially, this is where the action is, and it is where you are encouraged to start experimenting by refactoring and extending.

Listing 1 collapses the code logic just to the function signatures, so that you

Listing 1: control.go

```
01 // All the interesting functions are here !
02 // Implementation functions that do the work and the rendering
03 // This is the stuff you want to mess with to get different game behaviour
04 func (c *control) spin() error {....}
05 func (c *control) hold() {....}
06 func (c *control) nudge() {....}
07
08 // FIXME - Requires a file path to the background image
09 func (c *control) drawbackground(i string) error {....}
10 func (c *control) drawboard() error {....}
11 func (ctrl *control) playnext() error {....}
12 func (c *control) checkscore() (int, error) {....}
13 func (c *control) spinimation() error {.?}
```

can see, simply, the main functions. Each should be self explanatory, with perhaps the exception being `spinimation()`, which provides little at the moment but is designed to enable the development of animations on the board. An example might be to implement an animation that highlights a winning line, or perhaps even something more exotic, such as explosion animations across matching or scoring rows and columns.

Resources Directory

The `/resources` directory has five subfolders – `backgrounds`, `controller_assets`, `examples`, `fonts`, and `tile_images` – all of which should be self-explanatory. However, `controller_assets` provides the button images, and `examples` contains some useful Go code examples that provide useful reading for the inquisitive.

Start Coding!

Now that you have explored the files, types, and interfaces and have a better understanding of the overall layout, you are in a position to start making some changes and experiment with the project.

You'll find a number of *FIXME* markers, some of which also have GitHub issue number tags (e.g., *Issue #3*). These should serve to mark easy places to get hands-on with the code.

Ultimately, I focus in this article on getting new contributors' feet into the water, so they can prepare and submit their first pull request (PR) and experience the thrilling feeling of having a PR accepted and merged into a project codebase.

At this point, I want to encourage you to wander over to the project's GitHub page and review the issues board. There, you can either pick up an issue to work on, open an issue to request a fix or feature, or, importantly, open an issue for an idea that you are going to work on, which also lets the community know that you're working on a fix or feature and enables the community to help contribute to your efforts. Finally, if you have any questions or need any help to get things working, open an issue, and I'll do my very best to help you get a resolution. ■■■

Info

[1] SDL2 dependencies: *https://github.com/veandco/go-sdl2*

[2] GoBandit project on GitHub: *https://github.com/ricktimmis/gobandit*

[3] Go docs: *godoc.org*

Author

Rick Timmis (*http://www.ricktimmis.com*) is a charismatic, optimistic, and sociable geek. He is an active participant in the free software and open source community, as well as a founding member and former CEO of the UK Open Source Consortium. He is currently a community manager, council member, and developer with the Kubuntu flavor of the Ubuntu Linux distribution.

Working with AVR microcontrollers

Hop on Board

Learn about the main tools needed to work with AVR microcontrollers, including the avrdude command and the Arduino IDE. *By Bruce Byfield*

AVR microcontrollers are among the most popular boards used today in open hardware and embedded systems. Originally made by Atmel and now by Microchip Technology, today they are often compatible with Arduino's boards and tools. Rather than full computers, they operate on firmware or relatively simple instruction sets that require only a moderate knowledge of programming to customize by uploading revised firmware. Although graphical tools exist for flashing firmware on other operating systems, on Linux, the main tools are the `avrdude` command [1] and, increasingly, the Arduino IDE [2].

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/*.

Installation and Setup for avrdude

Along with its basic dependencies, `avrdude` is available from the repositories of most major distributions. However, what is not immediately apparent is that, to write code or flash firmware, you will also need to install `make`, `avr-gcc`, and `avr-libc` (or, in Debian, `libc-avr`). In addition, coding will also require `gcc` for compiling.

To communicate with a device, you will need the setting for at least two options: `-p DEVICE-CODE`, which takes a code that identifies the type of microcontroller, and `-c` option `PROGRAMMER`, which defines the instruction set used by the board. Although a list of microcontrollers for use with `-c` appears in the man and info pages, it appears not to have been updated for at least two releases, while neither Help file lists programmers. To get a current list of both microcontrollers and programmers, scan `/etc/avrdude.conf`, or, if you have created a personal configuration file, `.avrduderc` in your home directory (Figure 1). Otherwise, if you are using a board released in the last couple of years, you may not have the information required. Sometimes, the ID code for a similar board may give you partial functionality, but don't count on it.

In some cases, you may also need to specify the USB port a device is using. You may be able to simply specify `-P usb` to have the device detected. In some cases, though, you might have to be more exact. In Debian, you can identify the exact port by logging in as `root`, and running `dmesg`. In Figure 2, some of the `dmesg` output for a Treasure Macropad Type-9 (reviewed elsewhere in this issue) is shown that contains port information.

Testing and Flashing with avrdude

When you finish installing all necessary packages, you should test the connection with the device. The most practical way to make the test is given in Adafruit tutorials [3]: Create a file called `main.c` that contains the following code:

```
int main(void) {
    return 0;
}
```

Photo by Fabrizio Forte on Unsplash

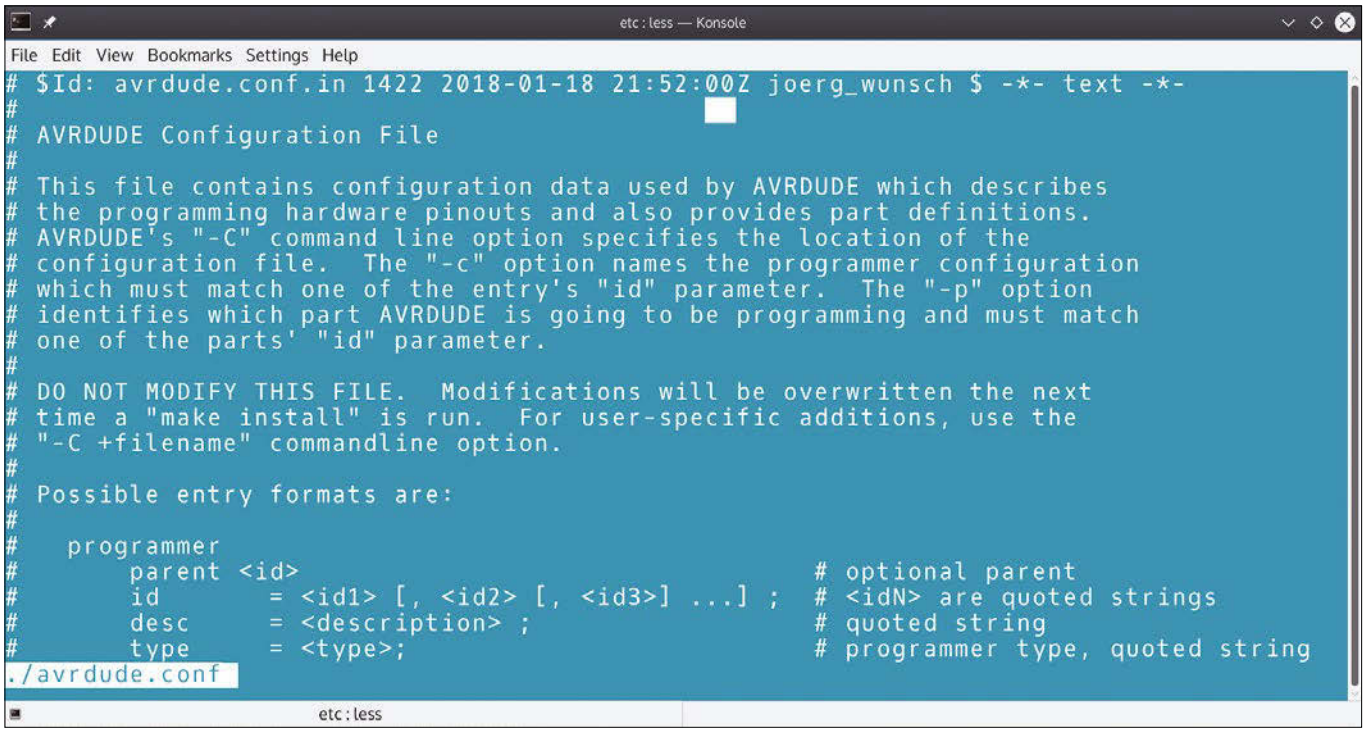


Figure 1: The start of a configuration file for `avrdude`.

and then copy it with the following command to compile it:

```
avr-gcc -mmcu=MCU-CODE
  -Wall -c main.c -o main.out
```

Use the appropriate device code for the `-c` option (see above).

`-Wall` tells the command to display all warnings, while the rest of the code outputs a copy of the file called `main.out`. As you can probably tell, the file being compiled does nothing.

When the file compiles, load it to the device with:

```
avrdude -p DEVICE-CODE -P PORT
  -c PROGRAMMER main.out
```

See above for details about how to structure the command. The structure given assumes you are using a modern board, connected by a USB port. If not, consult the man page for other options you may need.

When the file is flashed, use

```
rm -f main.out
```

to remove it from the device. You can use the same command structure to upload custom firmware. Alternatively, for convenience, you can create a makefile to save typing, then use the command `make`, disable the bootloader, and add `makeflash`. Depending on the device, you can disable the bootloader by pressing a button, or poking a straightened paper clip into the hole on the back of the device. Before flashing, `avrdude` will erase the current firmware on the board unless the `-D` option is used. Generally, however, you want to erase, because many devices do not have the memory for two firmware files and would not have the ability to choose between them.

Note that most other options are for legacy support. Most users are unlikely to need options like `-b` to set the baud rate or to configure parallel or serial port connections today. Exceptions are `-v` for verbose mode and `-C CONFIG-FILE` to choose a configuration file in an unusual position.

Using the Arduino IDE

Many Arduinos support variants of the AVR developed by Atmel, including those that use the STK600, AVR109, and AVRISP mkII programmers. These include ATtiny4/5/9/10 devices and the AVR Butterfly evaluation boards. All supported boards can be programmed using the Arduino IDE, which might be preferable for those who would rather use a graphical interface. A particularly handy aspect of the Arduino IDE is that you can download a script (or sketch, as the IDE calls it) to the device directly from the IDE.

Before using the Arduino IDE for development, first set up the board in *File | Preferences*, by adding a URL to the board in the *Additional Boards Manager URLs* field (Figure 3). The field supports

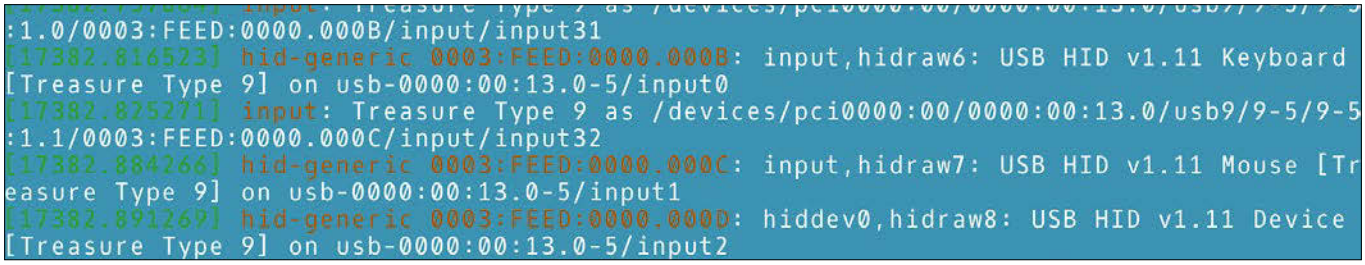


Figure 2: On Debian-based systems, running `dmesg` shows to which USB port a device is connected.

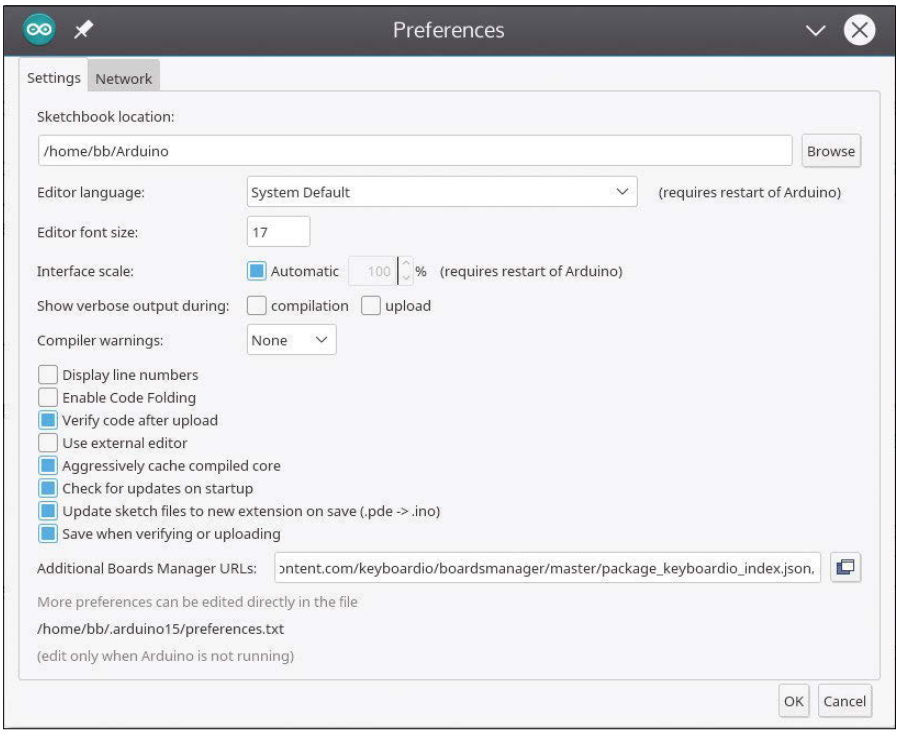


Figure 3: To work with an AVR microcontroller in the Arduino IDE, first add the Boot Manager URL for the device.

multiple URLs in a comma-separated list. After you click the *OK* button, settings for the board appear in the Tools menu (Figure 4). From the Tools menu, you can set the board manager, the connecting port, and the programmer. Most boards these days come with a bootloader that allows them to be flashed via a USB port, and are marked by a slightly different name from related boards.

If you want to use the board as a boot manager for another board, you can generally use a sample in *Files* | *Example* as a starting point or find an example that can be modified online. Then, setting the device in the Tools menu, set up the board's pins in a specific configuration, which you will have to locate in its documentation, and select *Burn Bootloader*. Once setup is successful, you can write your firmware in the Arduino IDE and load it to the device directly.

Figure 4: Settings for a device in the Arduino IDE (in this case, the Keyboardio Model One keyboard).

After Setup

Once your computer can communicate with `avrdude` or the Arduino IDE, you are ready to flash a device. Typically, coding will be in C, and possibly C++. Basic programming can sometimes be done with a minimum of programming knowledge. For example, in firmware for a keyboard, you can expect a keymap, a text-based diagram of character assignments to each key (Figure 5). Intermediate programmers may be able to add macros that further customize a device's behavior. Such macros can often begin with modifications of code snippets from already installed macros or from examples in the Arduino IDE. You may also want to write a configuration file that is in your path on the computer, so that you can enter `avrdude` commands without continually adding the same options. If you want to work at a more advanced level, a place to start is with the `avrdude` manual [4]. General introductions to AVR microchips are also available [5].

Working with AVR boards is a specialized area of programming. It changes quickly and can be confusing for newcomers. For example, I recently spent most of three days trying to work with a board, getting no results until I realized that the device I was trying to flash used a specialist bootloader. However, with open hardware and the Internet of Things making such strides, a little basic knowledge can be useful. Armed with even the basics, you can customize an increasing number of devices exactly to your liking – a form of consumer empowerment that will be appreciated by many Linux users. ■■■

```
[QWERTY] = KEYMAP_STACKED
(___, Key_1, Key_2, Key_3, Key_4, Key_5, Key_LEDEffectNext,
LCTRL(Key_X), Key_Q, Key_W, Key_E, Key_R, Key_T, Key_Tab,
LCTRL(Key_C),   Key_A, Key_S, Key_D, Key_F, Key_G,
LCTRL(Key_V), Key_Z, Key_X, Key_C, Key_V, Key_B, Key_Escape,
 Key_LeftControl, Key_Backspace, Key_Delete, Key_LeftShift,
 ShiftToLayer(FUNCTION),

 LCTRL(Key_Z),  Key_6, Key_7, Key_8,     Key_9,        Key_0,        LockLayer(NUMPAD),
 Key_Enter,     Key_Y, Key_U, Key_I,     Key_O,        Key_P,        Key_Equals,
                Key_H, Key_J, Key_K,     Key_L,        Key_Semicolon, Key_Quote,
 Key_RightAlt,  Key_N, Key_M, Key_Comma, Key_Period,   Key_Slash,    Key_Minus,
 Key_RightShift, Key_LeftAlt, LCTRL(Key_S), Key_Spacebar,
 LockLayer(FUNCTION)),
```

Figure 5: Simple firmware coding can include light editing of existing files. Here, the keyboard layout can be edited with standard codes for each key.

Info

[1] avrdude: *https://linux.die.net/man/1/avrdude*
[2] Arduino IDE: *https://www.arduino.cc/en/Main/Software*
[3] Adafruit tutorial: *https://forums.adafruit.com/viewtopic.php?t=23266*
[4] Intro to avrdude: *https://www.nongnu.org/avrdude/user-manual/avrdude_4.html*
[5] Intro to AVR chips: *https://fossbytes.com/introduction-avr-microcontrollers-basic/*

The sys admin's daily grind: Tuning ntpd

Tempus Fugit

Charly Kühnast, sys admin columnist for 15 years, is searching for lost microseconds. *By Charly Kühnast*

Time is the topic I'm focusing on right now. On the one hand, I am of course happy to be celebrating the 25th anniversary of *Linux Magazine* and the fact that I have been allowed to contribute this column for more than 15 years. On the other hand, I am currently working on configuring my time servers. In the last issue [1], I briefly touched on the topic when we looked at `ntpviz`, the statistics visualization tool for the Network Time Protocol daemon (`ntpd`).

How to tune the time server for maximum accuracy using quite simple means was the topic in the October 2017 issue [2]. (Is it really already two years ago? How time flies.) At the time, I picked up a high-precision pulse per second (PPS) signal from GPS satellites. This allowed me to line up ticks received locally or from remote time servers to achieve microsecond accuracy on my server. Apart from scientific applications, nobody really needs that, but it's cool, so it was done.

When I tackled this configuration back in 2017, it was still by a fairly circuitous route. Feeding the PPS signal to `ntpd` required additional software, which I found on GitHub. It's easier today. The two daemons involved, `gpsd` and `ntpd`, work hand in hand on my Debian 10 – and without any extra software.

Listing 1: /etc/ntp.conf

```
01 server 0.de.pool.ntp.org prefer
02 server 1.de.pool.ntp.org iburst
03 server 2.de.pool.ntp.org iburst
04 server 3.de.pool.ntp.org iburst
05
06 server 127.127.28.0 minpoll4 noselect
07 fudge 127.127.28.0 refid GPS
08
09 server 127.127.22.0 minpoll 4 maxpoll 4
10 fudge 127.127.22.0 flag3 1 refid PPS
11 tos mindist 0.2
```

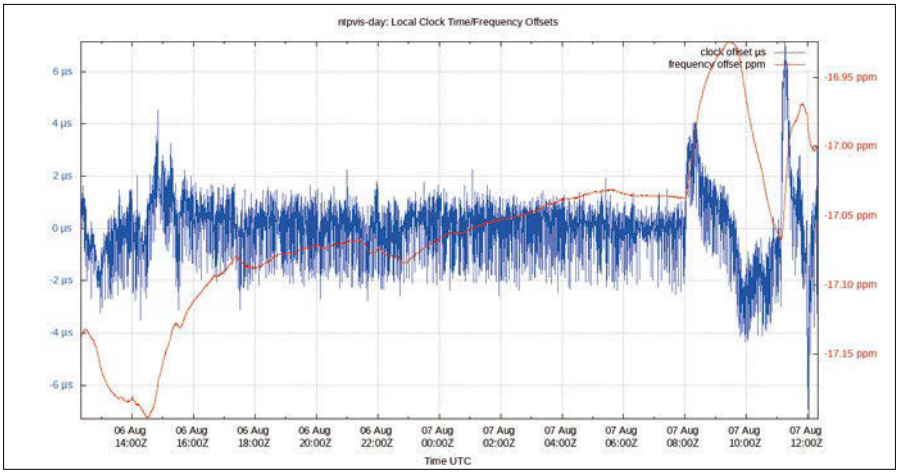


Figure 1: The computer time deviates from the ideal by just +/-2 microseconds. The irregularity starting at eight o'clock is a human-shifted-antenna fault.

I only have to tell the GPS daemon in the `/etc/default/gpsd` configuration file where to find the GPS hardware. In my case, it is connected to a USB port, so I need the following line:

```
DEVICES="/dev/ttyUSB0"
```

Also the `ntpd` configuration file is quickly completed with Listing 1. Lines 1 to 4 probably already exist in the file. It is important that one of them ends with the `prefer` keyword. Lines 6 and 7 include the GPS time signal, but only for statistics. The `noselect` keyword prevents it from actually being included in the calculations, because it is not very accurate. Lines 9 to 11, on the other hand, provide accuracy and bridge the gap to the PPS signal, which announces the beginning of a new second with high precision.

A Little Crazy

This allows my time signal to fluctuate by only a few millionths of a second (Figure 1). By the way, the deflections in the right part of the graph were caused by me frivolously moving the GPS antenna on the window sill.

Of course, my experiments did not go as smoothly as the story thus far suggests. In the first attempt, my time signal was no more accurate than before, whatever I did. After tearing my hair out for a couple of hours, it turned out that you can buy some GPS hardware that simply ignores the PPS signal. This realization cost me half an afternoon – it seemed to drag on endlessly if you ask me. On the other hand, time flies – but not always, apparently. ■■■

Info

[1] "Charly's Column – ntpviz" by Charly Kühnast, *Linux Magazine*, issue 228, November 2019, p. 26

[2] "Charly's Column – Precision Timekeeping" by Charly Kühnast, *Linux Magazine*, issue 203, October 2017, *http://www.linux-magazine.com/Issues/2017/203/Charly-s-Column-Precise-Timekeeping/(language)/eng-US*

Author

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

Sort and organize media files with Mediapurge

Distribution Point

If you have a download folder full of photos and music, Mediapurge can help you sort files and even remove duplicates, but beware of its quirks. *By Tim Schürmann*

Mediapurge is a real jack of all trades. It sorts media into subdirectories based on file names or metadata, converts file names to reflect a uniform pattern, and removes duplicates from your hard disk. To detect duplicates, it analyzes content and even recognizes photos stored in different formats. If desired, the software synchronizes your collection with a backup on an external hard drive. Plus, it can convert a batch of audio files into another format.

Although the proprietary software originated in the Windows world, Mediapurge v6.61 introduces a free Linux version. If you are using Debian, Ubuntu, or one of their derivatives, you can download Mediapurge from the developers' repository using the commands in Listing 1. For a 32-bit system, replace `archive` with `archive-i386`.

For other distributions, download the appropriate tarball for your system [1], unpack it on your hard disk, change to the new `usr/bin/` subdirectory, and call `./mediapurge` from there. If the program prompts you for `canberra-gtk-module` at startup, install the *libcanberra-gtk-module* package via the software manager.

After agreeing to the license, the main window opens (Figure 1) and guides you through several steps to perform your desired task. To get

Listing 1: Downloading Mediapurge

```
$ wget -O - http://archive.peter-ebe.de/keyFile | sudo apt-key add -
$ cd /etc/apt/sources.list.d
$ sudo wget http://archive.peter-ebe.de/peter-ebe-main.list
$ sudo apt-get update
$ sudo apt-get install mediapurge
```

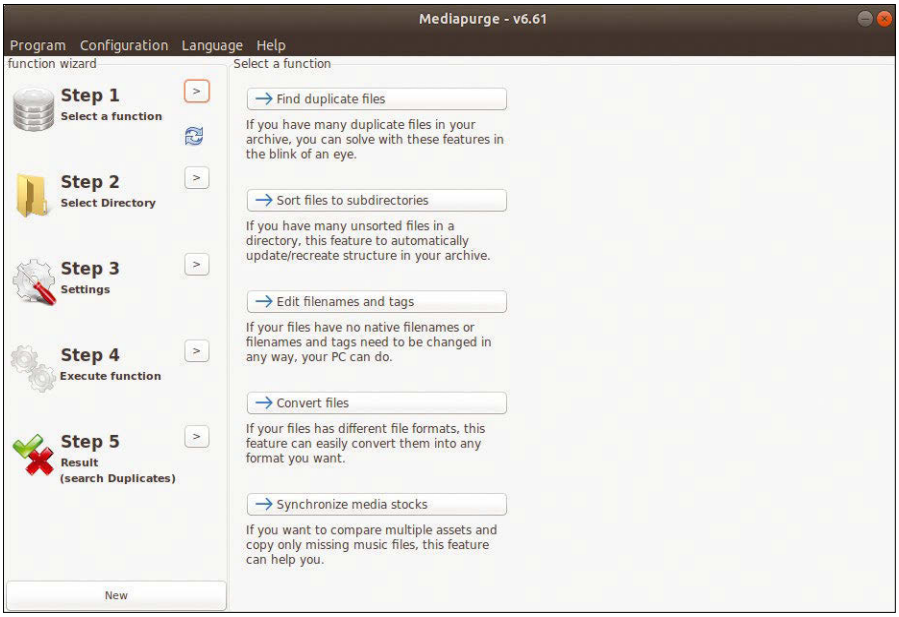


Figure 1: The buttons in the left sidebar guide you through the options for organizing your files.

Photo by Ramon Cordeiro on Unsplash

started, select a function you want to perform, then the files, (Figure 2) change the settings if necessary, and let Mediapurge get to work.

