FREE DVD

Double-Sided DVD
INSIDE!

NAVI
Interactive cheat sheet for the shell commands

EDGE COMPUTING

LINUX PRO MAGAZINE
ISSUE 234 – MAY 2020

EDGE COMPUTING

Get ready for the low-latency cloud

Bash Scraping
Gather, parse, and filter web data with Bash commands

Free as in Freedom
Which Linux distros meet the strict FSF definition of "free?"

Pi Tricks
Turn your Pi Zero into a USB stick

chromedp
Extract data from dynamic web pages

Helium
Lean Linux distro for 32-bit systems

LINUXVOICE

- FreeDOS: Is this FOSS version of MS-DOS the ultimate retro project?
- Maddog: Courage and imagination
- FlightGear Flight Simulator: Slip the surly bonds of earth

FOSSPicks
- OpenShot 2.5
- vokoscreenNG
- OpenSnitch

Tutorials
- Printing in the Shell
- LÖVE Animation

LINUX NEW MEDIA
The Pulse of Open Source

Secure and Private

All systems by TUXEDO Computers come ready to start with Linux. In addition to impressive performance, they offer comprehensive protection of your personal data and strong protection of your privacy.

Privacy+
Intel ME, webcam, audio and WiFi deactivatable

Fully encrypted
System and data completely protected against unauthorized access

Zero Spyware
Verifiable security thanks to Open Source software

Automatic Installation
System reset thanks to web-based, fully automatic installation

100% Linux

5 Year Warranty

Lifetime Support

Built in Germany

German Privacy

Local Support

TUXEDO COMPUTERS

tuxedocomputers.com

HOME ALL DAY

Dear Reader,

I don't know what world you are living in as you read this, but the world that I'm living in while I write this (three to eight weeks earlier than your world, depending on where you live) is busy with reinventing itself to contend with the coronavirus pandemic. Stores and restaurants are closed, schools are canceled, and offices are shutting their doors as employees make plans to "shelter in place."

It seems a triumph of our 21st century civilization that so much of our life can move so suddenly and gracefully online. Community groups, book clubs, yoga classes, university classes, churches, and therapy sessions are all transitioning to online only. Still more significant are the millions of office jobs that are suddenly relocating.

As a society, we have been working on these tools for supporting remote work for more than a generation. The pieces are all in place – we just need to escalate the scale of the home office revolution. We also need to extend it to people who never really thought of themselves as home-office workers.

Most office networks are managed in a systematic way – either by a professional admin or at least through a process that has had some professional attention. Home networks, on the other hand, vary widely and support an astonishing variety of activities: games, music downloads, and social media, with mysterious websites popping up in search windows and spam email entering without the usual filters employed at the office. As we all prepare to work from home, keep in mind that Internet intruders and miscreants are well aware of this sudden migration and are hard at work right now devising ways to exploit it.

Here at my home office, with my cat crawling over my laptop keyboard (I think she's hungry), I jotted down a few important considerations for anyone who suddenly finds their home network is now their work network:

- *Router* – home routers are notoriously insecure. Many need security updates and bug fixes. Others are altogether out of date and can't even be patched anymore. The erroneous security-by-obscurity protection that passed without objection on your home network won't apply anymore if your home becomes a site for financial records, employee records, and business plans. Find the password, log in, and make sure your router is up to date.
- *Backup* – most offices have a system for regular backups. Will that system follow you home? If not, what are you going to do about it? Cloud backup is fairly easy to obtain now, but it still seems haphazard, and possibly risky, for every employee to act independently. Think about defining a coherent and consistent backup policy for all your organization's home users.
- *VPNs* – another technology that has been around for years but that will receive lots of renewed attention with the tsunami of home connections. If the VPN to your corporate network is an occasional-use thing that was set up a few years ago, be aware that the state of the art for Internet encryption has changed significantly in the past few years, and some technologies that used to be considered secure are now on the outs with the experts. For instance, Transport Layer Security (TLS) 1.0 and 1.1 are now deprecated, and other protocols that used to power VPNs of the past are also suspect. Be sure your VPN uses secure and contemporary encryption.
- *Video chat* – Tools like Zoom, Skype, and Microsoft Teams are the superstars of the pandemic era. Users all over the world are scheduling work sessions, doctor's appointments, and strategic planning retreats through video conferencing. As powerful as these tools are, be aware that they also have raised security concerns in recent months [1] [2]. I'm sure they are fine for water cooler talk, but if you really need to keep secrets, know their limitations.

A daunting list, and this is only the beginning. If you're lucky, your company's IT department will assist with ensuring your work-from-home configuration is as secure as your office configuration. (Of course, if you are reading this magazine, maybe you *are* your company's IT department.)

If your workplace has shifted to home, don't forget to attend to these details. But also don't forget that not everyone has the privilege of bringing their job home. In these turbulent times, remember those among us who find themselves suddenly out of a job and scrambling for shelter and grocery money.

Help out if you can, and hunker down. We'll get through this.

Joe Casad,
Editor in Chief

Info

[1] New Zoom Security Warning: Your Video Calls at Risk from Hackers – Here's What You Do:
https://www.forbes.com/sites/zakdoffman/2020/01/28/new-zoom-roulette-security-warning-your-video-calls-at-risk-from-hackers-heres-what-you-do/#6017dfb27343

[2] Microsoft: Skype for Business Server Spoofing Vulnerability:
https://fortiguard.com/encyclopedia/endpoint-vuln/61494/microsoft-skype-for-business-server-spoofing-vulnerability

WHAT'S INSIDE

The Edge is a popular buzz word in high-tech news, but what does it mean really? We introduce you to an exciting new technology that could be changing the way we think about the cloud.