Sorted

To tame a wild and woolly collection of holiday photos or music files, select *Sort files to subdirectories* in Step 1. The tool usually orients itself on the file name or the metadata (which is only possible with audio files). If you are working with audio files, select *Build directory structure from tags*; otherwise, select *Build directory structure from file names*.

Now fill the list with all files you want to sort. To select an entire directory, click on *Add Directory*; to select individual files, click on *Add file selection*. If you accidentally add an entry, it cannot be removed from the list individually. In this case, restart by pressing *New*.

Press *Next* to go to the next step. When sorting the media by file name, the tool is mainly guided by the specified *Separator* (Figure 3). If you have decided to sort by tags, select the desired criteria under *Sort by*. If you choose *Artist*, Mediapurge sorts all tracks by, say, Queen into a single folder. If you have only one track by Queen, but still want to assign it to its own folder, enable *move individual files*.

The software sorts the files directly in the respective directory. If you uncheck *Source directory of media files* and select a new destination directory, all

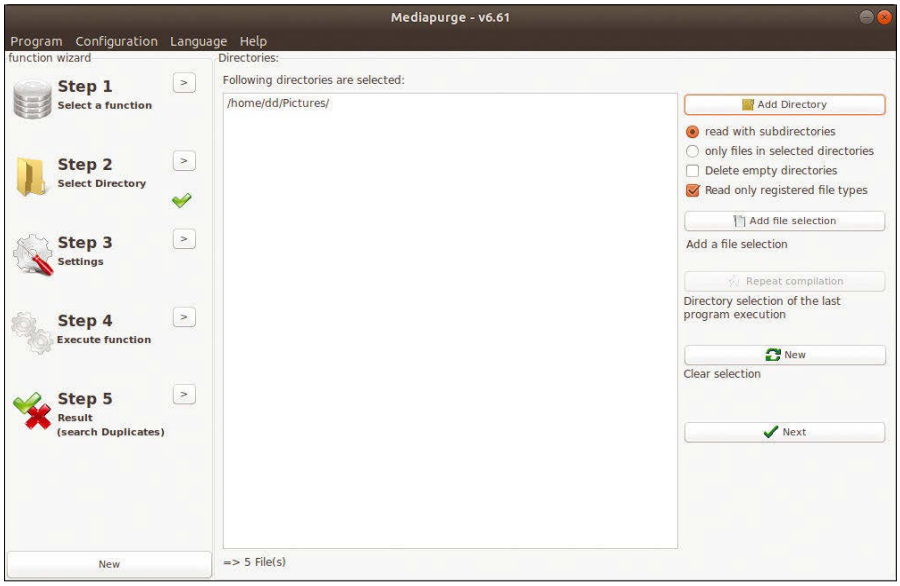


Figure 2: Checking *Read only registered file types* tells Mediapurge to only consider file types that it supports.

sorted files will end up there. If you have specified several sources in the second step, you can bundle your photos in a folder in this way. Click *Start* to begin the process.

Duplicate Hunt

Step 1 also gives you an option to find duplicate files with Mediapurge. In the simplest case, the tool collects some information about each file, such as the tags in MP3 files, and compares them with the data in the other files. If this quick comparison is sufficient for you, select *Duplicates (similar file information)*. Alternatively, you can let Mediapurge compare the contents bit by bit. To do this, select *Duplicates (identical file copies)*.

As a third possibility, the software offers to analyze the files' contents. In the background, it generates hash values using a procedure (which is not described in detail) and then compares the hashes with each other. According to the developers, the process ignores small differences in volume and quality in audio material; for images, it ignores differences in brightness, contrast, and color as well as minor retouching.

To speed up a new duplicate search, the program remembers all fingerprints it creates using FFmpeg. If necessary, you can import this software via the package manager. Then go to *Configuration | Decoder/Encoder settings* and select *Mediapurge apply default settings for FFmpeg*.

In Step 2, select the files you want to process. To create file fingerprints with Mediapurge, uncheck *Read only registered file types*. The software then forwards all files to FFmpeg, which recognizes significantly more file types.

Press *Start* to begin the search for duplicates, which takes you directly to Step 5. After clicking on *Start auto selection*, Mediapurge shows all files that it thinks are duplicates. In the main window, you can press *Delete duplicates* directly or select the button to move the duplicate files to a folder. However, Mediapurge is often wrong. In my tests, it was particularly good at detecting duplicate photos of the same size with very similar motifs. On the other hand, it did not recognize thumbnails and scaled-down versions as duplicates.

Continue manual selection opens a window (Figure 4) where you can manually sort the duplicates. The list shows all the files with the same content. After clicking to select a file from the list, you can proceed to *Delete*. Clicking *Keep* deletes all other files except for the selected file. Select *Next* to scroll to the next duplicate. Clicking *Open* activates a preview.

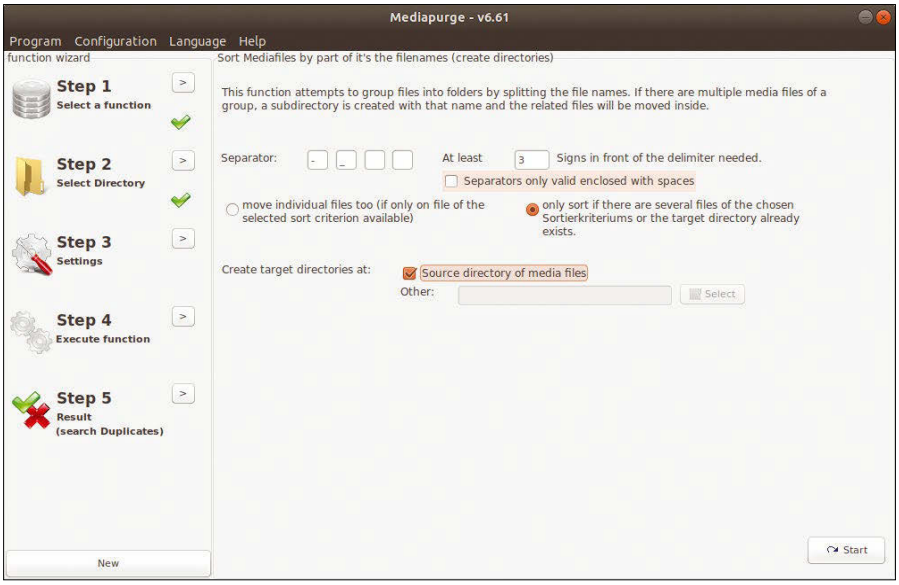


Figure 3: If you had a file named `IMG-2019-06-01.png`, the settings shown here would allow Mediapurge to sort it into a folder named `2019` and into a subdirectory named `06`.

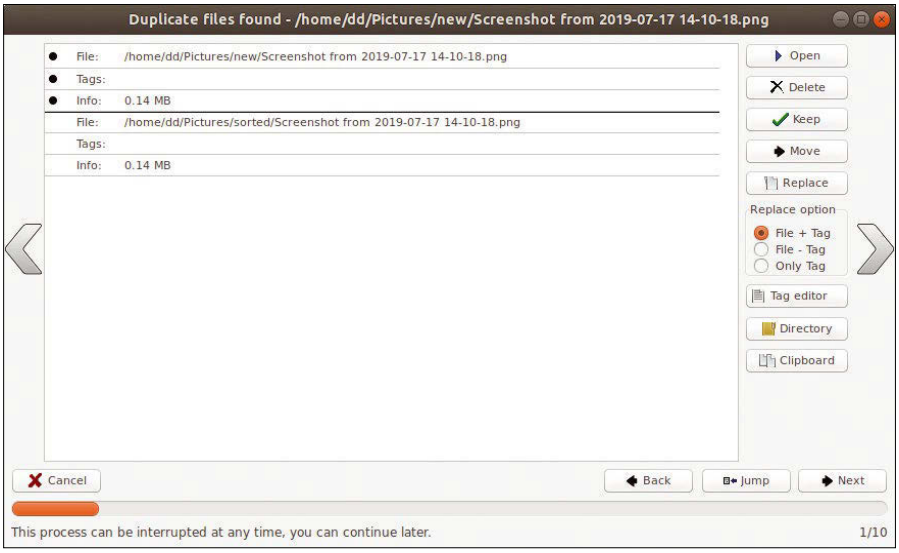


Figure 4: Mediapurge has determined that the content of these two screenshots is the same.

Comparison

Mediapurge can synchronize a collection with a backup on an external hard disk. To do this, you first create a stock file: In the first step, click *Synchronize media stock*, then *Select Directory*, and add the files to be backed up. Then select *Create inventory file* and give the file a meaningful name. If there are already files at the destination, create an inventory file in the same way.

Next to *Inventory 1*, click on *Select* and select the first inventory file. Then click on *Select* next to *Inventory 2* and enter the second inventory file. If there is no file here, click *Empty* and then *Yes*. Whatever the case, press the *Create* button next to *Deficiency 2*. The program creates a list of all files missing from the backup (i.e., *Inventory 2*).

Enter a file name for the list of missing files. You can copy the files found here

by pressing *Open* next to *Copy deficiency*, selecting the list of missing files you just created, then clicking on *Select* next to *Target path*, defining the directory in which you want to store all the missing files, and finally selecting *Copy files* bottom right.

You can unify the media's file names by clicking *Edit filenames and tags* in Step 1 and then *Edit filenames and tags*. Select the files you want to modify, and then point and click to define the pattern for the new file names (Figure 6).

Mediapurge can also convert audio files. Select *Convert files* in Step 1, then select the files to be converted, define the desired *Output format*, and press *Start* to convert.

Conclusions

When it comes to cleaning up large music collections, Mediapurge can be especially helpful because of its ability to convert audio files to another format. While it cannot convert image files, Mediapurge can sort files and give meaningful file names. When it comes to finding duplicates, Mediapurge only finds duplicate photos if the images are very similar. As for synchronizing a collection, it is a painstaking process that involves creating lists. Despite its quirks, Mediapurge can save you time and effort with large file collections by replacing many manual operations with a small number of mouse clicks. ■■■

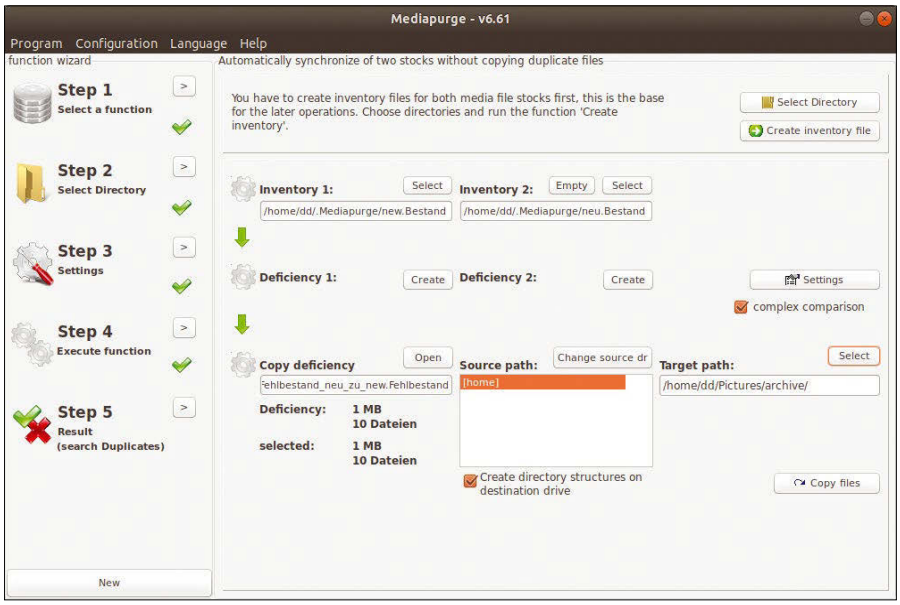


Figure 5: These settings tell Mediapurge to move a photo collection to the `/home/dd/Pictures/archive` directory.

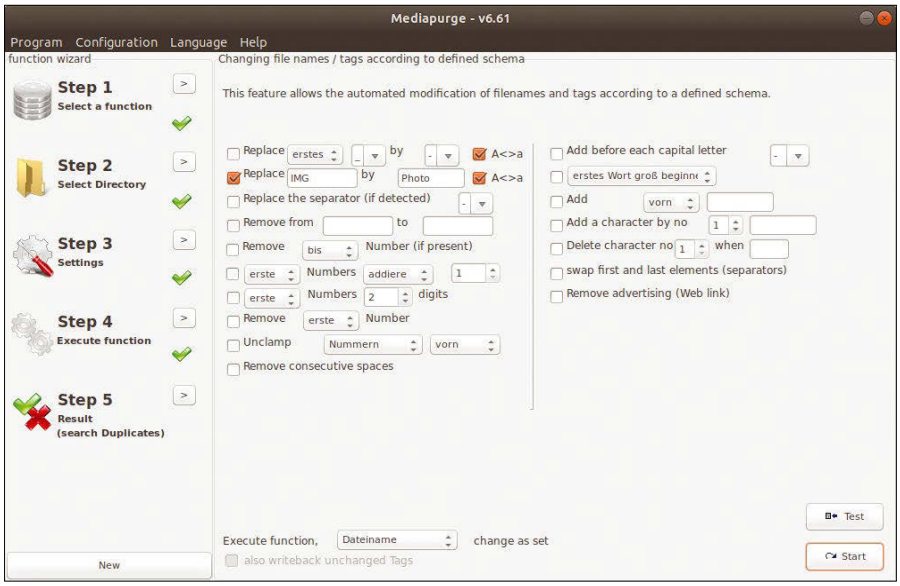


Figure 6: Enabling *Replace* allows Mediapurge to rename files. With these settings, `IMG_2567.JPG` would become `Photo_2567.JPG`.

Info

[1] Mediapurge: *http://www.peter-ebe.de/index_en.php*

Explore Fyne, a GUI framework for Go

Straight to the Point

With the Fyne framework, Go offers an easy-to-use graphical interface for all popular platforms. As a sample application, Mike uses an algorithm to draw arrows onto images. *By Mike Schilli*

My new favorite language, Go, impresses with its image processing routines, manipulating the pixels in an image just as quickly as ordinary numbers or text [1]. I often paint arrows in digital images, either for illustration purposes or to point out something funny in a snapshot, and I wondered how hard it would be to automate this task. Thus far, I have always had to fire up Gimp, select a path with the Path tool, and then place an arrow in the image with an Arrow plugin that I downloaded from somewhere on the Internet. With Go, there's got to be an easier way!

GUI Preferred

Actually, I prefer command-line tools, but sometimes a traditional graphical interface is just more practical, for example, to select a place in a photo where I want the program to draw a red arrow.

The arrow in Figure 1 illustrates, for example, that the tape measure in the photo, which I found while rummaging through my dusty desk drawers, is an age-old tchotchke from the once thriving company Netscape. Remember them? Yes, the browser manufacturer, where I worked for a few years during the 1990s, shortly after I left Munich and escaped to Silicon Valley "for a year" but somehow never found my way back.

This tape measure is thus more than 20 years old and was my antique contribution to the recent 25th anniversary celebrations of the German edition of *Linux Magazine*. I fondly remember the day when a young lady from Netscape's own ergonomics department presented it to me so that I could adjust my desk chair to a height that was gentle on my back and hand tendons. Talk about traveling down memory lane!

Spoiled for Choice

For mouse-driven input on graphical interfaces, Go provides about half a dozen different GUI frameworks, described by

Figure 1: The algorithm has drawn an arrow on the image at the position selected in the application.

Listing 1: hello.go

```
01 package main
02
03 import "fyne.io/fyne/app"
04 import "fyne.io/fyne/widget"
05
06 func main() {
07   app := app.New()
08
09   win := app.NewWindow("Hello World")
10   win.SetContent(widget.NewVBox(
11     widget.NewLabel("Hello World"),
12     widget.NewButton("Quit", func() {
13       app.Quit()
14     }),
15   ))
16
17   win.ShowAndRun()
18 }
```

Lead Image © artqu, 123RF.com

Andrew Williams' excellent book [2] with rewarding examples, and all more or less run across platforms (i.e., on Linux, the Mac, and even Windows, Remember them?) All based on a single codebase, and one of the frameworks is even supposed to run on mobile phones at some point, a real marvel of technology!

Another alternative for the problem solved today would have been the Electron GUI, which runs with JavaScript and made its debut in this column at the end of last year [3].

Williams also published a Go-based GUI package named Fyne [4] that looked very appealing to me aesthetically, so I used it for the arrow software that had popped into my head. A simple "Hello World" program in Fyne is created with just a few lines of Go code, as illustrated by Listing 1 [5]. It sets a text label and a button one below the other and then waits for the user to click the *Exit* button with the mouse. As always in Go, Listing 1 compiles with a two-liner:

```
go mod init hello
go build
```

The new Go module created with `go mod init hello` causes the compiler in the build statement to fetch all the packages not yet installed from the GitHub repositories specified in the code, making them available to the current and future builds requiring them.

The result is a binary file `hello` that doesn't need anything else to run and looks almost exactly the same on every target platform after recompiling. However, in order for Go to find the correct libraries for Ubuntu during compilation, the following packages must be installed:

```
sudo apt-get install libgl1-mesa-dev ➥
  xorg-dev
```

On the Mac, the whole thing works without further ado.

Hello World

Listing 1 first creates an app, and then its one and only window. The `SetContent()` method adds a `VBox` widget to the window in line 10, which in turn arranges two widgets stacked on top of each other: The first one is a label reading "Hello World," and the second is a button that invites users to exit the program with `Quit`.

The click action assigned to the button is accepted by the `NewButton()` constructor in line 12 as a callback function. The callback in this case just invokes `app.Quit()` and removes the GUI, leaving the user with a clean shell prompt.

Now that all widgets and their potential interactions are defined, it is up to line 17 to jump into the event loop with `ShowAndRun()`, which will be waiting for user input while refreshing the UI continuously.

Now, let's move on to something more challenging – how to pop up a GUI that draws an arrow on a displayed

Listing 2: picker.go

```
01 package main
02
03 import (
04   "flag"
05   "fmt"
06   "fyne.io/fyne"
07   "fyne.io/fyne/app"
08   "fyne.io/fyne/canvas"
09   "fyne.io/fyne/widget"
10   "image/jpeg"
11   "log"
12   "os"
13 )
14
15 type clickImage struct {
16   widget.Box
17   image *canvas.Image
18   filename    string
19 }
20
21 func (ci *clickImage) Tapped(
22     pe *fyne.PointEvent) {
23   imgAddArrow(ci.filename,
24             pe.Position.X, pe.Position.Y)
25   canvas.Refresh(ci)
26 }
27
28 func (ci *clickImage) TappedSecondary(
29     *fyne.PointEvent) {
30   // empty, but required for Tapped to work
31 }
32
33 func main() {
34   flag.Parse()
35   flag.Usage = func() {
36     fmt.Printf("usage: %s imgfile\n",
37       os.Args[0])
38   }
39
40   if len(flag.Args()) != 1 {
41     flag.Usage()
42     os.Exit(2)
43   }
44
45   file := flag.Arg(0)
46
47   os.Setenv("FYNE_SCALE", ".25")
48   win := app.New().NewWindow("Pick Arrow")
49   img := canvas.NewImageFromFile(file)
50
51   width, height := imgDim(file)
52   clickimg := clickImage{image: img,
53                          filename: file}
54   clickimg.image.SetMinSize(
55       fyne.NewSize(width, height))
56   clickimg.Append(clickimg.image)
57   win.SetContent(&clickimg)
58   win.Resize(fyne.NewSize(width, height))
59   win.ShowAndRun()
60 }
61
62 func imgDim(file string) (int, int) {
63   src, err := os.Open(file)
64   if err != nil {
65     log.Fatalf("Can't read %s", file)
66   }
67   defer src.Close()
68
69   jimg, err := jpeg.Decode(src)
70   if err != nil {
71     log.Fatalf("Can't decode %s: %s",
72       file, err)
73   }
74
75   bounds := jimg.Bounds()
76   return (bounds.Max.X - bounds.Min.X),
77          (bounds.Max.Y - bounds.Min.Y)
78 }
```

photo at a location picked by the user. Listing 2 shows the UI application that opens a window and uses `NewImageFromFile()` to prepare a JPG file specified by the user on the command line for loading it later on.

Do-It-Yourself Widget

Curiously, Fyne hides the dimensions of the loaded photo from the program, so the `imgDim()` function defined as of line 62 simply loads the JPG file again from disk using `Decode()` and then calls `Bounds()` to fetch the coordinates of the image's upper left and lower right corners. These x/y values in turn are converted by line 76 into the width and height of the image by simple arithmetic. The resulting coordinates pair is then returned to the calling main program. The latter uses this information to adjust the size of the application window to fit the image exactly.

As before with "Hello World," Listing 2 starts an event loop with `ShowAndRun()` in line 59, which processes the user's interactions with the GUI – in this case intercepting mouse clicks and executing predefined actions on them.

In the program at hand, I want the displayed image widget to notice where the user clicked with the mouse, in order to draw an arrow there. However, the image loaded and displayed by Fyne is not a real widget, capable of processing mouse clicks. Listing 2 thus resorts to a hack that I learned from a helpful contributor to the Slack #fyne channel. Line 15 defines a structure that, thanks to the `widget.Box` element, inherits its widget properties from the general purpose `Box` widget in the Fyne catalog, including its capability of processing mouse clicks in its vicinity. On top of this, the structure saves the image object in the `image` attribute and the name of the loaded JPG file for later use in `filename`.

Lines 21 and 28 now define the methods `Tapped()` and `TappedSecondary()` for this new DIY widget. Surprisingly, Fyne won't be satisfied with a measly `Tapped` event to register mouse clicks; it also insists the user define a secondary pointer events handler for Multi-Touch, which is not used in this widget at all. The program leaves `TappedSecondary()` blank,

Listing 3: arrow-draw.go

```
01 package main
02
03 import (
04   "image"
05   "image/color"
06   "image/draw"
07   "image/jpeg"
08   "log"
09   "os"
10 )
11
12 func imgAddArrow(file string, x, y int) {
13   src, err := os.Open(file)
14   if err != nil {
15     log.Fatalf("Can't read %s", file)
16   }
17   defer src.Close()
18
19   jimg, err := jpeg.Decode(src)
20   if err != nil {
21     log.Fatalf("Can't decode %s: %s",
22       file, err)
23   }
24
25   bounds := jimg.Bounds()
26   dimg := image.NewRGBA(bounds)
27   draw.Draw(dimg, dimg.Bounds(), jimg,
28     bounds.Min, draw.Src)
29   arrowDraw(dimg, image.Point{X: x, Y: y})
30
31   dstFileName := file
32   dstFile, err := os.OpenFile(dstFileName,
33     os.O_RDWR|os.O_CREATE, 0644)
34   if err != nil {
35     log.Fatalf("Can't open output")
36   }
37
38   jpeg.Encode(dstFile, dimg,
39     &jpeg.Options{Quality: 80})
40   dstFile.Close()
41 }
42
43 func arrowDraw(dimg draw.Image,
44                start image.Point) {
45   length := 300
46   width := 20
47   tiplength := 80
48   tipwidth := 90
49
50   stem := image.Rectangle{
51     Min: image.Point{X: start.X,
52                      Y: start.Y},
53     Max: image.Point{X: start.X + length,
54                      Y: start.Y + width},
55   }
56   rectDraw(dimg, stem)
57
58   triDraw(dimg,
59     image.Point{X: start.X + length,
60       Y: start.Y + width/2 - tipwidth/2},
61     image.Point{X: start.X + length,
62       Y: start.Y + width/2 + tipwidth/2},
63     image.Point{
64       X: start.X + length + tiplength,
65       Y: start.Y + tipwidth/2},
66   )
67 }
68
69 func rectDraw(dimg draw.Image,
70               bounds image.Rectangle) {
71   red := color.RGBA{255, 0, 0, 255}
72   draw.Draw(dimg, bounds,
73     &image.Uniform{red},
74     bounds.Min, draw.Src)
75 }
76
77 func triDraw(dimg draw.Image,
78              t1, t2, t3 image.Point) {
79   ymiddle := t1.Y + (t2.Y-t1.Y)/2
80
81   for x := t1.X; x < t3.X; x++ {
82     height := int(float64(t2.Y-t1.Y) *
83       (float64(t3.X-x) /
84        float64(t3.X-t2.X)))
85     rect := image.Rectangle{
86       Min: image.Point{X: x,
87         Y: ymiddle - height/2},
88       Max: image.Point{X: x + 1,
89         Y: ymiddle + height/2}}
90
91     rectDraw(dimg, rect)
92   }
93 }
```

but it has to be there, because otherwise `Tapped()` won't work either.

With mouse clicks, Listing 2 will therefore jump to the callback defined in line 23 for the `Tapped` event, which invokes the `imgAddArrow()` function from Listing 3. `imgAddArrow()` draws the arrow into the image and saves the modified file. A subsequent `Refresh()` on the widget loads the modified photo from disk again and displays it. The user only notices that they clicked on a point in the image and that a horizontal red arrow appears there as if by magic. Line 56 in Listing 2 uses `Append()` to append the image object to the objects managed by the new DIY `clickImage` widget. This is done using the image object pointer stored in the `clickImage` structure in line 17; the image data resides in the `struct`'s `image` element.

Fyne is designed to dynamically adapt its appearance to the display being used like a T-1000 Terminator. However, this conflicts with some of the requirements of the arrow software, mainly to display the window in such a way that the image pixels are shown 1:1, to make sure mouse click coordinates can easily be converted into image pixels.

If the call to `SetMinSize()` were left out in line 54 in Listing 2, Fyne would shrink the image beyond recognition. Line 58 makes sure that the application window assumes exactly the same dimensions as the image. Thus, nothing is compressed, the width to height ratio remains overall.

Line 47 with the `FYNE_SCALE` environment variable reduces the size of the application window by a factor of four (to 0.25) so that even a relatively large image, for example from a modern mobile phone, will fit on a reasonably sized PC screen.

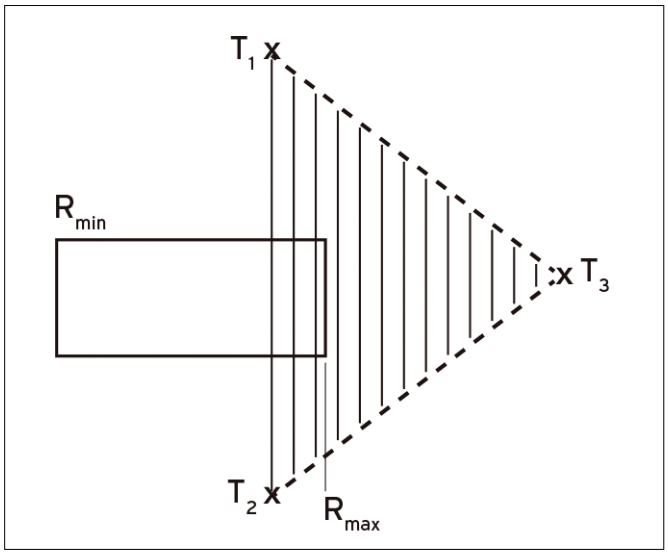


Figure 2: Simple algorithm for a filled arrow.

Anatomy of an Arrow

How does the algorithm paint the arrow? The arrow's thin trunk can be easily painted as a horizontal rectangle. The tip of the arrow pointing to the right is a triangle with the coordinates T1, T2, and T3 (each stated as x- and y-values), which must also be filled with red color (Figure 2). It's not as trivial as you'd think at first glance. However, if we dismantle the triangular shape into narrow vertical rectangles of descending height, things fall into place. The rectangles' height is at a maximum for x values near T1 and T2, and decreases linearly until it reaches zero at T3.

The `imgAddArrow()` function from line 12 of Listing 3 implements the necessary steps. Line 19 decodes the loaded JPG image and calls the internal `arrowDraw()` function, which draws the arrow, converts the image back into JPG format and saves it under the same name on disk.

The arrow is painted with a slender horizontal rectangle and a lateral triangle at its right end. The tip points to the right. The numerical values in lines 45 to 48 define the shape and size of the arrow. Graphics libraries like `image/draw` from Go's core package can paint rectangles of any kind and even fill them in with an appealing color, so that's easy.

One caveat: They don't specify a rectangle's corner coordinates as four x/y values, but – as the method `Bounds()` reveals – with only two: the `Min` and the `Max` point, that is, the upper left corner and the lower right corner, since x coordinates run from left to right and y coordinates from top to bottom.

Both bounds points are available as x/y pairs. The logic between lines 51 and 54 calculates the `Bounds()` values of the desired rectangle, given the x/y coordinates of a starting point, and the desired length and thickness of the arrow's trunk.

The `rectDraw()` function from line 69 draws a specified rectangle in red on the image using Go's `images/draw` library. The triangular tip of the arrow comes courtesy of `triDraw()` in line 77. Besides the handle on the draw image, it accepts the coordinates of the points T1, T2, and T3 in Figure 2. The formula for the height of the triangle at different x-values is determined by line 82. It divides the distance between the current x-value and the end point T3 by the total x-spacing between T2 and T3, thus reporting the maximum height of the triangle near T2, and a height of zero pixels near T3. These many triangular parts in turn consist of narrow vertical rectangles with a width of one pixel. The whole thing looks surprisingly convincing for such a simple algorithm.

Listings 2 and 3 divide the program into two parts for the sake of clarity, but both define the package `main`. This is why

```
go build picker.go arrow-draw.go
```

generates a binary `picker` that starts the user interface, displaying the specified image. It lets the user select a point with the mouse and then quickly draws an arrow there. None of this is rocket science; any algorithm ultimately boils down to a few simple, reproducible steps. ■■■

Info

[1] "Programming Snapshot – Go" by Mike Schilli, *Linux Magazine*, Issue 221, April 2019, pg. 46-49

[2] Williams, Andrew. *Hands-On GUI Application Development in Go,* Packt Publishing, 2019

[3] "Programming Snapshot – Electron" by Mike Schilli, *Linux Magazine*, Issue 216, November 2018, pg. 46-50, *http://www.linux-magazine.com/Issues/2018/216/Clever-Sampling/*

[4] Fyne: *https://fyne.io*

[5] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/229/*

Current status of the Oil shell

Slippery Shell

With its innovative scripting language, Oil, the Bash-compatible Oil shell aims to make life easier for script developers. *By Tim Schürmann*

Developer Andy Chu is currently working on two construction sites at the same time: the Oil shell (OSH) and the Oil language. His work on OSH is already quite advanced, but, as of version 0.6.pre20, it only supports a subset of the Bash constructs. The Oil language, on the other hand, is still a work in progress.

OSH

A shell is a small program that uses text commands to control the system. Many shells also let you write scripts to automate processes. One of the best known and most common shells is the Bourne-Again Shell (Bash) [1]. The Oil shell is a Unix shell that is compatible with Bash.

According to Chu, OSH is already capable of processing the *abuild* shell script, which is over 2,500 lines long and builds the packages for the Alpine Linux distribution. OSH can also set up an Ubuntu base system via `debootstrap` and `chroot` into it.

Unlike other shells, OSH shell scripts not only evaluate command by command, but fully load some command sequences, including mathematical expressions; this makes it possible to identify bugs at an earlier stage. Furthermore, OSH offers automatic completion at the command line (Figure 1).

Test Drive

To try out OSH, download the latest release from the Oil project's homepage [2] and unpack the archive. To build the shell, you need a C compiler, `make`, and the developer package for the Readline library. If these components are missing on your system, you can install them on Ubuntu using the command shown in line 1 of Listing 1. Then change to the Oil shell's source directory, compile the program, and set it up on the system (lines 3 to 5). You can use the Oil shell much like Bash and apply it to shell scripts:

```
$ osh -c 'echo hello world!'
```

The call to `osh` is only a symbolic link to the actual program, which goes by the name of `oil.vm`. If you start it via `osh`, the shell will also log on under this name.

Chu wrote OSH in Python. He chose this language, because it helped him obtain results quickly. The disadvantage is a significantly lower speed com-

Figure 1: If you press Tab at this point in OSH, the shell automatically completes `$HOSTNAME`.

Listing 1: Installing OSH

```
01 $ sudo apt install build-essential libreadline-dev
02 [...]
03 $ ./configure
04 $ make
05 $ sudo ./install
```

Photo by Javardh on Unsplash

pared to other shells. OSH runs in a virtual machine (VM), the OVM. This is a modified version of the CPython VM, Python's official reference implementation.

The Python 2 VM is currently used, but its support will expire in January 2020. The VM's code is now generated by the OPy bytecode compiler. The source code for the current OSH development version is available from GitHub [3].

The Oil Language

Parallel to OSH, Chu is developing the Oil language (Oil for short), a completely new scripting language, which is intended to eliminate some of the inconveniences of previous Unix shells and thus make work easier for shell script developers. For example, the expression `x=1` leads to the same result as `x = 1`. Furthermore, Oil's goal is to be easier to learn, write, and debug; do more than the Bash can do; and have a more consistent grammar.

According to Chu, Oil's specification and structure is largely fixed, but the documentation is only rudimentary. A shell of this kind would not execute every command directly one after the other, but first load in a script completely and store its structure in the main memory. Thanks to this static parsing, the shell is able to detect typos and syntax errors at an early stage.

The shell also stores the script's structure in RAM in a special data structure known as a syntax tree. OSH constructs this syntax tree in such a smart way that you can generate the original script from it. With a lossless syntax tree like this, you should be able to detect errors more easily.

Oil offers shell-like functions that are defined with the keyword `proc`, as well as additional conventional functions introduced by the `func` keyword. The latter are similar to the Python and JavaScript functions and can accept and return more complex data structures. Chu wants to improve the shell syntax for the manipulation of strings.

Oil encloses code blocks in curly brackets, so the following statement

```
if ... then ... fi
```

converts to a shorter counterpart

Listing 2: The Original Bash Script

```
make_hdb()
{
  # Some distros don't put /sbin:/usr/sbin in the $PATH for non-root users.
  if [ -z "$(which  mke2fs)" ] || [ -z "$(which tune2fs)" ]
  then
    export PATH=/sbin:/usr/sbin:$PATH
  fi

  truncate -s ${HDBMEGS}m "$HDB" &&
  mke2fs -q -b 1024 -F "$HDB" -i 4096 &&
  tune2fs -j -c 0 -i 0 "$HDB"
```

Listing 3: The Converted Oil Script

```
proc make_hdb
{
  # Some distros don't put /sbin:/usr/sbin in the $PATH for non-root users.
  if test -z $[which  mke2fs] || test -z $[which tune2fs]
  {
    export PATH = "/sbin:/usr/sbin:$PATH"
  }

  truncate -s $(HDBMEGS)m $HDB &&
  mke2fs -q -b 1024 -F $HDB -i 4096 &&
  tune2fs -j -c 0 -i 0 $HDB

  test $Status -ne 0 && exit 1
```

```
if ... { ... }
```

Oil also supports arrays familiar from other programming languages that use square brackets for initialization:

```
Hello = ['Hello' 'World']
```

Bash scripts should be easy to translate into Oil. Chu is already working on a suitable translator, which will convert OSH scripts into corresponding Oil code. At present, however, this only exists as a proof of concept prototype.

An example of the results from the OSH-to-Oil converter can be found in the Oil project's blog [4]. Listing 2 shows an original Bash script before conversion, and Listing 3 shows the Oil language variant generated from it. The parser is also written in Python; its source code can be found on GitHub in the OSH source code under `tools/` [5].

Conclusions

In its current state, the Oil shell only supports a subset of Bash's capabilities, making the more mature Bash shell, which is preinstalled on most Unix-like systems, a more practical choice. In addition, the Oil language's properties sound interesting, but a working implementation is currently not available.

If you want to find out more about the Oil project, you will need to read numerous blog posts that contain virtually no concrete information about the Oil shell, but instead look into the reasons for its development. Given its current state of development, the Oil shell will take a few more years before it is ready for practical use. ■■■

Info

[1] Bash: *https://en.wikipedia.org/wiki/Bash_(Unix_shell)*

[2] The Oil project: *https://www.oilshell.org*

[3] OSH on GitHub: *https://github.com/oilshell/oil*

[4] OSH-to-Oil converter: *http://www.oilshell.org/blog/snapshots/2017-02/aboriginal/osh-to-oil.html#sources/toys/make-hdb.sh*

[5] Oil parser on GitHub: *https://github.com/oilshell/oil/tree/master/tools*

MakerSpace

Multiple operating systems on a single memory card

Versatility

BerryBoot turns your Raspberry Pi into a multiboot system, with several operating systems on a single card. *By Erik Bärwaldt*

The Raspberry Pi is suitable for a wide variety of application scenarios; accordingly, a wide variety of operating systems are available for installation. Whether you want to use the small computer as a media center, a gaming console, or a solid all-rounder, you will find the right system for every application.

However, the most common operating systems are each installed on their own SD memory card; if you want to boot another system, you have to change the card. You can use your Rasp Pi in a far more elegant way with the BerryBoot [1] boot manager. BerryBoot allows the installation of multiple operating systems on a single SD memory card and supports a convenient selection dialog from a beautifully designed bootloader.

In the Cards

In contrast to conventional images with one operating system that is then copied to a memory card, you can download a BerryBoot ZIP archive with a minimal system from the project's website. Two different archives are available for download: The larger archive (~60MB) is suitable for all Rasp Pi models; the smaller archive (~40MB) is intended exclusively for models from the Rasp Pi 2 (RPi2) upward [2]. After downloading, unpack the archive directly on a memory card that has been formatted with the VFAT filesystem. Next, insert the memory card into your Rasp Pi and power it up. BerryBoot will now take you to a graphical configuration dialog.

The first window (Figure 1) prompts for the video settings, network connection, and system locale. The RPi2 provides immediate wired access to the Internet; the RPi3 requires you to press a radio button to the left of the matching option if you want to connect over WiFi. The locale is then largely adapted automatically. You might need to change your keyboard layout from a drop-down list.

In the second dialog, you need to provide authentication data if you want to access the Internet over WiFi. The third dialog is about preparing the storage medium. Here, you can select whether the Rasp Pi's internal microSD memory card or a connected external storage medium, such as a network-attached storage (NAS) system or a USB memory stick, will be used as the target medium for the operating systems. The default setting is the internal memory card, because an external drive is not always connected to a Rasp Pi system. In this dialog, you also choose the filesystem for the target medium, and you can also enable encryption by checking a box. The target medium is then formatted.

In the fourth and last dialog, select the desired operating system from

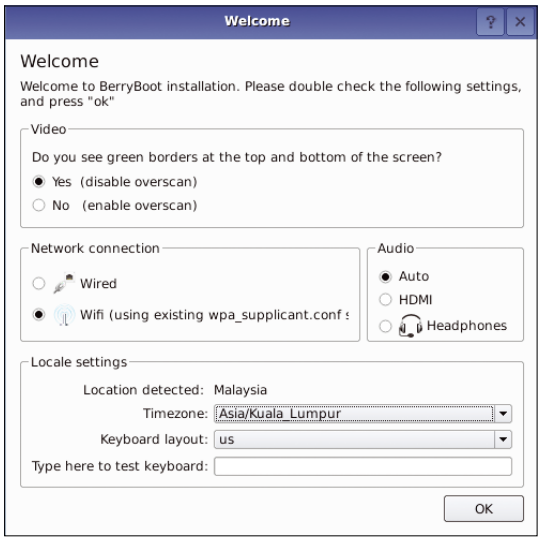


Figure 1: The installation dialog prompts for a few options.

among the selection lists presented as horizontal tabs at the top of the window (Figure 2). The default list includes Raspbian, the OpenELEC media center, Ubuntu desktop with the MATE desktop environment, Ubuntu server, and the BBMC media center by BerryBoot. A thin client is available from BerryBoot, as well, that is intended for connecting to an Ubuntu server. This thin client, known as BerryTerminal, is explicitly intended for use in learning environments in schools.

Selecting the operating system causes it to be downloaded from the web and installed on the target medium. The system then prompts you to reboot.

First Start

After a reboot, the BerryBoot boot manager appears in an attractively designed GUI. The installed operating system is then activated with a preset wait time of 10 seconds. Because only one operating system can be selected during installation, you now have the option of adding more systems to your drive. To do this, press the *Edit menu* button in the selection window and then click *Add OS* in the upper left corner of the horizontal buttonbar in the now open BerryBoot menu window. The routine establishes access to the Internet and opens the selection window for the operating systems chosen for installation.

If you set up your Rasp Pi to access the Internet over WiFi during the initial installation on the SD card, you might need to re-establish the connection in some cases after extending the boot menu. After a reboot, BerryBoot only wants to use wired Internet access – if this is not available, it displays a small dialog in the middle of the start window, telling you that you do not have an Internet connection. After clicking in this window and selecting the desired WiFi connection, you enable wireless access.

As soon as you see the selection window for the other operating systems, you can select another Linux derivative for installation. After pressing the *OK* button at the bottom, your chosen system is installed on the SD card. Bottom left in the settings window, the routine also shows how much free storage space is left on the memory card. While downloading the new system, a progress bar is displayed in a separate pop-up window. After completing the installation, the new operating system appears in the menu editor (Figure 3).

Restart the system by clicking the *Exit* icon in the buttonbar. In the boot menu that appears, the newly installed operating system is listed, and you can call it in the usual way in GRUB. In the case of the Ubuntu desktop system, the basic installation is then performed by setting the locale and creating a user account. In our lab, we discovered that the RPi3's WiFi hardware is not supported, so you should configure wired Internet access for the system on Ubuntu 16.04 LTS. Another drawback was the alpha version of Puppy Linux, for which the X server failed to launch, and the screen remained black.

Once you have installed several

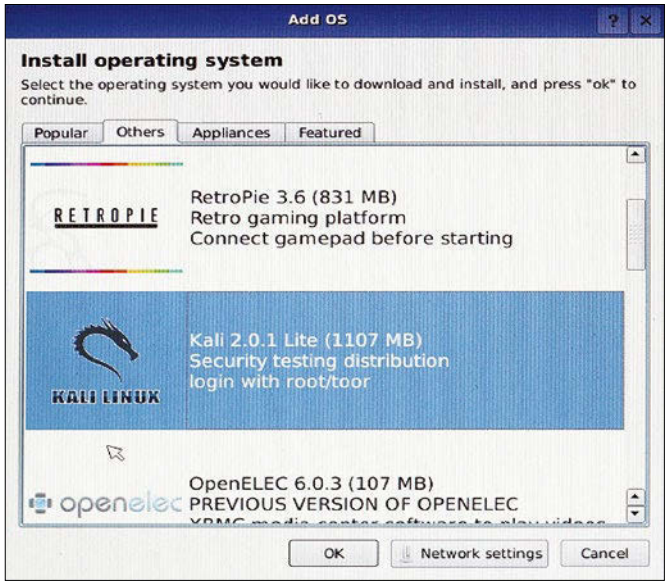


Figure 2: Operating systems that can be installed are presented in tabs.

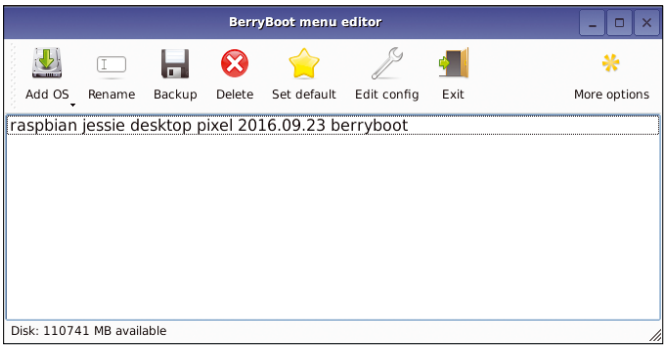


Figure 3: The menu editor lets you configure the system with a few mouse clicks.

operating systems on the memory card, you can also select the one to be booted by default each time the Rasp Pi is powered on. Simply click on the desired system in the configuration window and then press the *Set default* button. An asterisk is now displayed to the left of this operating system, which is automatically booted after 10 seconds the next time you power on the computer.

Backup

You can save the BerryBoot configuration and the contents of the SD memory card in various ways in the configuration window. Pressing the *Backup* icon opens a window with extensive backup and restore options. To back up content, you can use the radio button to specify whether you want to duplicate the memory card or back up individual images; you can save only an original image downloaded off the web, or configured images and their associated data. You can save this content on a USB memory stick.

The dialog also lets you exclude individual files from the backup and compress the data. You can specify the files to be excluded individually in an input area. In the same window, you can also completely duplicate (clone) the memory card, but in this case, an external SD memory card reader equipped with a similar SD card must be connected to the Rasp Pi. The window also allows you to reconstruct an image from an external USB memory stick, whether it is an original image off the web or one you have already configured. Additionally, you can use the *More options* | *Clone* dialog to create an identical full backup of your BerryBoot memory card; the routine also supports cloning from the original image or an individually configured image.

Fail

The BerryBoot developers have built a serious vulnerability into the configuration dialog of the bootloader. This collection of configuration files, which can be accessed from the *Edit config* icon, is listed in the *wpa_supplicant.conf* tab in plain text, so that unauthorized third parties can gain access to the WiFi network with the WPA2 key that is displayed in the clear. If different WiFi accounts are used, however, only the last one is listed in the configuration file, so that only the authentication data of one WiFi network appears in the file.

You can eliminate this security risk by selecting *Edit menu* in the main window to open the configuration dialog and selecting *More options*, where you press *Set password* in the new buttonbar that opens. A password is then entered and verified in an overlapping dialog after you have checked the *Protect access to settings with password* box. After pressing *OK*, you are then prompted for the password the next time the settings dialogs are called. In this way, you can protect all the options against unauthorized access.

Maintenance

BerryBoot also lets you repair damaged filesystems on the SD card and reset an operating system to the original image. You can access both functions under *Repair file system* | *Reset OS* | *More options*. Whereas the filesystem repair preserves the original operating system you used with all its settings and only fixes inconsistencies in the filesystem, selecting *Reset OS* enables the original image from the web. In this case, your personal configuration settings are lost.

Conclusions

BerryBoot is a mature and functional boot manager that is easy to use without training. Although the software is stable, it does not yet offer all operating systems available for the Rasp Pi. The only drawbacks are the security issue with storing WPA2 keys in plain text and the difficulty in finding the function for password protecting the system. BerryBoot is the tool of choice for people who want to choose from several Rasp Pi operating systems in parallel on a single large memory card. ■■■

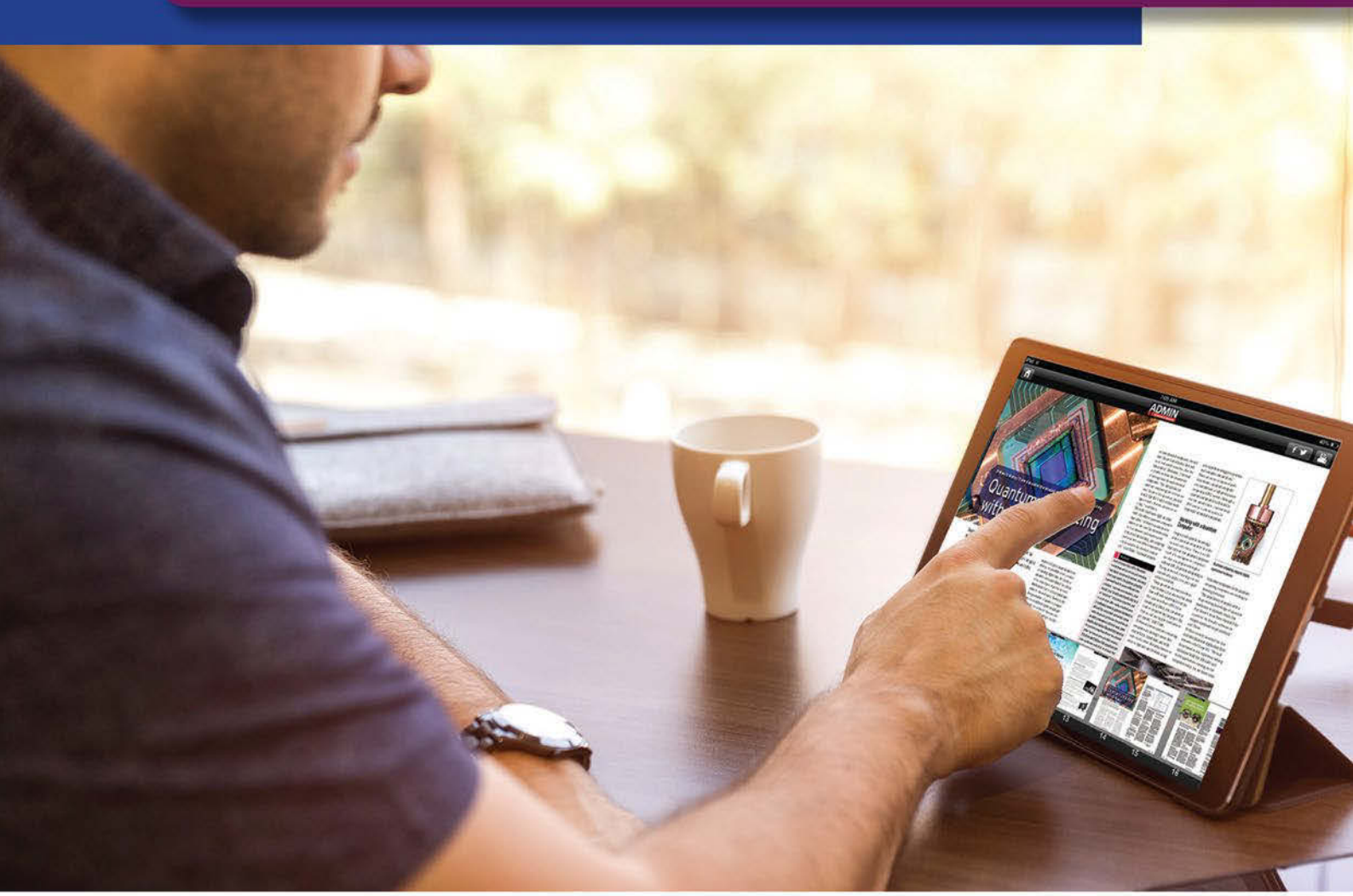
Info

[1] Project website: *https://www.berryterminal.com/doku.php/berryboot*

[2] Download: *https://sourceforge.net/projects/berryboot/files/*

MOBILE USERS

search for us today at your digital newsstand!



Only a swipe away!

Download our convenient digital editions for your iPad, iPhone, or Android device.

Visit our apps page for more information:
shop.linuxnewmedia.com/us/apps

MakerSpace

The Treasure Macropad Type-9

At Your Fingertips

This DIY, programmable input device is a compact companion to your keyboard, with nine keys and 16 layers that can be customized for different applications and games. *By Bruce Byfield*

From keyboards to graphic tablets, modern input devices are sporting programmable keys. These keys are not only convenient, but help to reduce repetitive stress injuries by keeping the fingers on the keyboard. Yet there is something to be said for a single compact programmable device rather than several scattered across your workspace. That is the advantage of the Treasure Macropad Type-9 (TMT9) [1], the first product of Eric Boudo's new company (Figure 1). Measuring a little over 2x2 inches, the TMT9 has only nine keys, but includes 16 different layers, for a total of 144 programmable keys altogether. It's a compact little device, although for Linux users it requires a certain amount of work and patience, since only graphical setups for Windows and macOS are available.

No matter what your operating system, the TMT9 requires some DIY adjustments, especially since the company is not set up for technical support. Fortunately, ample help is available on the Quantum Mechanical Keyboard (QMK) website [2], although configuration takes

Figure 1: The TMT9 ships with some assembly required.

Figure 2: The TMT9 ships without keycaps, providing another opportunity to customize.

several steps. In addition, units ship unassembled, although no soldering is required. Nor do units ship with keycaps for the Cherry MX mechanical key switches, which have to be ordered separately from a third party [3] (Figure 2). Also, while a newer model that supports USB C is due out soon – and may be available by the time you read this article – the TMT9 requires a USB 2.0 A-Male to Mini-B cable, which is not easy to find in computer stores these days. In the end, I had to order one from Amazon.

```
root@nanday:/home/bb/qmk# git clone --recurse-submodules https://github.com/qmk/qmk_firmware.git
Cloning into 'qmk_firmware'...
remote: Enumerating objects: 182392, done.
remote: Total 182392 (delta 0), reused 0 (delta 0), pack-reused 182392
Receiving objects: 100% (182392/182392), 132.63 MiB | 4.21 MiB/s, done.
Resolving deltas: 100% (117111/117111), done.
Submodule 'lib/chibios' (https://github.com/qmk/ChibiOS) registered for path 'lib/chibios'
Submodule 'lib/chibios-contrib' (https://github.com/qmk/ChibiOS-Contrib) registered for path 'lib/chibios-contrib'
Submodule 'lib/googletest' (https://github.com/google/googletest) registered for path 'lib/googletest'
Submodule 'lib/lufa' (https://github.com/qmk/lufa) registered for path 'lib/lufa'
Submodule 'lib/ugfx' (https://github.com/qmk/uGFX) registered for path 'lib/ugfx'
Cloning into '/home/bb/qmk/qmk_firmware/lib/chibios'...
remote: Enumerating objects: 156956, done.
Receiving objects:  36% (56929/156956), 80.18 MiB | 5.04 MiB/s
```

Figure 3: The first step in configuration is to clone the QMK repository.

```
bb@nanday:~/qmk_firmware$ util/qmk_install.sh
[sudo] password for bb:
Get:1 http://security.debian.org stretch/updates InRelease [94.3 kB]
Get:2 http://security.debian.org stretch/updates/main Sources [213 kB]
Get:3 http://security.debian.org stretch/updates/main amd64 Packages [506 kB]
Get:4 http://ftp.debian.org/debian stretch-backports InRelease [91.8 kB]
Ign:5 http://ftp.us.debian.org/debian stretch InRelease
Get:6 http://ftp.us.debian.org/debian stretch-updates InRelease [91.0 kB]
Hit:7 http://ftp.us.debian.org/debian stretch Release
Get:9 http://ftp.debian.org/debian stretch-backports/main amd64 Packages.diff/Index [7.8 kB]
Get:10 http://ftp.debian.org/debian stretch-backports/main Translation-en.diff/Index [27.8 kB]
```

Figure 4: The `qmk_install.sh` script installs and updates the necessary packages.

Configuring QMK for Linux

However, for Linux users, the challenge is just starting. The most common tools for flashing firmware, the Arduino IDE and the `avrdude` command, will not work. The TMT9 uses the Atmel DFU bootloader, which neither of the two usual tools supports. However, this fact is obscured by `avrdude`'s man page being out of date, which can easily lead the inexperienced – like me – on a false path.

Instead, what you need is to install QMK, an open source toolset specifically designed for keyboard input devices that must be installed before creating firmware. The instructions are for Debian or one of its derivatives. With `git` and AVR-GCC installed, create a clone of the QMK repository (Figure 3):

```
git clone --recurse-submodules
https://github.com/qmk/qmk_firmware.git
cd qmk_firmware
```

The repository includes a number of keyboards that use QMK that will help you build firmware. Continue by running this script to finish setting up the repository:

```
util/qmk_install.sh
```

The script is set up for Debian and its derivatives, and installs or updates all the necessary packages (Figure 4). However, be aware that the script mistakenly looks for *pip3*, when the package required is *python3-pip*. Installing *python3-pip* will also pull in a lot of dependencies, but it is necessary for successful setup.

You can check that the build environment is sound by running:

```
make treasure/type9:default:
```

```
avr-gcc (GCC) 4.9.2
Copyright (C) 2014 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Size before:
   text    data     bss     dec     hex filename
      0   25148       0   25148    623c .build/treasure_type9_default.hex

Compiling: tmk_core/common/command.c
                [OK]
Linking: .build/treasure_type9_default.elf
                [OK]
Creating load file for flashing: .build/treasure_type9_default.hex
                [OK]
Copying treasure_type9_default.hex to qmk_firmware folder
                [OK]
Checking file size of treasure_type9_default.hex
                [OK]
 * The firmware size is fine - 25148/28672 (87%, 3524 bytes free)
root@nanday:/home/bb/qmk_firmware#
```

Figure 5: If the environment test gives no error message and pressing a key results in a number from *0* to *8*, then the default firmware was uploaded to the device.

This command gives the path to the firmware tools for a default piece of software, followed by the name of the firmware version after the colon. You may need to reset the bootloader by poking the wrench that comes with the device into the hole in the back of the device. If no error message is given, try tapping one of the keys in a text editor. The result will be a number from *0* to *8* if the firmware was successfully loaded (Figure 5).

Creating and Uploading the Firmware

The easiest way to create new custom firmware is to copy the default one and modify the original. Copying the default not only preserves a workable firmware if you need to recover from a flawed one, but saves you typing. To customize, open a copy of `keymap.c`, and scroll down to the line:

```
const uint16_t PROGMEM
keymaps[][MATRIX_ROWS][MATRIX_COLS] =
```

Following this line are the key layouts for each layer. The first layer is designated 0, and has the nine keys laid out in three rows, with each row a comma-separated list of keycodes (Figure 6). You can have up to 16 layers, each customized, if you choose, for a particular application or game, or any other combination of keys you find useful. Note that some keys may have no effect in a particular application; for example, an F11 key will have no effect if an application does not use that function key. You can edit the default layer or add another layer by copying the default layer, changing the layer number, and modifying the keycodes. In either case, use the summary of keycodes as a guide [4].

Keys can be alphanumeric, control or function keys, or for sound, Bluetooth, or toggling backlights and changing their brightness. Advanced key functions include one-shot keys that remain active until another one is pressed, making key combinations easier, and Mod-Taps keys that behave differently when tapped instead of held down. When building a custom setting, it is best to do it one key at a time, because not all keycodes in the QMK documentation appear to be implemented, and some are specific to the operating system. Be sure to preserve the commas and the surrounding parentheses, or your custom layer will not compile.

If you have multiple layers, be sure that you provide a navigation system to switch between layers. Since the documentation suggests not switching to a lower level, the best method is to create a toggle system, using the keycode `DF(1)` on layer 0 to allow switching to layer 1, `DF(2)` on layer 1, and so on, ending with `DF(1)` on the highest layer you are using, `DF(2)` on layer 1, and so on. This navigation system uses one key on each level, but still leaves plenty for other macros.

When your custom firmware is completed, save it as the default and upload it using the version of the command structure given above to test the environment. Contrary to the documentation, a keymap with a different name may not compile. At times, too, an *Error 1* message is given, but checking the device shows that the firmware was successfully uploaded.

If you prefer a graphical interface, you can customize layers using the online instructions' steps 9-16 [5], and place the resulting `.json` file in a place where you can conveniently compile and flash it. To customize, open the QMK Keyboard Firmware page, and scroll down to the Treasure Keyboard Macropad button. Either way, a simple map of the TMT9's keys displays (Figure 7). Do not change anything on the *Wiring* or *Pin* tabs, but click instead on the *Keymap* or *Macro* tab.

To program a key with the online interface, select the layer (starting at 0), and click a key. At the bottom of the page, the current configuration for the key displays in a box. Click the box, and chose another configuration from the graphic interface. The site contains a summary of the available keycodes built into the interface.

When you have configured all the keys you want in the graphical interface, go to the *Settings* tab to save the source file on your machine. The only setting you need to change is the name, but you might want to set the brightness of the backlights on a 10 point scale, with 10 being the highest (Figure 8). Then move to the *Compile* tab to produce the finished `.hex` file and

```
const uint16_t PROGMEM keymaps[][MATRIX_ROWS][MATRIX_COLS] = {
  [0] = LAYOUT( /* Base */
    KC_0,  KC_1, KC_2, \
    KC_3,  KC_4, KC_5, \
    KC_6,  KC_7, KC_8  \
  ),
```

Figure 6: To customize the firmware, edit the text-based map of the keys in a copy of `keymap.c`.

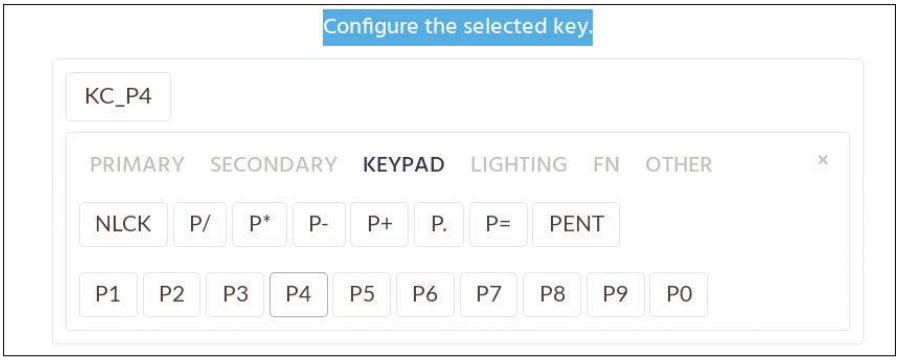


Figure 7: The online configuration tool.

Figure 8: You can adjust the backlight's brightness in the Settings tab.

download it to your computer. As you work, you may want to record a layer's contents on one of the two cards that comes with the device for future easy reference.

To upload the `.json` file, install QMK CLI [6], and use the command `qmk compile`; don't forget to disable the bootloader by poking the wrench into the hole on the bottom of the device.

The End Results

Countless variables make setting up the TMT9 with Linux a slow process. Here, I can only discuss the circumstances you are most likely to find. Setup is not helped by the fact that the TMT9 hides the indicator light that shows when it is reset, and the DIYer can only go by the slight give when the wrench is poked into the inner workings. Would an external LED really be too much to ask?

Still, that is the nature of DIY products. While my stubbornness in insisting on working in Linux complicated setup immensely for me, in the process I learned a lot about AVR devices and how they work, gaining a body of knowledge that I would have missed had I given in to the temptation to work on a neighbor's Windows machine.

I am surprised, though, that Treasure has not bothered to add Linux support. After all, while Linux users may be a small minority, I suspect that they include a percentage of DIYers that is far higher than among Windows or macOS users. Or possibly, the expectation is that Linux users will know enough to follow the instructions on the QMK site.

However, in the end, I don't regret the effort. The TMT9 offers more convenience and efficiency than any other device with the smallest footprint that I have ever encountered. I would far rather have a TMT9 on a keyboard than a number pad, for instance. Moreover, in its red anodized aluminum case, it is an elegant device. I am still organizing layers, but I can already see that it will lead to a major improvement in my workflow – to say nothing of my efforts to reduce repetitive stress injuries. ■■■

Info

[1] TMT9: *http://macropad.co/*

[2] QMK documentation: *https://beta.docs.qmk.fm*

[3] Cherry MX Keycaps: *https://duckduckgo.com/?q=cherry+mx+keycaps+blank&t=ffnt&ia=shopping*

[4] Keycodes for programming: *https://docs.qmk.fm/#/keycodes?id=keycodes-overview*

[5] Windows and macOS instructions: *http://macropad.co/support/*

[6] QMK CLI: *https://docs.qmk.fm/#/cli*

MakerSpace

Build a custom Linux for your Rasp Pi with Yocto

Tailor-Made

Yocto is a tool for creating custom Linux images for embedded devices. We'll show you how to create a customized Linux for your Rasp Pi. *By Johan Thelin*

Yocto [1] is a distribution builder. The tool helps users create customized images. A custom image is particularly tempting in the embedded environment, where hardware resources are limited.

In the Beginning

Yocto builds images based on recipes. The user can modify the recipe to determine how the system should be structured – you can even specify which compiler will be used for compiling the source code.

Yocto comes with a reference distribution called Poky, which includes many ready-made recipes and is based on OpenEmbedded [2]. A tool called `bitbake` lets you combine the Poky layer with a hardware-specific board support layer.

One of Yocto's strengths is the many layers available from the start, including layers from most board manufacturers and also from numerous open source projects. Figure 1 shows the extensive and searchable OpenEmbedded Layer Index [3]. Yocto also has an excellent

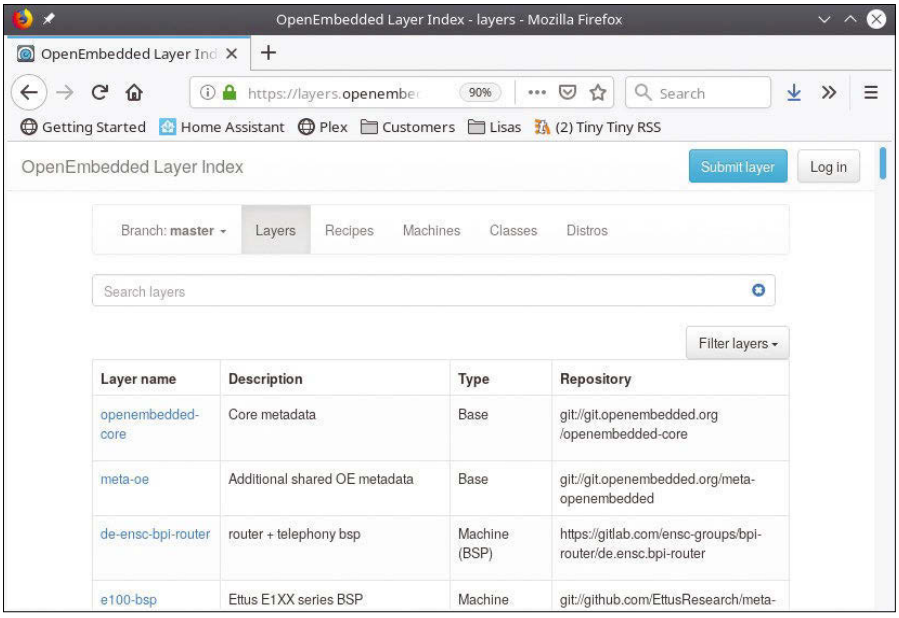


Figure 1: The OpenEmbedded Layer Index.

Listing 1: Cloning Repositories

```
01 git clone -b warrior git://git.yoctoproject.org/ poky.git
02 cd poky
03 git clone -b warrior git://git.yoctoproject.org/meta-raspberrypi
```

manual [4] that explains all aspects in detail.

The easiest way to get started with your own image is by downloading `poky` and `meta-raspberrypi` and cloning the Git repositories (Listing 1). The convention is to place all layers below `poky`. It is important to always clone the same branch name of each layer, e.g. Warrior. Warrior is the name of the current Poky version. The layer names usually start with *meta-*. In fact, Poky contains a large number of *meta-** directories, because it consists of several layers.

After cloning, you need to create a suitable environment by running the following command:

```
source oe-init-build-env
```

This command also updates the `PATH` variable so that the Yocto tools can be called from anywhere. The command creates a `build` directory with a `conf` subdirectory. `conf` contains two interesting files: `bblayers.conf`, a list of the layers used, and `local.conf`, instructions for Yocto on how to build the image.

In the `bblayers.conf` file, edit the line with `meta-yocto-bsp` and enter the location of the cloned `meta-raspberrypi` layer. In `local.conf`, add the following line:

```
MACHINE ?= "raspberrypi3"
```

This line tells Yocto which hardware model the build will be for. Look online for a list of the models supported by the Raspberry Pi layer [5].

Listing 2: lm-hello.c

```
01 /*      HelloWorld.c    */
02
03 #include <stdio.h>
04
05 main()
06 {
07  printf("Hello Linux Magazine\n");
08 }
```

Initially Minimal

To check if everything is set up correctly, build a `core-image-minimal` with the following call from the `build` directory:

```
bitbake core-image-minimal
```

Do not be put off by the time and space required for this action. Yocto builds an image that contains a small Linux system, as well as all the tools needed to build the distribution, such as a compiler. The next `build` call then works incrementally; in other words, later steps do not need to rebuild the tools created in a previous step.

When the build process is done, examine the `build` directory. You will find four new subdirectories: `cache`, `downloads`, `sstate-cache`, and `tmp`. The most interesting of these subdirectories is `tmp`, in which the resulting image takes its place. Below `downloads` are downloaded files, and `sstate-cache` and `cache` serve the build process as cache storage. In more complex scenarios, several computers can share `sstate-cache`.

The `tmp` directory has a number of subdirectories, one of which is called `deploy`. The `deploy` directory contains all packages used for the platform and their licenses, as well as the generated image. The `core-image-minimal-raspberrypi3.rpi-sdimg` file in the `tmp/deploy/images/raspberrypi3/` directory is a link to the last successful build.

With the help of the `dd` command, you can easily transfer the image to an SD card. Be very careful with `dd`, which overwrites the copy target. Make doubly sure that the correct output device is specified so that you don't accidentally overwrite the operating system.

After plugging in the SD card, search the `dmesg` output for messages from the last device added; the name of the device appears in the output. You can then specify the device name in the `dd` command using the `of=` option:

```
sudo dd if=core-image-minimal-➥
raspberrypi3.rpi-sdimg of=➥
/dev/mmcblk0 bs=1MB
```

The `if=` option refers to the input file, and `bs=` is the block size.

Once the microSD card is ready, you can insert it into a Raspberry Pi, connect the keyboard and monitor, and boot. Afterwards, you can log in as root without a password.

Customizing the Image

Once the basic image is created, you can customize and extend it. This is a three-step process: First, a new layer is

Listing 3: lm-hello_1.0.bb

```
01 SUMMARY = "Hello World program for Linux Magazine"
02 SECTION = "examples"
03 LICENSE = "GPLv2+"
04 LIC_FILES_CHKSUM = "file://COPYING;md5=b234ee4d69f5fce4486a80fdaf4a4263"
05
06 SRCREV = "6c2970ab5277e23f55de9853a38c6c173d0b04ee"
07 SRC_URI = "git://github.com/e8johan/lm-hello.git"
08
09 PV = "1.0+git${SRCPV}"
10
11 S = "${WORKDIR}/git/"
12
13 EXTRA_OEMAKE = "'CC=${CC}' 'CFLAGS=${CFLAGS}' 'LDFLAGS=${LDFLAGS}'"
14 TARGET_CC_ARCH += "${LDFLAGS}"
15
16 do_install() {
17     install -d ${D}${bindir}
18     install -m 0755 lm-hello ${D}${bindir}
19 }
```

created, then you create a recipe for a new application you wish to integrate, and then a new image recipe.

For the first step, the new layer, run the following command in the `poky` subdirectory:

```
bitbake-layers create-layer meta-lm
```

This command creates the new layer with a sample recipe. Then add this layer to the list in `bblayers.conf`, which can either be done manually or with the `bitbake-layers` command:

```
cd <I>Path/to/Build<I>
bitbake-layers add-layer ../meta-lm
```

Finding Dependencies

A manifest file attached to a Yocto image contains a list of all included packages, along with their version numbers and architecture details.

For example, the `core-image-minimal-raspberrypi3.rpi-sdimg` image has a `core-image-minimal-raspberrypi3.manifest` file. The file contains a line for the `lm-hello` package, and it says:

```
lm-hello ➥
cortexa7t2hf_neon_vfpv4 ➥
1.0+git0 +6c2970ab52
```

To find the dependencies for the package, use the `bitbake` command with the `-g` option:

```
bitbake -g lm-hello
```

This command creates two files, `task-depends.dot` and `recipe-depends.dot`. The first file describes the order in which Yocto executes the `bitbake` tasks, and the second describes the dependencies between the recipes. The `.dot` files can be rendered to PNGs but are not quickly readable. For the `lm-hello` package, there are more than 400 dependencies. If you repeat the exercise for `lm-image-minimal`, you get more than 6,000 dependencies.

You can search the `.dot` files for information. If, for example, you filter out all rows starting with `lm-hello`, you see a result like the one in Listing 4. From this information, you can deduce that `lm-hello` depends on the packages listed on the right. The remaining lines include headers that are not deployed, as well as the C library and other run-time dependencies of the C program.

The new layer doesn't do anything until you add a recipe. In the present case, the recipe is a simple small C program in "Hello World" style: `lm-hello.c` (Listing 2).

Create a `lm-hello` directory below `recipes-lm` and remove the sample recipe. In the new `lm-hello` directory, you need a `lm-hello_1.0.bb` file with the contents of Listing 3. The file extension `bb` stands for *BitBake Recipe*; the underscore in the filename must be followed by a version number, which will be used later when `bitbake` builds a package based on the recipe.

The file itself consists of variable declarations and functions. The variables `SUMMARY` and `SECTION` act as metadata describing the content. `LIC_FILES_CHKSUM` contains an MD5 checksum of the license file to ensure that the content has not changed since the recipe was written.

A reference to the source code follows, here in the form of a URI (`SRC_URI`) and a specific revision (`SRCREV`) of a Git repository. In this way, different versions of a recipe can exist, each referring to a different revision. The version number of the package is specified by `PV`; in this case, the number comes from the version identifier in the filename.

The following lines apply to settings for the build process. The `lm-hello` project is based on a makefile. Yocto can use a variety of build systems, including CMake, Autotools, and custom files. To support them, the file exports the variables `EXTRA_OEMAKE` and `TARGET_CC_ARCH`. These variables ensure that the right compiler with the right flags is used.

Whenever `bitbake` processes a recipe, it goes through a series of steps. This recipe can use default values for most of these steps, except for the installation step. It needs a custom function that first creates a `bin` directory (`${bindir}`) below the stage area (`${D}`), where it then installs the binary of `lm-hello`.

As soon as the recipe is prepared, you can test it outside an image with `bitbake`:

```
cd Path/to/Build
bitbake lm-hello
```

The result is a package instead of an image. This package can be found at `build/tmp/deploy/rpm`.

Two things are particularly noteworthy. First, even the simplest recipe always produces three packages at once: An installation package without an appendix in the filename, a development package with `dev` in the name, and a

Listing 4: Filtered Files

```
01 "lm-hello" [label="lm-hello\n:1.0+gitAUTOINC+6c2970ab52-r0\n/home/e8johan \
/work/linuxmagazine/yocto/build/poky/meta-lm/recipes-lm/lm-hello/lm-hello_1.0.bb"]
02 "lm-hello" -> "binutils-cross-arm"
04 "lm-hello" -> "gcc-cross-arm"
05 "lm-hello" -> "gcc-runtime"
06 "lm-hello" -> "glibc"
07 "lm-hello" -> "libgcc"
08 "lm-hello" -> "linux-libc-headers"
09 "lm-hello" -> "pseudo-native"
10 "lm-hello" -> "quilt-native"
11 "lm-hello" -> "rpm-native"
```

Listing 5: lm-image-minimal.bb

```
01 require recipes-core/images/core-image-minimal.bb
02 DESCRIPTION = "Minimal image for Linux Magazine"
03 IMAGE_INSTALL_append = " lm-hello"
```

debug package identifiable by `dbg` in the name.

Second, the packages are always located in a CPU-specific subdirectory of `build/tmp/deploy/rpm` (in the example, this is `cortexa7t2hf_neon_vfpv4`) and not in a subdirectory named after the board. This means that the package can be used on all computers with the same CPU. The `bitbake -g` option will help you find dependencies associated with the package. (See the box entitled "Finding Dependencies.")

Creating a New Image

If you want to add the `lm-hello` program to an image, you need an image recipe. The image recipe is another variant of a BitBake recipe, but it differs significantly. In this case, the image recipe will be named `lm-image-minimal.bb`, and it must reside below the `meta-lm` layer in the subdirectory `recipes-lm/images`:

```
cd Path/to/poky/meta-lm
mkdir -p recipes-lm/images
vim recipes-lm/images/
lm-image-minimal.bb
```

The contents of the file are shown in Listing 5. The recipe includes the `core-image-minimal.bb` recipe used earlier in this article. It then inserts another `DESCRIPTION` before adding the `lm-hello` package to the `IMAGE_INSTALL` variable, causing the package to end up in the image.

Appending the package to `IMAGE_INSTALL` highlights one of Yocto's special aspects. With most other tools, the obvious approach would be to append the `lm-hello` variable to the `IMAGE_INSTALL` variable with an operator like `+=` or `= ... + lm-hello`. But this would just confuse `bitbake`. Instead, you will need a second variable with the `_append` suffix, whose contents are added to the previous variable value.

After building, the resulting image is then called `core-image-minimal-raspberrypi3.rpi-sdimg`.

Conclusions

This article introduced you to Yocto, a tool for building custom distributions. Yocto offers far more freedom and control than would be possible with a pre-built distribution. Having said this, these additional features do come at a price of significantly more complexity and overhead. ■■■

Info

[1] Yocto: *https://www.yoctoproject.org*

[2] OpenEmbedded: *http://www.openembedded.org/wiki/Main_Page*

[3] OpenEmbedded Layer Index: *http://layers.openembedded.org/layerindex/branch/master/layers/*

[4] Yocto Project Reference Manual: *https://www.yoctoproject.org/docs/2.7/ref-manual/ref-manual.html*

[5] Raspberry Pi layer: *https://meta-raspberrypi.readthedocs.io/en/latest/readme.html*

Author

Johan Thelin works as a system architect. Over the past 15 years, he has developed embedded systems for various industries including medical laboratories, home automation, security, and automotive. He also wrote a book on the QT basics, as well as one on QML. In his spare time, he organizes the FOSS-North Conference, which takes place annually in Gothenburg, Sweden.

Linux package managers have been around for years, and for most of that time, the technology chugged along without a lot of changes or disruption. But in recent years, a new generation of package managers has invaded the scene. Modern package systems like Snappy and Flatpak bring their dependencies with them and run in protected environments for enhanced security and stability. Where do you learn about these new technologies that descend upon the Linux world and disrupt the established order? Why here of course.

We don't just cover these new-age package formats – we cover all the other tools surrounding them. This month we bring you Chob, a command-line application that lets you perform keyword-based searches of the Flatpak, Snap, and AppImage app stores. We also show you Anbox, a cool tool that lets you run Android apps on your Linux desktop, and we show you how to animate your drawings with Pencil2D.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE

Doghouse – Updating Technical Skills 70
Jon "maddog" Hall
As operating systems and computer languages evolve, programmers need to keep learning new skills.

Chob 72
Ferdinand Thommes
If you are looking for an application in the AppImage, Flatpak, or Snap app stores, Chob lets you perform a keyword-based search from the command line.

Anbox Android Apps on Linux 76
Erik Bärwaldt
Thanks to Anbox and Snap, you can launch Android apps in a Linux virtual environment.

Foliate 80
Christoph Langner
You can customize the Foliate ebook reader to suit your needs with bookmarking, translation, and read aloud features.

Pencil2D 84
Tim Schürmann
Pencil2D, an easy-to-use painting and 2D animation program, lets you create small animations quickly. Despite the simple user interface, you might need a little help getting started.

FOSSPicks 90
Graham Morrison
This month Graham explores Zrythm, Mumble 1.3, NoteKit, Kirogi, pastel, Stunt Car Racer Remake, and more!

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

As operating systems and computer languages evolve, programmers need to keep learning new skills. BY JON "MADDOG" HALL

Old dog, new tricks

I belong to several social media sites that are frequented by older programmers and engineers. We exchange memories and technical tidbits from the past. Many of these people are retired, but some are not.

Recently, one person mentioned that people do not listen to him anymore, particularly when he is telling a story about something that happened to him years ago in the tech field. Many others chimed in and said the same thing happened to them. Some stated that they had stopped talking about technical items altogether.

Over time, it became clear that often these people were either talking to non-technical people and the technical conversation went way over their heads, or the person was talking about technical detail from many years ago that was just not relevant to their audience.

While the fact that you had to toggle in 17 12-bit instructions to a Digital Equipment Corporation (DEC) PDP-8 computer system in order to get it to read a binary tape loader into memory may be interesting in terms of computer history, most people (unless they are at a computer museum) would not care. They are more interested in trying to get their iPhone to work or figure out how to more quickly create a document.

Sometimes a conversation might turn to a deep technical issue that calls for some interesting bit of technical data, but that would be a rare cocktail party indeed.

When I mentioned that most people today are more interested in modern day operating systems and languages than the ones used 30 years ago, a frosty silence came over the conversation. One person said "I am not going to learn anything new … what is the use."

I find that philosophy puzzling. As a technical person, I always like reading about new technology. Perhaps I do not have the time to learn exactly how that new technology works, but I do like knowing of its existence and why it is valuable. With the explosion of new software and hardware in so many different areas, I often may not know what a new piece of technology is or does. I simply tell other people that I have not had the time to read about it, and therefore I have no opinion as to its use. After that, I will do a quick Internet search and get at least a little insight into what the product or project does for the next conversation.

Many times, I run into programmers who have used older systems and languages and now do not fit into the work environment. Their technical skills are old and jobs using those skills are few and far between. However, if they took just a little time to learn a new language or some of the newer skills, they would "refresh themselves" and probably could re-enter the job market very successfully. They have skills that younger programmers do not have yet, and both groups could learn from each other.

Often at a conference, young people will come up and ask questions about their career path or whether they need heavy math skills for programming. They listen as I pull out real-life examples of why my advice should be a sound foundation for their decision making. I can not make the decision for them, but I can help them see different parts of the decision tree based on my experiences.

Another group of people whose skills often need "refreshing" are young parents who learned computer science or computer engineering and then took time off to start and raise a family. After their children are old enough to start attending school, the parent now wishes to go back to work, but finds out that their former skills are "stale." At the university where I taught from 1977-1980, we had a program that taught these bright, knowledgeable people the latest methods and procedures that would let them re-enter the workforce. The course lasted only nine months, but most of them were successful at re-entering after being out of the industry for 10 years or more. These days, people can go to the web and find the information about what they need to study and even the study materials they need.

There is a young man of 16 years who is giving a lecture on quantum computing at my conference. He started programming at the age of eight and started a PHP user's group at the age of 14.

We had a great conversation at breakfast about different types of processors that will probably be used in the near term along with new types of computing that may not be seen in production for some time.

It was the right conversation with the right person at the right time, and we both enjoyed it. ■■■

COMPLETE YOUR LIBRARY

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

ORDER YOURS TODAY!
shop.linuxnewmedia.com

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

A command-line search tool for AppImage, Flatpak, and Snap

Rich Harvest

If you are looking for an application in AppImage, Flatpak, or Snap app stores, Chob lets you perform a keyword-based search from the command line.

BY FERDINAND THOMMES

The alternative package formats AppImage, Flatpak, and Snap can be found in almost all distributions. Some projects even rely entirely, or at least predominantly, on these modern constructs that include most dependencies in the package. For example, Fedora's Silverblue Workstation is completely based on Flatpak; Clear Linux OS supports Flatpaks, as well as in-house bundles. Endless OS is also increasingly taking this path to deliver software. Although Ubuntu mainly uses its Snap format in the cloud, it also offers an increasing number of Snap packages for its desktop versions.

Testing

These new formats are especially useful for testing new software versions. For instance, the LibreOffice developers recommend staying with the previous version and waiting a couple of minor releases before making the final switch for production use.

However, if you want to find out whether the new version fixes a certain bug, trying out a corresponding Flatpak does not conflict with the previously installed, production-use version. In the AppImage format, there are even four different development stages of LibreOffice letting you safely try out new features in advance.

Complex Search

What if you want to find out which applications are available in the new formats for a certain application genre? Until now, you had to manually search each of the app stores. Flathub [1], Snap Store [2], and AppImageHub [3] each offer hundreds or even thousands of apps in their respective formats.

To remedy this, Mohammed Kaplan has developed Chob [4], a command-line search engine that searches these app stores using a keyword. Chob is licensed under Apache v2, builds on Node.js, and is available on GitHub.

On Chob's download page [5], you will find a package in deb format, as well as the executable `chob-linux` for other distributions. The source code is also available. You can install the deb package in the usual way with:

```
dpkg -i chob
```

After downloading the `chob-linux` binary, first make it executable by typing

```
sudo chmod +x chob-linux
```

and then launch the tool by entering the program name followed by the search term.

To build the tool yourself, you will need to install at least the latest Node.js LTS version (currently version 10.x). Then drag the software from the repository (Listing 1, line 1) onto your hard disk, change to the newly created directory (line 2), and install the application (line 3).

No Options

Chob is easy to use, because apart from the parameter for the search word, the small tool has no options. If you enter `chob music` in a terminal emulator, or (depending on your version) `./chob-linux music`, you will see a list of matches. In a test run, the list contained 15 applications that contained the string `music` (Figure 1).

Listing 1: Building Chob

```
01 $ git clone https://github.com/MuhammedKpln/chob.git
02 $ cd chob
03 $ npm install && yarn
```

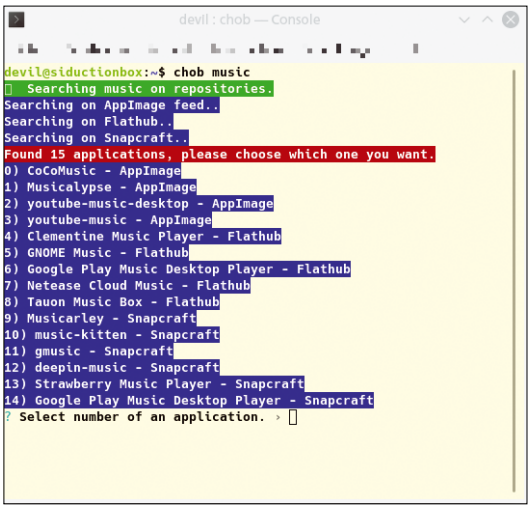


Figure 1: Searching for `music` revealed 15 applications whose name contained that string.

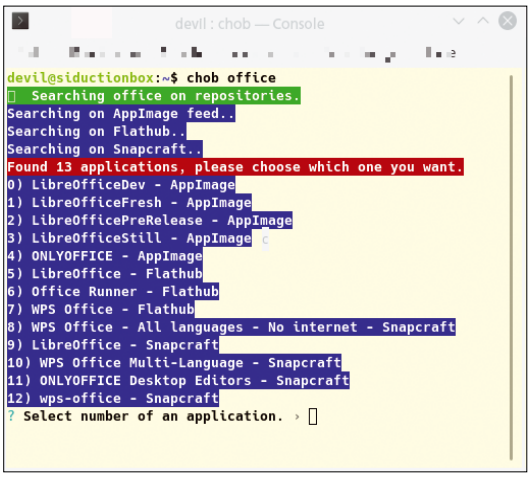


Figure 2: Four versions of LibreOffice at different stages of development are available – a great example of the advantages of alternative package formats.

Keeping with the LibreOffice example, searching with a keyword of `office` results in the four previously-mentioned development versions of LibreOffice as AppImages, one package each in the Flatpak and Snap formats, as well as other office suite packages (Figure 2).

Unix Philosophy

Entering a number from the search results at the command line opens the selected application's page from the respective app store in your browser and offers to either download or install the app depending on the package (Figure 3). Following the Unix philosophy of "do one thing and do it well," Chob does not do any more than this.

Conclusions

Chob proves to be a useful little tool when it comes to finding software that is available in alternative package formats. You may even find applications for a given keyword that you didn't know existed, because the app stores offer numerous applications that are missing from many distributions' repositories. ■■■

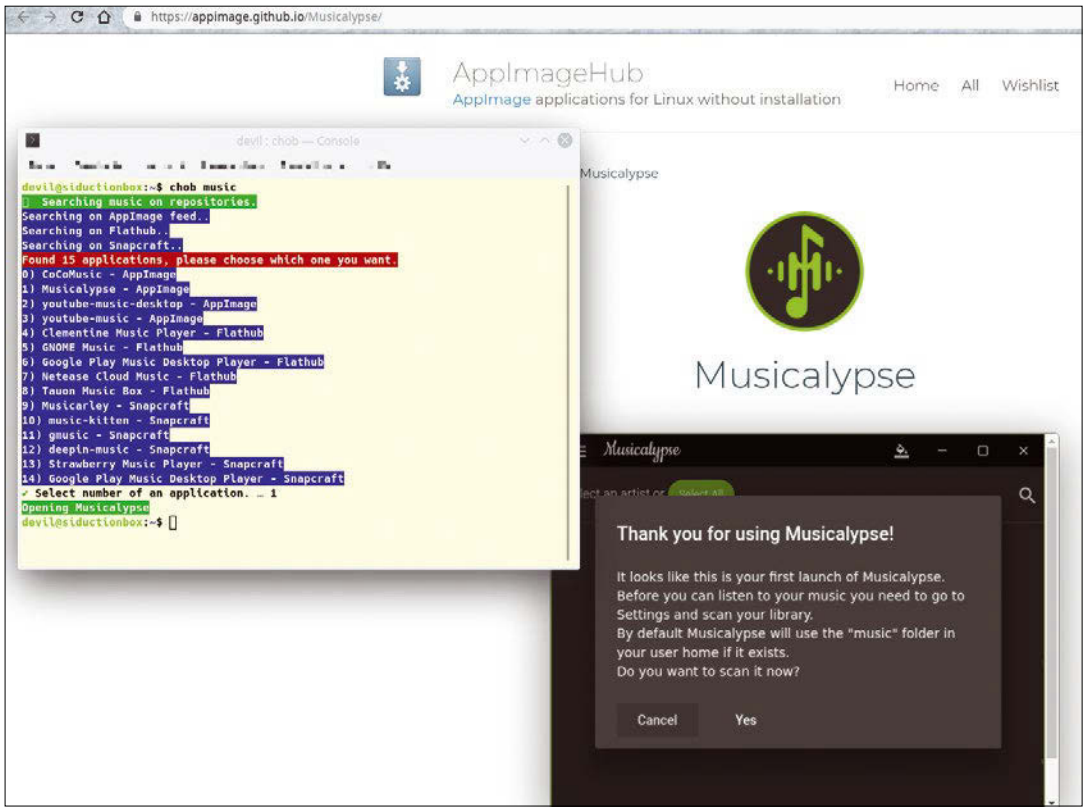


Figure 3: Entering the number of the desired application takes you to the corresponding app store in your web browser. In the case of AppImage, you can download the application, while Flatpak and Snap let you install it.

Info

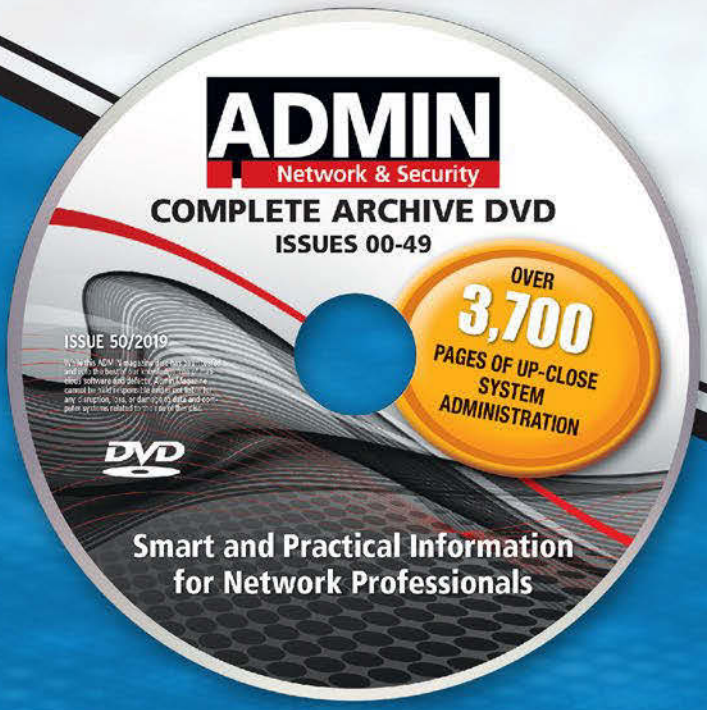
[1] Flathub: *https://flathub.org/home*

[2] Snap Store: *https://snapcraft.io/store*

[3] AppImageHub: *https://appimage.github.io/apps/*

[4] Chob: *https://github.com/MuhammedKpln/chob*

[5] Download Chob: *https://github.com/MuhammedKpln/chob/releases*



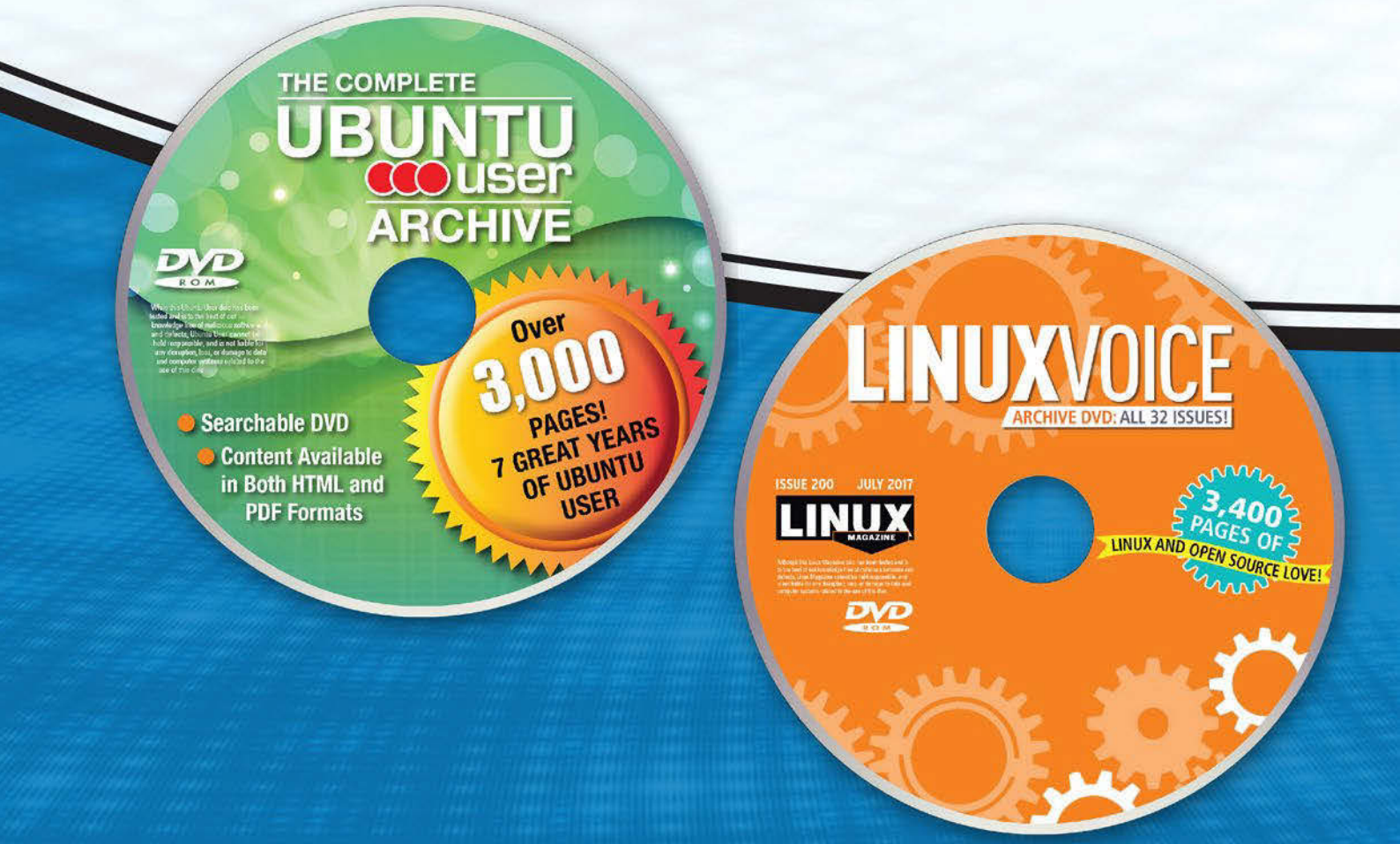
Complete Your Open Source Library with Archive DVDs!

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

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

Order Your DVD Now!

shop.linuxnewmedia.com



Launch Android apps on Linux with Anbox
Piggyback

Have you ever wished you could use an Android smartphone app on your Linux desktop? Thanks to Anbox and Snap, you can launch Android apps in a Linux virtual environment.

BY ERIK BÄRWALDT

Linux already supports Java applications on the desktop, as well as native programs. You can rehash some legacy Windows software with Wine or Crossover. With the help of the DOSbox in Linux, you can even breathe life into very old applications for the 16-bit DOS operating system.

Anbox [1] pursues a similar approach. You can integrate a full-fledged Android operating system under Linux with a few simple steps and work with the included apps even more comfortably than on your Android smartphone.

The free Anbox software is only available as source code and as a Snap archive [2]. This means that the application only runs on Linux distributions that support Snap's virtual container technology, which includes all Ubuntu and Debian derivatives.

Anbox contains a complete Android system in a Snap container. The container is completely isolated from the host system thanks to Linux containers (LXC) [3]. Android thus does not have direct access to the Linux host's hardware. Instead, this access is managed by the Anbox daemon on the host, which makes it difficult to transfer malware to the Linux host.

To use Anbox, you first need to set up the Snap daemon. You can do this easily on Ubuntu, Linux Mint, or Debian by using Synaptic to install the *snapd* package from the repositories. Next, you need to integrate two further kernel modules (Listing 1, lines 1 to 3).

Rebooting loads the modules. You will now find two new entries, `ashmem` and `binder`, in the `/dev/` directory. Next, use Snap to install Anbox with the command from line 4 of Listing 1. The installation process takes a little longer even on state-of-art hardware, and numerous status messages are output on screen.

When the installation is complete, you will find the *Anbox Application Manager* and *Android Settings* entries in your desktop's menu hierarchy. Click on the *Anbox Application Manager* to start the Android environment on Linux (Figure 1).

The software initially only displays preinstalled applications. You now need to prepare the Android container for integrating more applications by setting up the Android Debug Bridge (Listing 1, line 5).

Listing 1: Installing Anbox

```
01 $ sudo add-apt-repository ppa:morphis/anbox-support
02 $ sudo apt update
03 $ sudo apt install linux-headers-generic anbox-modules-dkms
04 $ snap install --devmode --beta anbox
05 $ sudo apt install android-tools-adb
```

Software

Because Google does not certify the Anbox project and there is no agreement to use Google Play, Google is not preinstalled with Anbox. Consequently, Google can't collect your personal data via the Play Store. Instead, the Debug Bridge lets you add packages in Android's own APK format to the Anbox container.

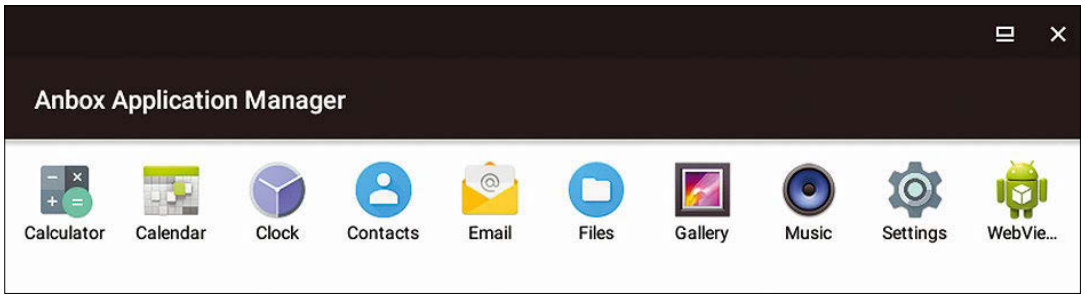


Figure 1: The Application Manager functions as Anbox's interface.

First search for an APK mirror on the Internet that provides the packages. You will usually find current and older versions of several thousand Android apps. After downloading the desired apps, type the following in the terminal window without root privileges:

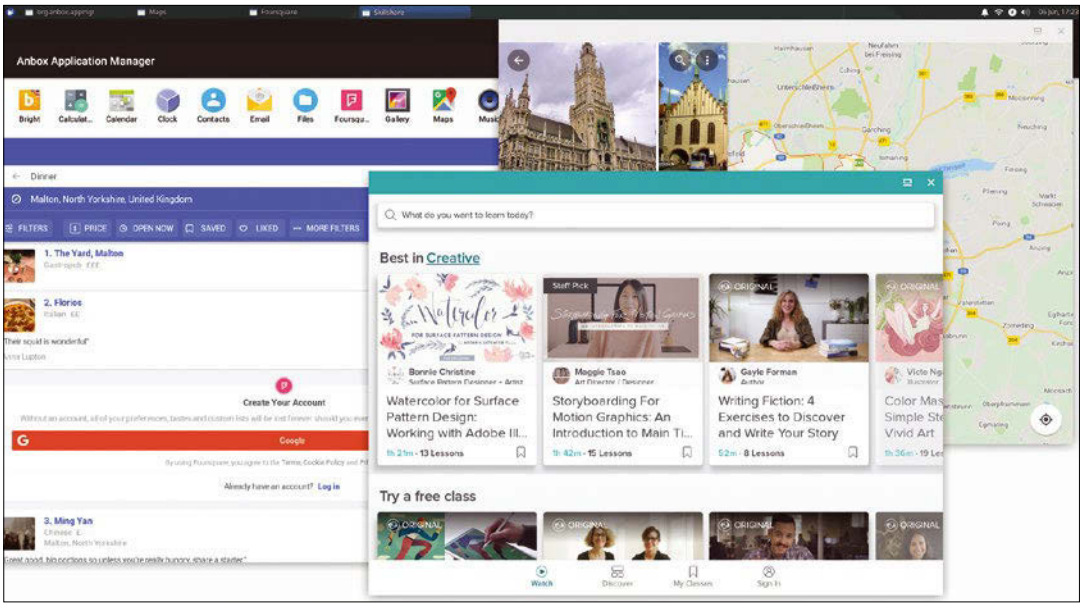


Figure 2: Anbox supports the simultaneous display of several Android apps in different windows on the desktop.

```
adb install <APK package>
```

This installs the app. Note that Anbox must be running when the installation command is called; otherwise, the installation aborts with a device error.

Although most of the programs offered on the mirror servers are designed for the ARM hardware used in smartphones and tablet PCs, this does not matter for the installation on Anbox. Problems only arise if apps require special WLAN or GPS hardware. If this is the case, an error message will appear in the Linux terminal.

Once you manage to install the APK package, the terminal displays a *Success* message. You simultaneously will find a starter icon for the software in the Anbox Application Manager that lets you open the app with a mouse click.

In contrast to a smartphone, several applications can be launched simultaneously as each runs in its own window. This lets you use multiple Android apps along with other programs on the desktop. Since you can resize the windows, an app's contents can be viewed much more clearly than on the comparatively small smartphone displays (Figure 2).

Troubleshooting

Occasionally, an Android app tries to access special hardware that is missing on the system. This kind of direct access typically crashes the program in question, taking Anbox down at the same time.

Fortunately, each Anbox window on the Linux system runs in a separate process with its own process ID (PID). To fix the problem, use `ps ax` to find the faulty process in the terminal and terminate it with

```
kill 9 <PID>
```

To remove the app from the Anbox Application Manager, enter the command

Listing 2: Updating and Deleting Anbox

```
01 $ snap refresh --beta --devmode anbox
02 [...]
03 $ snap remove anbox
04 $ sudo apt install ppa-purge
05 $ sudo ppa-purge ppa:morphis/anbox-support
```

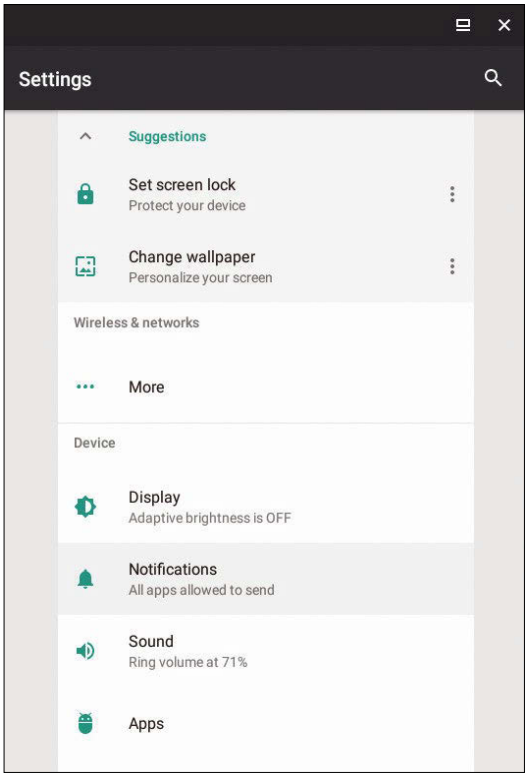


Figure 3: The Anbox settings are limited to a single dialog.

```
adb remove <APK packet>
```

in the terminal. The routine now removes the files and the starter icon from the Anbox window.

Settings

You can access Anbox's Settings dialog by clicking on the *Settings* icon in the Application Manager or via *Android Settings* in the desktop menu hierarchy. You will find numerous setting options not only for the physical target system, but also for the replicated smartphone.

The size settings for fonts and messages in the Anbox windows are especially useful. Making the appropriate adjustments means that the contents are easier to read in small windows (Figure 3). However, in our lab with several Linux distributions, some control elements did not react at all, while others crashed the window.

Resources and Administration

Anbox uses very little in terms of resources on the Linux host compared to other emulators or runtime environments. The main memory requirement for the container is also moderate. The Application Manager occupies about 240MB RAM, while the Settings Manager takes around another 85MB. Each app you launch increases the memory requirement by about 15MB. This means that Anbox is quite fast even on older hardware with little RAM.

The Snap daemon manages the Anbox container. Some terminal commands, which differ from their counterparts for conventional package management, are used for this purpose. For example, Snap for Anbox does not offer automatic updates; you have to do this manually (Listing 2, line 1). To discover the current version of the Anbox container, type:

```
snap info anbox
```

To remove Anbox from the system, delete the container (line 3). In addition, you also need to remove the matching kernel modules, which requires the *ppa-purge* package (line 4). Then remove the PPA archive (line 5). You have now completely removed Anbox and its helpers from the system.

Conclusions

Anbox is a quick solution for taking Android applications to the Linux desktop. The container solution, which is still under constant development, is already surprisingly stable and can be used in production.

Restrictions that apply to smartphones do not apply on Linux. Not only can multiple apps be opened simultaneously, but their windows can also be displayed at any scale. In addition, Android apps can coexist on the Linux desktop parallel to other windows without any problems. This greatly boosts the desktop's usefulness.

However, it would be nice if Anbox supported other container formats, such as AppImage, so that users outside of the Ubuntu/Debian universe could take advantage of it. ■■■

Info

[1] Anbox: *https://anbox.io*

[2] Snap: *https://snapcraft.io*

[3] LXC: *https://linuxcontainers.org*

Explore your e-reader options with Foliate

Reading Material

The Foliate ebook reader can be comprehensively customized to suit your needs, including bookmarking, translation, and read aloud features. BY CHRISTOPH LANGNER

About 10 years ago, "Shock your parents – read a book" was a slogan at German libraries. It didn't help much: Book publishers are still battling stagnation. Nevertheless, there is still some reading going on, although now it focuses more on ebooks and less on printed material. It's not without reason that Amazon regularly offers new versions of its Kindle ebook reader.

To read ebooks, you do not necessarily need a special device. The package sources of the current Linux distributions contain numerous software-based ebook readers. In addition to a read mode, they often offer additional functions such as bookmark management, annotations, or a translation tool.

This repertoire is also offered by the still very young Foliate [1] ebook reader. It is based on the modern GTK3 framework and harmonizes perfectly with the Gnome desktop. In addition, it offers many setting options, which already makes the application a good alternative to classics like Calibre.

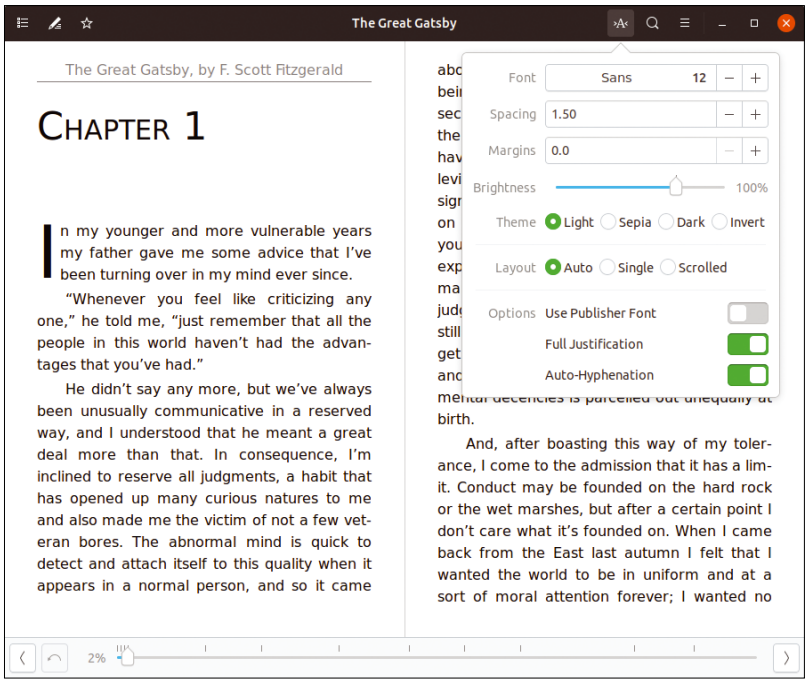


Figure 1: Foliate is a modern ebook reader with gesture support and integrated voice output.

Installation

To install Foliate, most users have to rely on packages provided by the developers. The program is not yet included in the package sources of Debian, Ubuntu or Linux Mint. On the GitHub *Releases* page you can pick up the latest version in the form of DEB packages [2]. At the editorial deadline, Foliate 1.5.3 was the latest version; it installed on a recent Ubuntu system without any issues.

Of the big distributions, only Fedora (F29 and higher) provides the program in the package manager, but this is an older version. Arch Linux users will always receive the current version of Foliate via the AUR; the entries you need here are *foliate* or *foliate-git*. Alternatively, the project provides detailed information on its homepage on how to build the software from the source code.

An Open Book

The software can load ebooks in EPUB, MOBI, AZW, and AZW3 formats. Suitable reading material can be obtained, for example, from Project Gutenberg [3].

At startup time, Foliate launches with an almost empty page. Use the *Open File…* button or the *Open* item from the hamburger menu in the top right corner to load an ebook file from the hard disk. Like all modern GTK3 applications, Foliate does without a classic menubar; all options are located in the window bar. On the left, you can open the table of contents; next to it are notes and the bookmarks. On the right, you can configure the display, launch a search, or open the settings menu (Figure 1).

The view can be easily configured. You can specify the type and size of the font, as well as the page margins to suit your own taste – if you do not want to use the specifications defined by the ebook's creator. Foliate comes with four different themes,

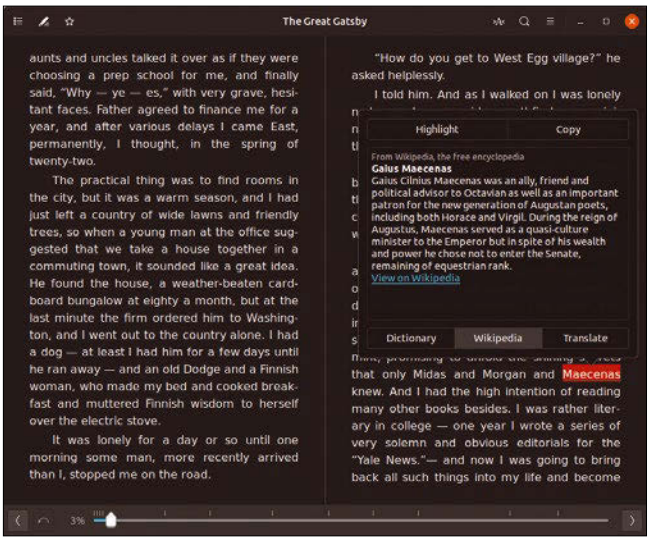


Figure 2: Foliate loads explanations or translations from the Internet for selected words and paragraphs.

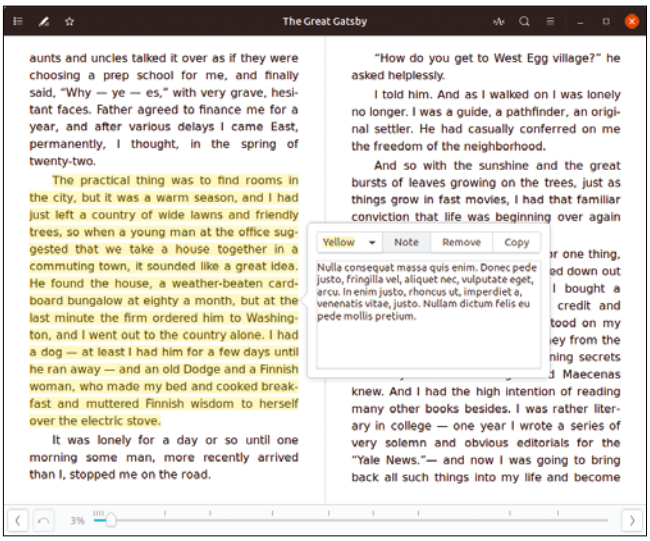


Figure 3: In addition to simple bookmarks, Foliate also stores color highlights and notes.

which you can configure via *Preferences* | *Theme*. There are also three layouts to choose from: *Auto* distributes the contents of the book over one page, or two adjacent pages, depending on the window width; *Single* uses only one page. *Scrolled* finally gets rid of pages and displays a continuous text flow.

If you are reading a book on a laptop, Foliate offers a series of gestures that can be triggered using the touchpad. Use a two-finger swipe to the left or right to scroll one page forward or back. The pinch-to-zoom gesture used on smartphones to zoom in and out also works with Foliate.

Notes

If you select a word in the text, a small window automatically appears; you can now look up the word in a dictionary – this only works for Wiktionary [4] right now. Alternatively, you can look up the term in Wikipedia; Foliate automatically loads the Wikipedia language that matches the language of the desktop environment for this; you also can use Google Translate to translate (Figure 2). In the dialog header you will find an option for copying the contents of the selection to the clipboard (*Copy*) or for highlighting the passage (*Highlight*).

The highlight function starts automatically as soon as you select an entire section or paragraph. Click *Highlight*, and then change the options in the context menu. You can now change the color of the selection or add your own *Note*. Foliate displays all markers with notes in the Notes dialog (or in the corresponding section of the sidebar if enabled in

the preferences) in an overview with colored markers (Figure 3).

Foliate stores your current position as well as the bookmarks and notes specifically for each book below `~/.local/share/` in the `com.github.johnfactotum.Foliate/` directory. If you move this folder to a cloud store such as Nextcloud or Dropbox, the data can easily be synchronized across multiple computers. You can start reading on your desktop computer during the day and continue reading on your laptop later in the evening.

Reader

If you prefer to listening to reading, you can tell Foliate to read aloud to you. For this purpose, the program uses the eSpeak [5], eSpeakNG [6], or Festival [7] speech synthesis tools. All three can usually be installed via the distribution's package manager. I find Festival's voice most pleasant, and the software also supports Spanish, Welsh, Italian, and Finnish.

Foliate automatically detects that the Festival speech synthesis tool is installed on the system. The program then enters the `festival --tts` command for speech output in the settings without any further action on the user's part. If several text-to-speech engines are available on the computer, you can switch between the various programs in the menu.

eSpeak and its unofficial successor eSpeakNG can also be used. You need to install one of the programs via your distribution's package manager, then restart Foliate, open the *Preferences* via the Hamburger menu and enter

```
espeak -vde -s 150
```

or

```
espeak-ng -vde -s 1
```

for eSpeak NG as the *Text-to-speech command* (Figure 4).

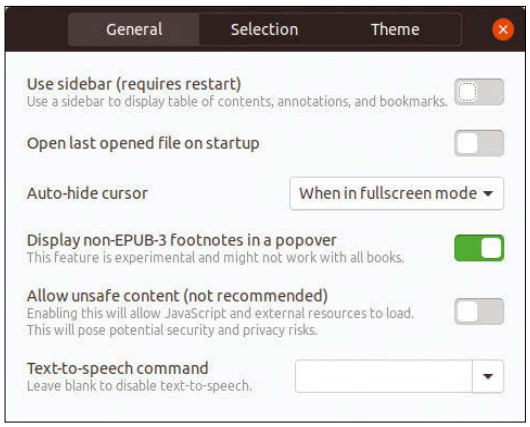


Figure 4: In the settings, you can configure the options for the speech output, among other things.

The optional `-s 150` parameter reduces the speech rate in both cases to a level that is understandable (at least for my ears). The default value is 175 words per minute, which causes speech output to stumble very quickly from one word to the next, very much affecting the intelligibility of the output for untrained ears.

Click on the headphone button in the bottom right corner of the program window to start the read aloud function. The output starts on the current page of the book. Click the button again to stop playback. There is a difference between eSpeak and Festival. Both eSpeak versions stop the speech output immediately, while Festival continues reading until the next sentence.

The fact that Foliate does not automatically orient itself on the book's language for speech output can make the read aloud function slightly inconvenient at the moment. The developers have already received an improvement suggestion to use the ebook's metadata as a guide; however, until this is implemented, you will have to make the changes to the settings manually.

Conclusions

Foliate integrates perfectly with the Gnome desktop as a GTK3 program. Compared to heavyweights like Calibre, the program lacks a way to synchronize ebooks with a hardware reader – but the Foliate project doesn't want to attempt that just yet. The application aims to make reading ebooks on the PC as pleasant as possible and fully meets this requirement.

Besides English, Foliate is localized for languages like Spanish or Italian. For the functions that are still missing, such as the ability to upload ebooks to ebook readers, there are already suggestions for improvements in the project's bug tracker. Support for PDF files is also already being considered. Due to the very large number of changes in the source code at the moment, there will certainly be some interesting new features in the program soon. ■■■

Info

[1] Foliate: *https://johnfactotum.github.io/foliate/*

[2] DEB packages: *https://github.com/johnfactotum/foliate/releases*

[3] Project Gutenberg : *https://www.gutenberg.org*

[4] Wiktionary: *https://en.wiktionary.org*

[5] eSpeak: *http://espeak.sourceforge.net*

[6] eSpeakNG: *https://github.com/espeak-ng/espeak-ng*

[7] Festival: *http://www.cstr.ed.ac.uk/projects/festival*

Animate drawings with Pencil2D
Moving Art

Pencil2D, an easy-to-use painting and 2D animation program, lets you create small animations quickly. Despite the simple user interface, you might need a little help getting started.

BY TIM SCHÜRMANN

If you want a hand-drawn ball to drop elegantly from the ceiling in a YouTube video, Pencil2D [1] is the program for you. Thanks to this drawing program, artists can do more than just paint pretty comic figures, balls, or other objects on virtual paper; they can also animate these objects. If required, the tool imports bitmap images or photos for use as backgrounds.

Pencil2D is primarily intended for cartoon artists, but the software also lets you quickly create animated illustrations or diagrams. All you have to do is add a little of your own artistic talent. An impressive YouTube video [2] shows what professional animators have created with Pencil2D. The program is licensed under the GPLv2; the source code is available from GitHub [3].

Some distributions have Pencil2D in their repositories. Arch Linux users will find the tool in the *pencil2d* package in the Arch Users Repository (AUR). On current Fedora systems, you can fetch and install the package by typing:

```
sudo dnf install Pencil2D
```

Debian and Ubuntu users install the *pencil2d* package at the command line with:

```
sudo apt install pencil2d
```

If the animation program is missing from your distribution's software manager, or if you only find an older version stored there, you can use a Flatpak or AppImage package – most current distributions now support both formats. Pencil2D's Flatpak package is available from the Flathub repository. To retrieve and install it from Flathub, use the first two commands in Listing 1; the third command starts the animation program.

The AppImage is available directly on the Pencil2D website [1]. Click on *Download*, and then select the *Current Stable Version*. Below the penguin icon, select the version that matches your distribution. Then make the downloaded file executable (Listing 2) and launch it.

Start Drawing

When launched, Pencil2D opens the main window shown in Figure 1. In the center, you can see the empty drawing area. In the top left corner, you'll find the drawing tools. The pencil draws thin lines; the brush makes thicker lines.

If you need a straight line, select the Polyline tool, the button that resembles an "M" (Figure 2). The first mouse click defines the starting point; clicking again adds a section. Double-click to end the drawing process. The eraser does pretty much what you would expect; the index finger smudges the area under the mouse pointer when you press the mouse button. The garbage can clears the complete drawing area without prompting.

When painting, a grid can be placed over the drawing area using *View* | *Raster*. You determine the size of the individual boxes under *Edit* | *Preferences* | *General* | *Grid section*. To zoom in or out of the drawing, use the mouse wheel, or hold down Ctrl while pressing the up or down arrow key.

Listing 2: Installing the Pencil2D AppImage

```
chmod +x pencil2d-linux-amd64-<Version>.AppImage
```

Listing 1: Installing the Pencil2D Flatpak

```
$ flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo
$ flatpak install flathub org.pencil2d.Pencil2D
$ flatpak run org.pencil2d.Pencil2D
```

In the *Color Box* window, you can define the drawing color (Figure 3). First select the base color from the wheel and then choose the brightness from the inner square. Alternatively, click *RGB* and use the sliders to mix the three basic colors (red, green, and blue). To avoid having to define a color time and time again, you can add the color by pressing the plus symbol next to the Color Palette. Pencil2D automatically gives the color a name, which you can change by double-clicking on it. You can then select a color from the palette with a single click.

Moving Images

Using the timeline at the bottom, you can breathe life into your drawing. Pencil2D relies on the flipbook principle. An animation consists of a sequence of individual images that the tool somewhat inconsistently also refers to as keys. The following example of a ball dropping demonstrates how quickly an animation can be created in Pencil2D in this way.

First draw a circle at the top of the drawing area. We now want the ball to slowly drop. To do this, create a new frame by selecting *Animation* | *Add Frame* or by pressing the F7 key. You will now see a new empty drawing area, while the vertical red mark moves one position to the right in the timeline.

Using the . and , keys allows you to jump back and forth between the existing images. In the top line of the timeline, Pencil2D numbers the images consecutively. If you click on a number there, you are taken directly to the matching single screen. The *Zoom* slider lets you expand the timeline's width on screen and thus simplify navigation. In any case, the red bar marks the image you are currently viewing and editing in the drawing area. Make sure Pencil2D displays the second frame you just created and the new empty drawing area.

Onion Skin

To drop the ball, draw another circle slightly below the first one. This is not so easy since you can't see the ball from the first picture. To change this, you can press *O*, go to *View* | *Onion skin* | *Previous*, or click on the second button from the right in the top line under Display (see Figure 4).

Now, the first image appears, colored light gray, in the drawing area. Pencil2D refers to this image as the onion skin. To make it more obvious that this is the ball from the previous image, you can color it red by clicking on the icon with the red box under Display. Now draw a new ball slightly below, which is colored black, as shown in Figure 4.

Repeat this procedure until the ball arrives at the bottom of the drawing area. To make your work easier, click on the Select tool (with the dashed rectangle) and draw a frame around the ball. You can grab the selection using the handles at the corners.

Select *Edit* | *Copy* to create a new frame. The selection rectangle should appear automatically.

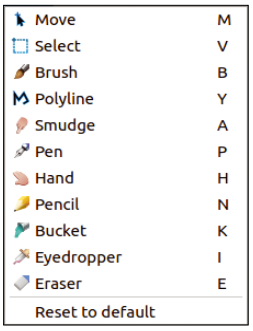


Figure 2: You do not need to navigate the menus to enable the painting and editing tools – all you have to do is press a shortcut key.

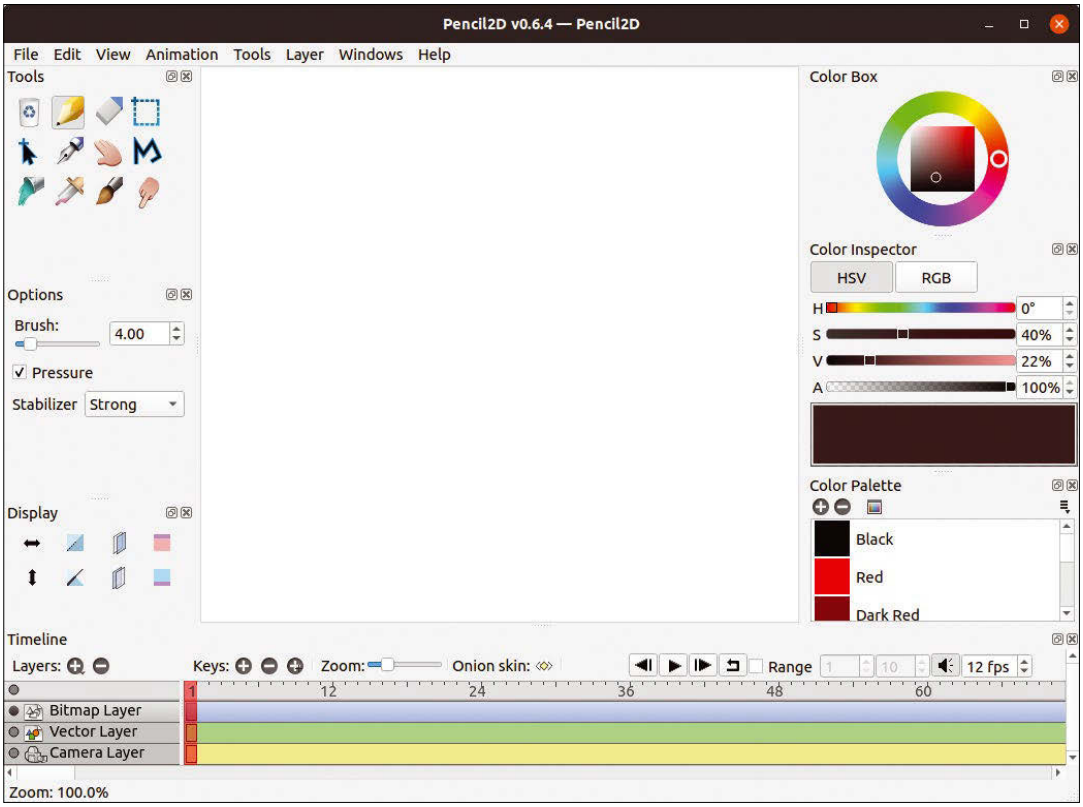


Figure 1: On the left below Options, you will find additional settings for the selected tool (e.g., a brush's stroke width).

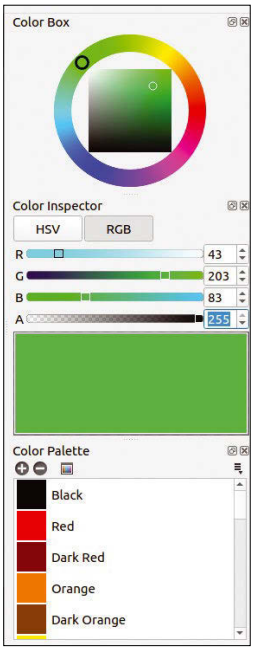


Figure 3: The *A* slider defines transparency. Translucent objects are especially important when using layers.

Move the mouse pointer into the rectangle until it turns into a circle. Now move the box a little to the right using drag and drop. Then use *Edit | Paste* to insert the copied circle at exactly the position shown in Figure 5. You can unselect by clicking anywhere on the drawing area.

When fine-tuning more complex animations, you will often need a preview of the following image. To do this, simply click on the second symbol from the right in the bottom line on the left side under Display. The following images are now displayed as grey shadows. To help distinguish them, you can color them light blue, as shown in Figure 6, by clicking on the blue square icon under Display.

Run It

Pencil2D always displays the onion skins of the last or next five images. To change this, go to *Edit | Preferences* and select *Tools*. To define how many previous images Pencil2D displays, add a number for *Number of previous onion frames shown*; similarly you can define the *Number of next onion frames shown*. (Pencil2D automatically extends the timeline up to 10,000 frames.)

You should now have created several frames where the ball slowly drops from the top to the bottom. To view the animation, select the first frame. Then press Ctrl+Enter or click the right arrow triangle in the timeline. Depending on how many images you have created, the animation may be over fairly quickly. This changes if you enable the curved arrow in the timeline or select *Animation | Repeat*. During playback, Pencil2D will repeat the animation until you press the corresponding button or Ctrl+Enter to stop it again.

The playback speed can be controlled using the *fps* input box on the timeline's right side. By default, the animation runs at 12 frames per second (fps). You can experiment with lower or higher values for your ball. Then make sure that your last frame is visible on the drawing area.

Transparency Report

Many scenes consist of several objects (e.g., the ball could fall on a wall). Pencil2D supports layers so that you do not have to repaint the background in each single image. Layers work like slides, which the tool simply superimposes on top of each other.

To create a new layer for the wall, choose *Layer | New Grid Layer* and assign a name such

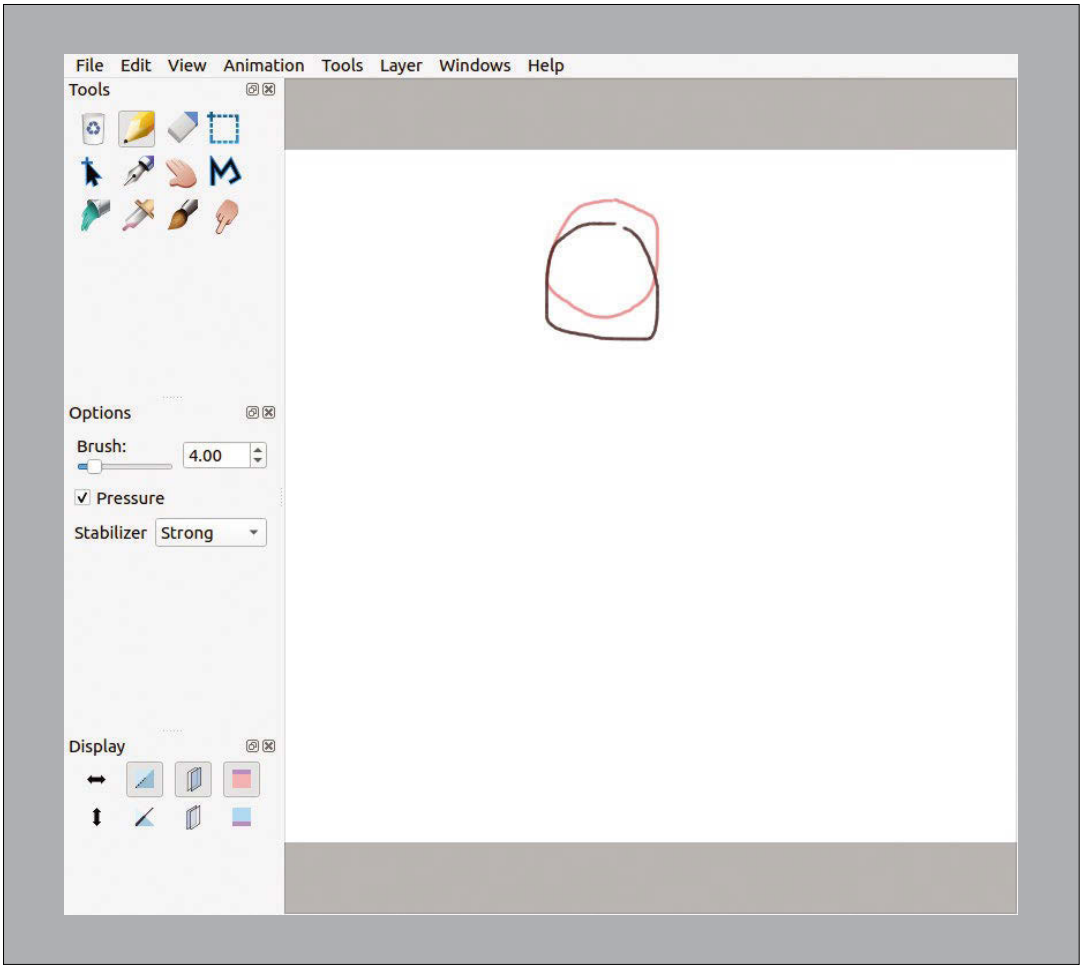


Figure 4: You can see two frames, which Pencil2D will play back one after the other. The black circle resides in the second frame; the circle in the first frame is bright red.

as *Background*. Pencil2D now adds a new line to the timeline. Click the *Background* layer. Everything you paint from now on is assigned to this layer.

Use the Polyline tool to draw a line under the ball from left to right. If you play the animation now, the ball drops onto the line. To include a photo of a real wall, use *File* | *Import* | *Image* to load an image on the currently active layer.

In the timeline, you will find several small boxes, each representing a single image. Pencil2D displays each of them until a new frame follows in the line. Try this out by selecting the middle of the animation with the . and , keys and then creating a new frame with F7. Paint a yellow sun with the brush. The sun will appear in the second part of your animation (Figure 7).

To delete a single frame, select the corresponding layer on the timeline, then select the corresponding frame, and select *Animation* | *Remove Frame*. Unlike clicking on the trash can icon, Pencil2D deletes the entire image rather than just clearing the drawing area.

Stackable

Pencil2D superimposes the drawings in the layers in the order in which they appear in the timeline. In Figure 7, all drawings in the *Background* layer would appear above the ball. You can change the order of the layers by dragging the name of a layer up or down to the desired position. During fine-tuning in particular, it can sometimes help to hide a layer. Simply click on the dot in front of a timeline name.

Thus far in my example, I've worked with bitmap images (i.e., with pixels). Pencil2D also has vector layers. One vector layer is automatically created by Pencil2D; others can be created by selecting *Layer* | *New vector layer*. These layers work like the image layers, but you can only draw lines and geometric shapes. In return, all objects can be moved and distorted individually later. To do this, click on the respective line with the Move tool. The object can be scaled using the handles that then appear and moved to another position using drag and drop (Figure 8).

If you want to share your animation, create a video by selecting *File* | *Export*. The *Movie* sub-

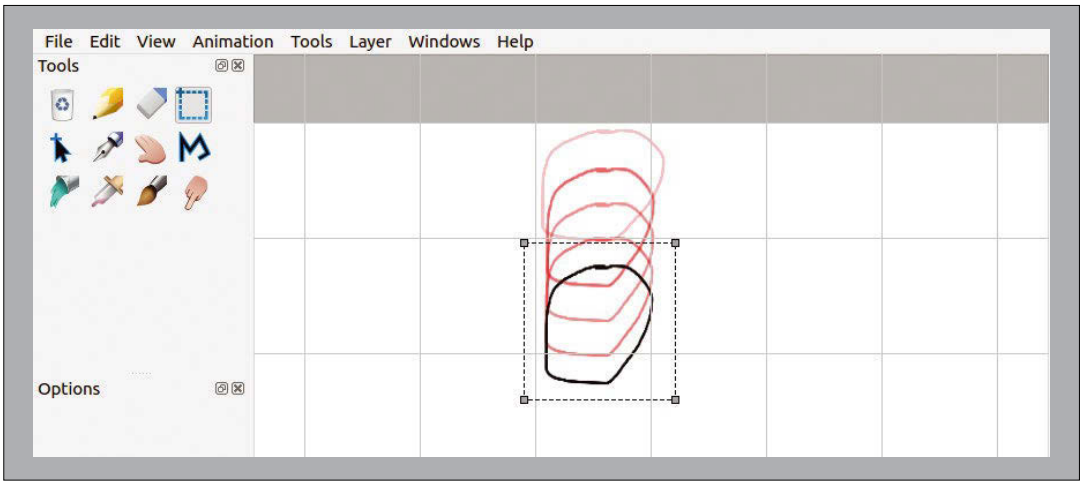


Figure 5: The Select tool helps you copy and move objects across multiple frames.

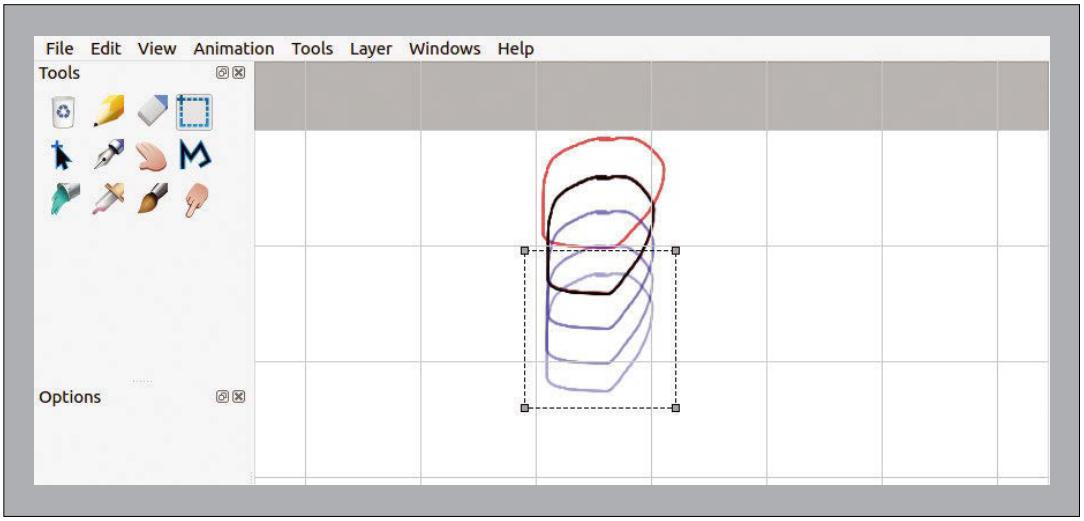


Figure 6: This figure shows a single frame selected from the middle of the animation. The previous circle is bright red, the current circle is black, and the three following circles are light blue. This makes it easy to follow the animation process.

item saves the animation as an MP4 file; alternatively, Pencil2D supports exporting to an animated GIF.

Conclusions

Pencil2D helps users to quickly create small animations. Compared to its competitors, however, the tool still lacks numerous convenient functions, such as circle and rectangle tools. On the other hand, the range of functions is manageable, which means that you can achieve results more quickly. If you are looking to create an animation for the first time, take a look at Pencil2D. The Pencil2D showcase [2] demonstrates what the program can do in the hands of capable artists. ■■■

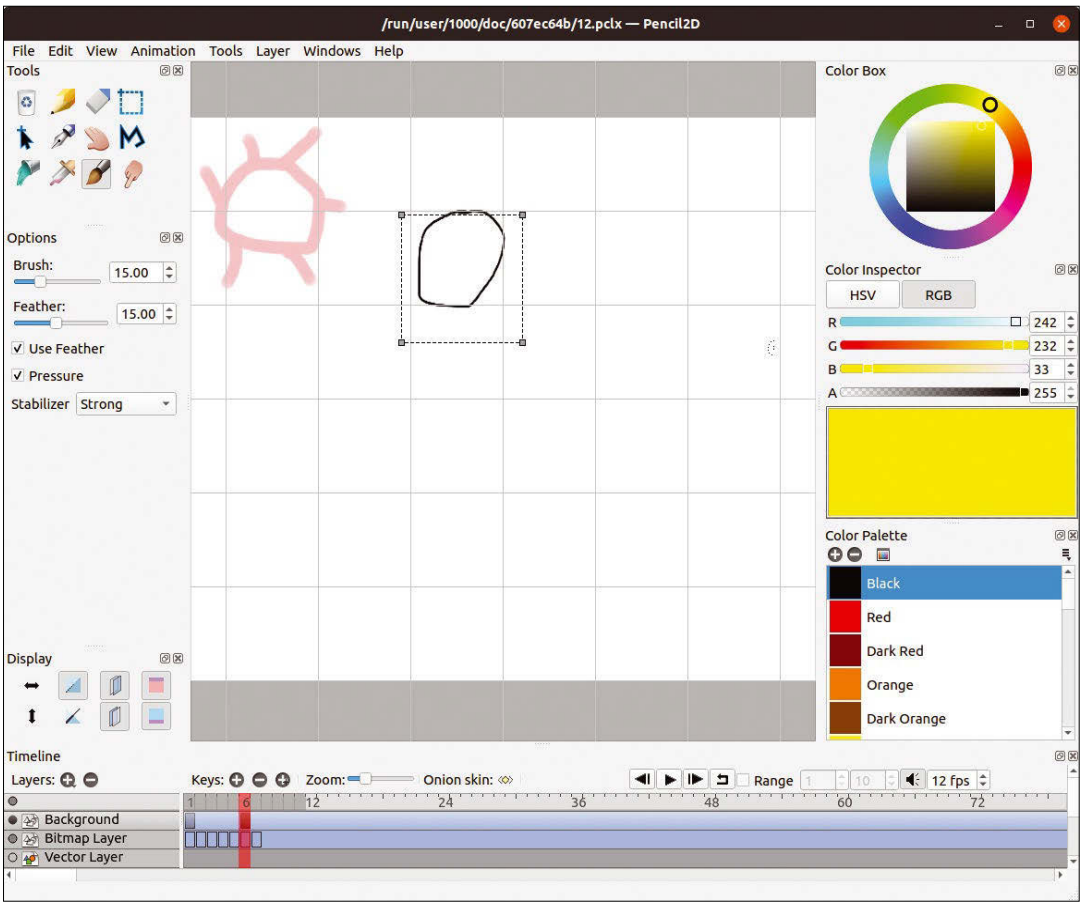


Figure 7: Here the first frame of the background would be seen for four frames. Then Pencil2D switches to the single image with the sun.

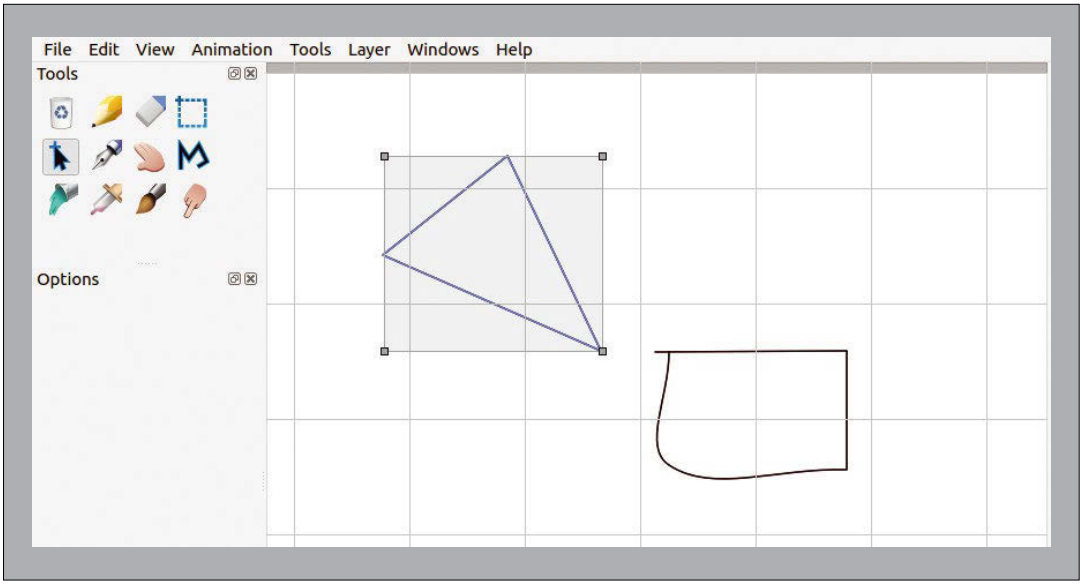


Figure 8: The triangle in the vector layer can be compressed and stretched using the handles.

Info

[1] Pencil2D: *https://www.pencil2d.org*

[2] Pencil2D showcase: *https://www.youtube.com/watch?v=aa9PCu_UZpg*

[3] Pencil2D source code: *https://github.com/pencil2d/pencil*

FOSSPicks

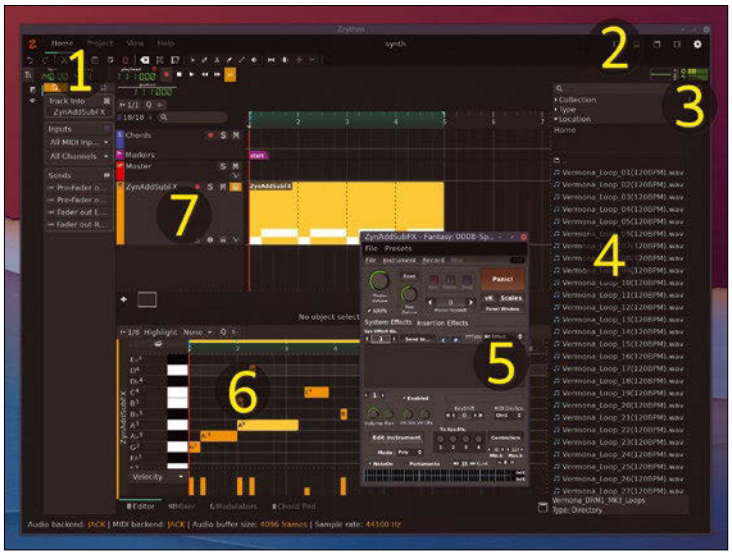
Sparkling gems and new releases from the world of Free and Open Source Software

After building a 3D printer last month, Graham's home is now filled with cup holders, phone holders, cable holders, and tiny PLA boats. BY GRAHAM MORRISON

Digital audio workstation

Zrythm

New audio applications on Linux are few and far between, because creating one is an arduous task that often takes years with little prospect of reward. We've been lucky over recent years with the release of the excellent (and proprietary) Bitwig, Reaper, and Tracktion 7, but there's been very little in the open source space. Which is why Zrythm is such a pleasant surprise, arriving seemingly out of nowhere as an already comprehensive audio recording, mastering, and MIDI sequencing application. It is considered an alpha release by its developer, so this is early days for development. The app does sometimes crash, and it's a little picky when you're not using JACK, but Zrythm is already capable of taking you from composition to completion in an application that feels very much like its commercial counterparts.



1. Audio engine: Use JACK or ALSA for audio and for MIDI, and sync across both. 2. Dynamic UI: Toolbars change, depending on the mode, and each pane can be enabled, disabled, and resized. 3. Media browsers: Much like Bitwig, you can easily switch between views of your content, your plugin list, and the audio outputs. 4. Clips and loops: Drag and drop audio loops, virtual instruments, and effects into the main view to add them. 5. Automation: Almost every parameter within the DAW can be automated. 6. Editor and mixer: Switch between audio and MIDI editors and a mixer view of tracks, inserts, and buses. 7. Track view: Audio, MIDI, and automation data are shown for each track, alongside a unique track for chords, and automation data.

The first thing you notice in Zrythm (other than that the word "rhythm" is obviously a tricky word to spell) is that it has a beautifully modern and dynamic GTK+3 user interface. The application operates like a DAW, such as Cubase, but its fantastic UI design makes it feel more like Bitwig Studio, Ableton, or Renoise. This is because it's split into regions that share similar functionality. The raw track list on the left is where you can add audio and MIDI tracks, as well as groups and a special "chord" track. On the right is the media viewer, which can list LV2 plugins, instruments, and audio files. The plugins themselves are distinguished between audio processors and instruments by their color, and you add plugins to your project by simply dragging them onto the track or into a blank area to create a new track. At the bottom is the clip or section editor, which can be tabbed to show the same kind of vertical mixer you get in Ableton or Bitwig. This then allows you to drag and drop effects or dynamically control the volume of a track across the mix.

Zrythm promises unlimited automation, which means you can adjust the value of almost every parameter over time. Automation data is beautifully drawn in the audio track timeline, as is the audio and MIDI data. These can be edited in the clip view, which is fast and responsive, and feels better integrated than the MIDI editor in something like Ardour, for example. There are also plans for lots of modulation options that will change internal parameters with something like an envelope or an LFO. This could be much like the latest developments in Bitwig, where you connect control signals between different sections of the audio engine. But even without these advanced features, and with the obvious caveat that it's currently too unstable and lacking in certain important features for production work, Zrythm already comes together as a slick package, and it is one of the best ways of composing music and exporting the audio files on Linux.

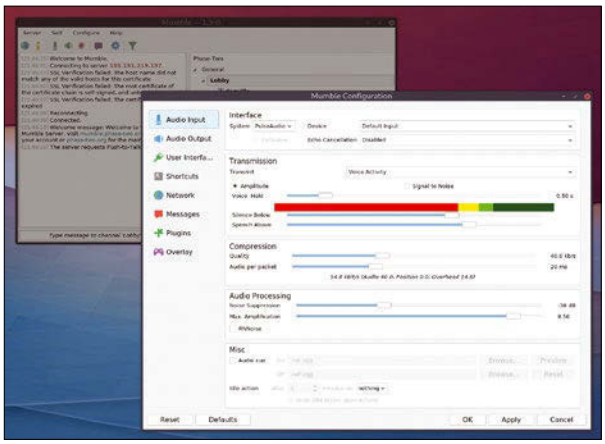
Project Website
https://www.zrythm.org

Voice chat software

Mumble 1.3

It's likely most people will have heard of Mumble. It's a voice chat platform that lets you create ad-hoc groups where everyone can hear and talk to everyone else, and it's been around for a long time. Mumble is good at achieving low-latency and high quality audio across a network, which means conversations sound and evolve more naturally, unlike the consistency and quality concerns that plague proprietary services like Google Hangouts or Skype. Another difference is that because it focuses on audio, it can handle many more consecutive connections, with groups consisting of 100 people being reported to work well. Connections can be encrypted with private/public keys, and much like IRC, there are many free public servers that you can use if you don't want to set up your own. The client allows you to search through these and join any one you like.

Version 1.2 was released in 2009, and thanks to its positional audio, it was perfect to help friends speak to each other whilst playing games. But thanks to its multitrack recording capabilities, it could also help podcasters hear each other and share recordings. Today, Mumble is still being actively developed, and this new release bundles over 3,000 updates. These include new light and dark themes for the user interface and the ability to locally adjust the volume of each stream (talking person) in your channel and to lower their volumes when you're speaking. Another feature that will please long-term users is the addition of toolbar entries for changing transmission modes between voice activation, push to talk, and continuous, as well as hotkey configuration for these modes. This is much easier to use than hunting through the configuration interface, often under pressure when other people can't hear you. Multichannel recordings that include synchronization for each person in the channel is another great addition for podcasters as it means they can edit a recording in a multitrack like Ardour without worrying about phasing or sync issues over longer periods of time.



On Linux, PulseAudio monitor devices can now be used as inputs for Mumble, allowing for all kinds of real-time audio effect fun.

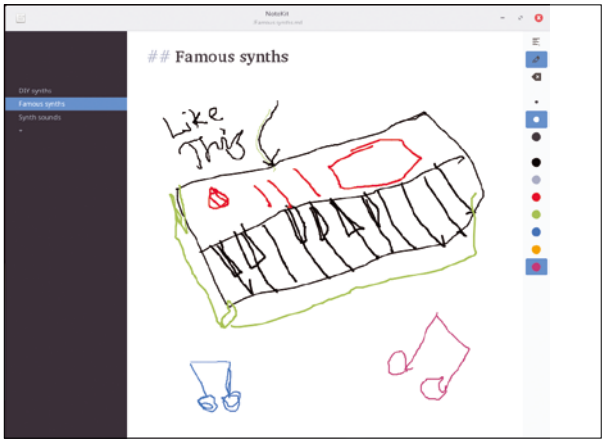
Project Website
https://www.mumble.info

Note taker

NoteKit

The simple text syntax of Markdown is still gaining popularity for all kinds of documentation needs. This is because it's a format that can be easily written and read without any special kind of renderer. But equally, thanks to its ability to define how a document should look and what it should link to, Markdown is often used to document code, websites, books, and technical material. It's also great for taking notes, but there hasn't been a good reason to use a special Markdown editor for this when a simple text editor would do – until now. NoteKit is a GTK+ 2 Markdown editor with a real-time preview that's been designed specifically for note taking. This means that as you type in Markdown, you can add # for titles, ** for bold, * for italics, or numbers for lists, and the edit area will magically updated to preview the style.

There are a few Markdown editors that do the same thing, but there are none that include NoteKit's most impressive feature – you can draw your own notes into the editing window, where they'll appear as hand-drawn annotations. On the right of the slick user interface is a toolbar with shortcuts to the previously mentioned Markdown elements, but also to a small color palette, three brush sizes, and a pencil icon. Using these, you can turn the editor into a simple drawing application. Where you draw, the text you've already entered will realign to make space for your doodles.



Create notes in Markdown and arrange them into hierarchical folders with NoteKit.

It's perfect for lecture notes or making notes about a document, especially if you're lucky enough to have a touch screen. Every edit you make is automatically saved. While the application is currently considered only alpha, it's already relatively stable and usable. With a few updates to include an undo feature and the ability to resize and move images, NoteKit will be a brilliant note-taking tool.

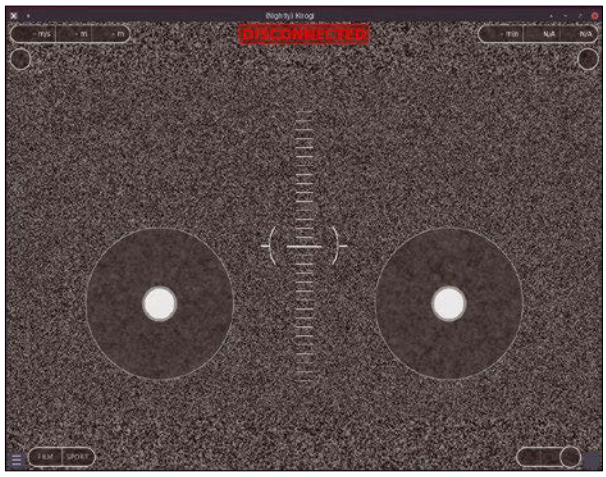
Project Website
https://github.com/blackhole89/notekit/

Drone control

Kirogi

Many modern flying drones come with their own smartphone applications that allow the pilot to control and steer the unit, often through a first person view that's shown on the screen. These applications are often propriety, and you're never sure where your data is being sent or even whether you can trust it to control your drone at all. This is something that Kirogi's developer, Eike Hein, after buying a drone and attempting to fly it during Lunar New Year celebrations in Busan, South Korea. The official app crashed shortly after takeoff, leaving the drone out of control. Luckily (or not!), it was being flown indoors and eventually crashed back to Earth, but not before Eike had resolved to write a new open source application that would allow the drone to be flown from the Linux desktop: Kirogi.

When first launched, you need to choose between the three models currently supported by Kirogi: Ryze Tello, Parrot Anafi, and Parrot Bebop 2. Support for more models is planned, as is support for open protocols like Micro Air Vehicle Link (MAVLink) and the MultiWii Serial Protocol (MSP). After selecting the device, the app will attempt to make a connection and, if successful, open the direct flight control view. Despite Kirogi's alpha state, this functionality already feels mature, and you can use your mouse or touch screen to move both virtual joysticks, switch between drone modes, and see exactly what's happening. The simple HUD also shows altitude, speed, flight time, signal strength, and battery status. If you have a game controller connected, you can even use this. If your drone has a GPS, you can switch to a top-down map view and simply point on a map where you want your drone to go.



If flying your drone from the KDE desktop isn't impressive enough, visit the Kirogi website and check out the excellently designed mascot for the application.

Project Website
https://kirogi.org/

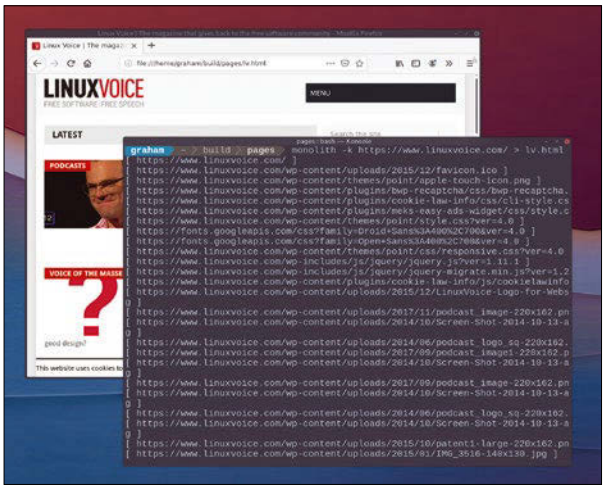
Web archiver

monolith

Back in the days of KDE 2 and 3, before Firefox was a mature project and before Chrome even existed, many of us browsed the web with KDE's own do-everything application, Konqueror. It's well known that Konqueror's web rendering engine, KHTML, was forked and used by Apple in Safari as WebKit, which was forked and became Blink, which became the dominant web rendering layer across the web.

But Konqueror had another brilliant feature – it could save an entire website as a single file, including its CSS and images. This is something you can't even do today, where the average browser restricts page saving to a single broken XHTML file that excludes all of the media assets and online context required to make it readable. Konqueror, by comparison, saved pages as a "web archive," a `.war` file, which was little more than a zipped up collection of everything necessary to view the page, but it worked.

Fast forward a decade or more, and there's been nothing to really compete with web archive files, other than perhaps *web.archive.org*. In theory, HTML5 should make encapsulating an entire page and its assets into a single file easier, which is what `monolith` attempts to do. Rather than being a browser add-on, `monolith` needs to be run from the command line and takes the URL to archive as its only argument. There are options for removing images, excluding JavaScript, ignoring invalid certificates, and setting a custom User-Agent. Other than these options, the command is simple. The only slight quirk is that you need to pipe the output to a file if you don't want your web page to spill onto the terminal, but that's the handy flexibility of the command line rather than a missing feature. Most importantly, `monolith` works, pooling even complex sites into a single file you can then access offline or store for posterity.



In a world of pervasive data, it's still amazingly useful to be able to save a website as a single local file.

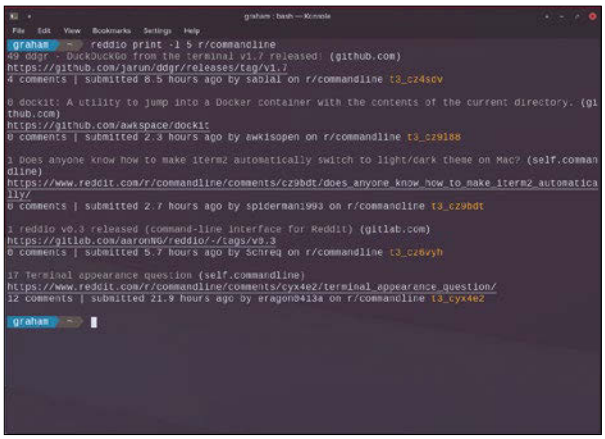
Project Website
https://github.com/Y2Z/monolith

Reddit interface

reddio

One of the best things about the command line is that you don't have the same distractions you have on the desktop. Outside of the task you want to accomplish, there's usually too little space to augment your terminal session with multiple tabs, streaming video, or various chat sessions, even with a multiplexer like `tmux`. Which is why we're reluctant to write about this little command-line gem – a command-line interface to Reddit, the ultimate online time sink. But as we're all going to waste too much time on Reddit anyway, we may as well make it as quick and as accessible as possible. `reddio` does this in a POSIX-compliant way, and offers a very streamlined, and not too distracting, experience. This is because it doesn't work as an interactive session, as you might expect. Instead, you run the command and the top stories are returned, complete with a story synopsis, comment count, and a link, like the contents of an RSS feed. This means it can be cut and modified on the command line to fit your preferences, using commands like `print`, `grep`, `wc`, and `sed` to make sure you only see what you want to see or search for.

Of course, there are additional arguments to add the usual interactions with Reddit that you'd expect. You can login to your account, follow and unfollow subreddits, vote and unvote, and you can even submit your own stories and comment on others. All of this can be accomplished with brief and efficiently constructed commands you run alongside your usual command-line magic. If you don't have the command-line skills to create your own commands, there's plenty of help in the documentation and many different command recipes you can simply copy and paste to make your own. One of the best will return the top five URLs of the month from *r/linux*, for example, while another recipe will return only new messages and comments for a specific sub.



If you're the kind of Reddit user who hates the new site design, you're going to love `reddio`.

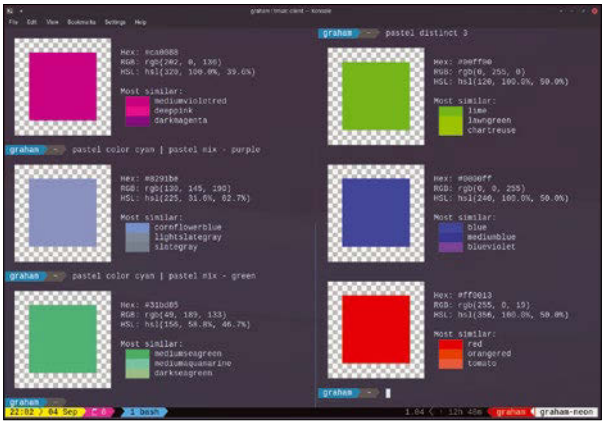
Project Website
https://gitlab.com/aaronNG/reddio

Color editor

pastel

Keeping within the territory of doing one simple job well, especially on the command line, `pastel` is a tool that helps you work with colors. You type `pastel color red`, for example, and a swatch of perfectly rendered color is dropped into your command buffer, complete with checkered background for contrast, Hex, RGB, and HSL values, and a few close color suggestions. Until you start to play with it, you don't realize that there's nothing simple about working with colors. For instance, `pastel` needs to parse and output the plethora of color formats, ranges, and color spaces that most of us would rather leave to Photoshop's small print. These include RGB (sRGB), HSL, CIELAB, and CIELCh alongside ANSI with both 8-bit and 24-bit output. However, what it can do with them is quite magical.

Taking the previous example, you can pipe the output from `pastel color red` into a new `pastel` command with the `mix` argument, such as `pastel mix - blue`. This will generate a swatch of the mixed color, complete with similar color names. You can use the `format` argument to convert one color format to another, show raw colors from their numeric input, list all CSS color names, or even generate any number of colors from which to choose. If you can't find what you're looking for, you can also use `pastel` to pick a color from your screen and take that as the input of the command. All of this is incredibly useful if you're generating colors for print or for CSS. You can even use `pastel` in your own scripts to generate color output instead of the incredibly arcane methods used by most terminals by default.



Turn yourself into a command line Bob Ross by mixing colors and palettes directly from the terminal.

Project Website
https://github.com/sharkdp/pastel

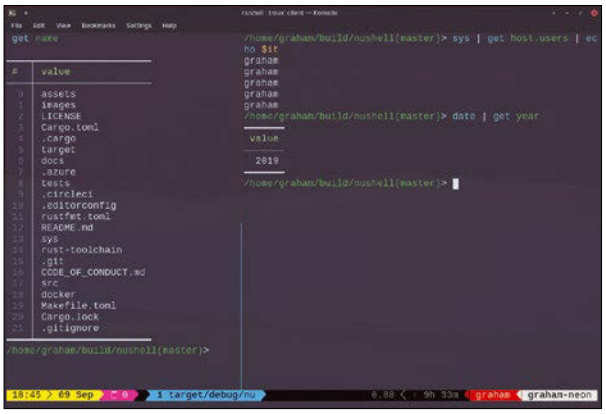
Data-centric shell

Nu Shell

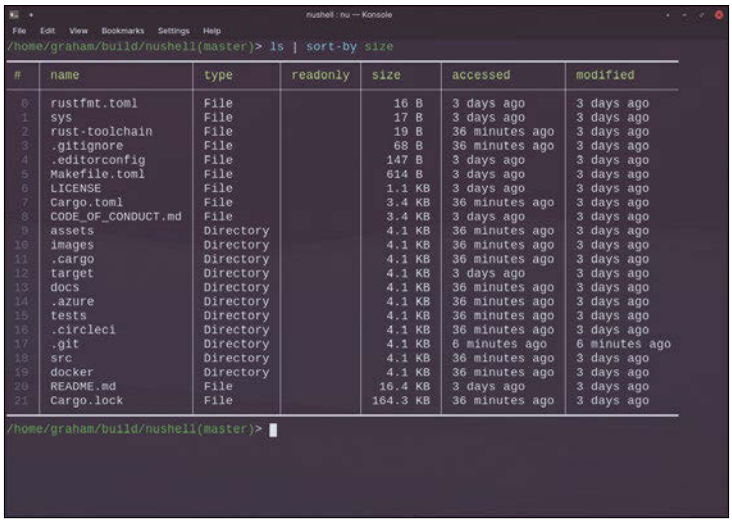
Most of us think the set of terminal environments that are included in most distributions are perennial. They're always there, they seem to have always been there, and they seldom change. People will stick with their chosen environment for decades, perhaps only switching between a distro's default Bash for something more ambitious (such as Zsh) when their needs change. However, all of this doesn't mean there shouldn't be new terminal environments, especially as the computing landscape is completely different from when Bash or Zsh were created. Microsoft's PowerShell has become very popular, for example, especially on a Windows environment when dealing with platforms like .NET and Azure. And Nu Shell attempts to do the same thing for Linux. It's a new shell environment built around the old Unix philosophy of simple commands connected together, built using the modern Rust programming language to better work with a new generation of terminal users, and built to take advantage of data.

As it's built using Rust, Nu can be easily installed via Rust's `cargo` package manager, which means it has great multiplatform support. With it installed, the first thing you're likely to do without even thinking about it is run `ls`, and this is where the Nu's differences first become apparent. Rather than returning the simple file list you might be expecting, the results from `ls` are returned in a formatted ASCII table, with separate columns for name, type, and size. This table output is the default for many of the built-in commands, including `date`, `ps`, and `sys`. It's this structuring of data that makes Nu different from other shells. Nu Shell can parse this data as either simple types like integers, booleans, paths, and string, or more structured types, such as a table row, a binary file, lists, and even chunks of code. You can see this in action with the `open` command, which is a viewer for various data formats. A YAML file is shown as a table, for instance, whereas a text file is simply displayed. Similarly, if Nu understands what constitutes a field within a file, or a field that has been delimited with a known character, those fields can be processed just as you might a single element of output.

To help with this data processing, Nu also includes an expanded pipeline. This works in three parts: an input, a filter, and the output. The excellent documentation includes an example that takes a TOML file as input and uses the filter to search and increment a version field before saving the output to a new file. There's also some great use of color. Command colors are different from argument colors, which are different from the values you may pass to a command. Pipes are colored differently too, and while the colors themselves will depend on your terminal's configuration, this differentiation makes commands easier to understand and helps you avoid mistakes. All of this is bound together with an intuitive autocomplete that helps you deal with the new syntax. It takes some time to learn, but Nu Shell is a great tool if you often find yourself constructing long lines in Bash to deal with esoteric output.



Accessing fields in command output and known file formats is much like querying a database, only your computer and its data are the dataset.



The main difference between Nu Shell and something like Bash is that output has a context that is passed to other commands, such as sorting the output of `ls` by size.

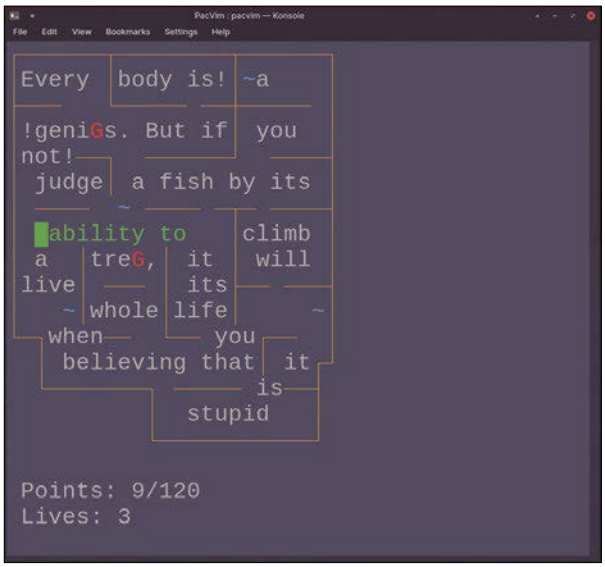
Project Website
https://github.com/nushell

Vim trainer

PacVim

Vim, the terminal-based text editor, has been around for a long time, but in all this time, its steep learning curve has never been associated with entertainment. PacVim almost manages to do this by turning Vim navigation keys into a challenging Pacman-like game that also happens to teach you how to navigate in the venerable editor. The whole game uses Curses to render each level into the terminal using characters, with bars and pipes for the tunnels and alphanumeric characters formed into words as their background. At first glance, these seem like little more than background decoration, but of course, they're not. You're supposed to use Vim's comprehensive word and character navigation keys to move as quickly as you can through the maze because your mission is to turn them all green while avoiding the red *G*s that act as ghosts chasing you.

It's actually difficult to get started, because the game doesn't hold your hand or tell you how you might best approach each level. Much like the arcade game, you need to try different strategies while you slowly get better. Even though the game starts with only two ghosts chasing you, it's still hard. They move in real time, which means there's no pause when you don't press a key. You also have to avoid tilde symbols and the edge of the maze, as these will take one of your lives away, and the corridors themselves are often multiple characters wide, which means you need to move through every character. The only way to really beat the game is to master the Vim keys that move you through words and up columns quickly, which if you can master this means you've also mastered Vim.



Alongside Vim's normal up, down, left, and right, you can also use commands like `w`, `E`, `$`, and `gg` to navigate the maze.

Project Website
https://github.com/jmoon018/PacVim

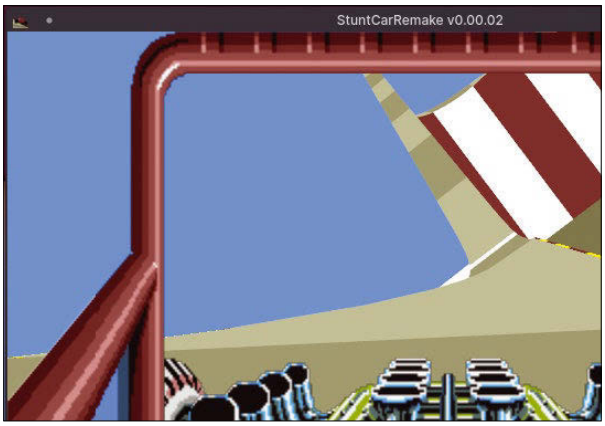
Driving game

Stunt Car Racer Remake

Geoff Crammond is one of those 1980s bedroom programmers who was able to escape the confines of 8- and 16-bit home computer hardware and forge a hugely successful career in the 100 billion dollar game industry he helped to create. He's best known for the Grand Prix series of racing games that changed our ideas of what a driving simulator could be, thanks to their exacting detail, brilliant 3D performance, and a burgeoning series of real life Formula 1 seasons those games replicated. But what's most amazing about Geoff Crammond was that he was a playability polymath. Long before Grand Prix, he created an incredible first-person puzzle game called The Sentinel. This game holds up even today and is aching for a virtual reality reboot. He also wrote Revs, a 3D racer for 8-bit hardware. But Mr. Crammond also wrote another genre defining title, another driving game called Stunt Car Racer.

What made Stunt Car Racer so incredible was its audacity and addictiveness. Its audacity came from being a first person driving game for 8- and 16-bit computers (the Amiga had the best version), and for dropping the player into a drag-racing car on a track that has more in common with a roller-coaster than a Grand Prix circuit. There are jumps, huge climbs, and even a loop-the-loop. The modern Trackmania doesn't come close. All of this was tied together with a driving mechanism that ached to be played. The more you crashed, or the harder you hit corners, the closer your vehicle got to destruction, and your progress depended on winning each race against an increasingly hostile AI. It was amazing and very difficult to play authentically on modern hardware. Until now, that is: Stunt Car Racer Remake is an utterly faithful remake of a classic you can play on your Linux box, which is what you should do right now.

Project Website
https://github.com/ptitSeb/stuntcarremake



This remake of Stunt Car Racer is brilliant, but it would be even better if it had the original's two player serial connection.

FEATURED EVENTS

Users, developers, and vendors meet at Linux events around the world. We at *Linux Magazine* are proud to sponsor the Featured Events shown here.

For other events near you, check our extensive events calendar online at *http://linux-magazine.com/events.*

If you know of another Linux event you would like us to add to our calendar, please send a message with all the details to *events@linux-magazine.com*.

Devopsdays Berlin

Date: November 27-28, 2019

Location: Berlin, Germany

Website: *https://devopsdays.org/events/2019-berlin/welcome/*

Devopsdays is a worldwide series of technical conferences covering topics of software development, IT infrastructure operations, and the intersection between them. Each event is run by volunteers from the local area. The event features a combination of curated talks and self-organized open space content.

Black Hat Europe 2019

Date: December 2-5, 2019

Location: London, United Kingdom

Website: *https://www.blackhat.com/eu-19/*

Black Hat provides attendees with the latest in research, development, and trends in Information Security. Here the brightest professionals and researchers in the industry come together for two days of technical hands-on training, followed by two days of the latest research and vulnerability disclosures.

Open FinTech Forum

Date: December 9, 2019

Location: New York, New York

Website: *https://events19.linuxfoundation.org/events/open-fintech-forum-2019/*

Focusing on the intersection of financial services and open source, Open FinTech Forum will provide CIOs and senior technologists guidance on building internal open source programs as well as an in-depth look at cutting-edge open source technologies.

Events

Linux App Summit	November 12-15	Barcelona, Spain	https://linuxappsummit.org/
Linux Presentation Day 2019	November 16	Cities across Europe	http://l-p-d.org/:en:start
SC19	November 17-22	Denver, Colorado	https://sc19.supercomputing.org/
KubeCon + CloudNativeCon North America 2019	November 18-21	San Diego, California	https://events19.linuxfoundation.org/events/kubecon-cloudnativecon-north-america-2019/
DevOpsDays Berlin 2019	November 27-28	Berlin, German	https://devopsdays.org/events/2019-berlin/welcome/
Blackhat Europe 2019	December 2-5	London, United Kingdom	https://www.blackhat.com/eu-19/
Open FinTech Forum	December 9	New York, New York	http://www.linuxpromagazine.com/Resources/Event-Calendar
IT Tage 2019	December 9-12	Frankfurt, Germany	https://www.ittage.informatik-aktuell.de/
Node+JS Interactive 2019	December 11-12	Montreal, Canada	https://events19.linuxfoundation.org/events/nodejs-interactive-2019/
Kubernetes Forum Sydney 2019	December12-13	Sydney, Australia	https://events19.linuxfoundation.org/events/nodejs-interactive-2019/
Open Source Forum 2019	December 16	Tokyo, Japan	https://events19.linuxfoundation.org/events/open-source-forum-2019/
36C3 - Chaos Communication Congress	December 27-30	Leipzig, Germany	https://events.ccc.de/
Fosdern 2020	February 1-2	Brussels, Belgium	https://fosdem.org/2020/

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Copy Editors
Amy Pettle, Megan Phelps

News Editor
Swapnil Bhartiya

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
© Svetlana Borovkova, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 3090 5128

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
2721 W 6th St, Ste D
Lawrence, KS 66049 USA

Publisher
Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America

www.linux-magazine.com – Worldwide

Printed in Nuremberg, Germany by hofmann infocom GmbH on recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 2721 W 6th St, Ste D, Lawrence, KS, 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 2721 W 6th St, Ste D, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany.

Authors

Konstantin Agouros	34
Ulrich Bantle	20
Pieter Barendrecht	24
Erik Bärwaldt	56, 76
Swapnil Bhartiya	8
Zack Brown	12
Bruce Byfield	42, 60
Joe Casad	3
Mark Crutch	69
Jon "maddog" Hall	70
Charly Kühnast	45
Christoph Langner	80
Vincent Mealing	69
Graham Morrison	90
Mike Schilli	50
Tim Schürmann	16,46, 54, 84
Mayank Sharma	28
Johan Thelin	64
Ferdinand Thommes	72
Rick Timmis	38

NOW PRINTED ON recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

United States Postal Service Statement of Ownership, Management, and Circulation

1. Publication Title: Linux Pro Magazine
2. Publication No.: 1752–9050
3. Filing Date: 10/1/19
4. Issue Frequency: Monthly
5. No. of Issues Published Annually: 12
6. Annual Subscription Price: $124.95
7. Complete Mailing Address of Known Office of Publication: 2721 W 6th, Suite D, Lawrence, KS 66049;
Contact Person: Brian Osborn; Telephone: 785–856–3080
8. Complete Mailing Address of Headquarters or General Business Office of Publisher:
2721 W 6th, Suite D, Lawrence, KS 66049
9. Full Names and Complete Mailing Addresses of Publisher, Editor, and Managing Editor:
Brian Osborn, Publisher, 2721 W 6th, Ste D, Lawrence, KS 66049
Joe Casad, Editor, 2721 W 6th, Ste D, Lawrence, KS 66049
Lori White, Mng Editor, 2721 W 6th, Ste D, Lawrence, KS 66049
10. Owner: Linux New Media USA, LLC, 2721 W 6th, Ste D, Lawrence, KS 66049; Stockholders: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany
11. Known Bondholders, Mortgages, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages, or other Securities: None
12. Tax Status: Has not changed during preceding 12 months
13. Publication Title: Linux Pro Magazine
14. Issue Date for Circulation Data: October 2019
I certify that all information furnished on this form is true and complete. Brian Osborn, Publisher, 10/1/19

	15. Extent and Nature of Circulation	Avg. No. Copies Each Issue During Preceding 12 Months	No. Copies of Single Issue Published Nearest to Filing Date
b. Paid Circulation	a. Total Number of Copies (Net Press Run)	6203	5692
	b. (1) Paid Outside-County Mail Subscriptions	995	961
	b. (2) Paid In-County Subscriptions	0	0
	b. (3) Sales Through Dealers & Carriers, Street Vendors, Counter Sales	2469	2160
	b. (4) Other Classes Mailed Through the USPS	10	1
	c. Total Paid Distribution	3474	3122
d. Free Distribution	d. (1) Outside-County	0	0
	d. (2) In-County	0	0
	d. (3) Other Classes Mailed Through the USPS	0	0
	d. (4) Free Distribution Outside the Mail	71	150
	e. Total Free Distribution	71	150
	f. Total Distribution	3545	3272
	g. Copies Not Distributed	2659	2420
	h. Total	6203	5692
	i. Percent Paid Circulation	98%	95%

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to *edit@linux-magazine.com*.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Approximate On Sale Date

UK / Europe	Nov 30
USA / Canada	Dec 27
Australia	Jan 27

Issue 230 / January 2020

Home Automation

The whole world is fascinated with home automation. Hundreds of tools and devices are flooding the market to open your curtains, turn on your lights, start your toaster, and more. Next month we'll report on some home automation innovations that will pique the interest of do-it-yourself Linux users, including Mozilla Webthings and the FHEM home automation server.

Image © higyou, 123RF.com

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***

Harness the power of Linux!

Want your friends and colleagues to make the switch to Linux?

This single issue shows beginners how to:

- install Linux
- download and install free software for your Linux system
- play games
- create documents and spreadsheets
- process photos
- play music and videos
- and much more!

GETTING STARTED WITH LINUX

ORDER ONLINE:
shop.linuxnewmedia.com/specials

HETZNER

DEDICATED ROOT SERVER EX62

High speed performance with the new **Intel Core i9-9900K octa-core** processor

ENTERPRISE HDDs or NVME SSDs

Dedicated Root Server EX62

- ✔ Intel® Core™ i9-9900K Octa-Core incl. Hyper-Threading-Technology
- ✔ 64 GB DDR4 RAM
- ✔ 2 x 8 TB SATA Enterprise Hard Drive 7200 rpm
- ✔ 100 GB Backup Space
- ✔ Traffic unlimited
- ✔ Location Germany or Finland
- ✔ No minimum contract
- ✔ Setup Fee $78

monthly from $ **72.50**

Dedicated Root Server EX62-NVMe

- ✔ Intel® Core™ i9-9900 Octa-Core incl. Hyper-Threading-Technology
- ✔ 64 GB DDR4 RAM
- ✔ 2 x 1 TB NVMe SSD
- ✔ 100 GB Backup Space
- ✔ Traffic unlimited
- ✔ Location Germany or Finland
- ✔ No minimum contract
- ✔ Setup Fee $78

monthly from $ **72.50**

www.hetzner.com