Also in this issue:

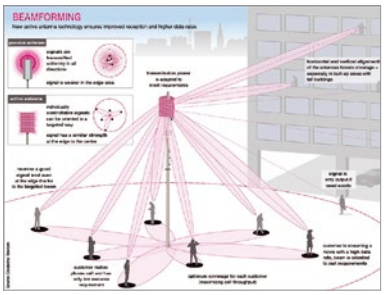
- **navi** – interactive cheat sheet for easy help on shell commands (page 38)
- **chromedp** – extract data from complex dynamic web pages (page 42)

Check out MakerSpace for a description of the versatile CircuitPython runtime environment, and turn to Linux-Voice for studies on the FreeDOS retro OS and printing in the shell.

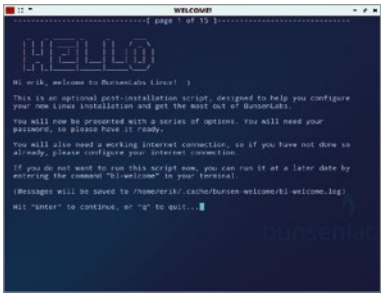
SERVICE

NEWS

COVER STORIES

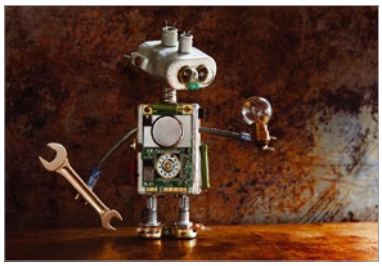


REVIEWS

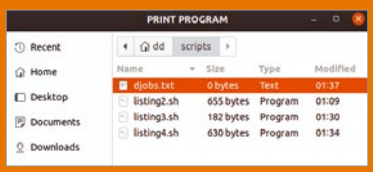


IN-DEPTH

MAKERSPACE



LINUXVOICE



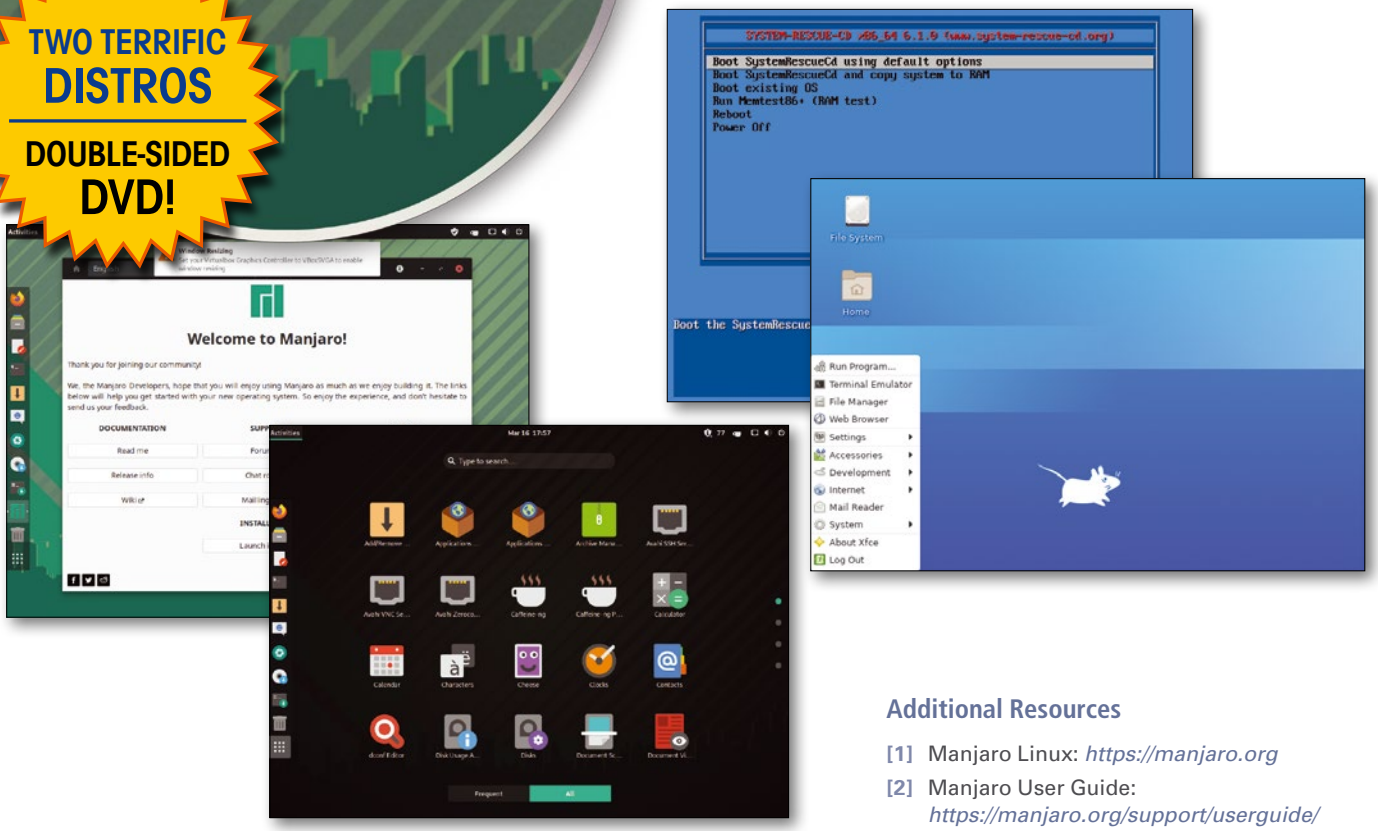
On the DVD

Manjaro 19.02 Gnome Edition

Manjaro is a full-featured Linux for those who love the simplicity and independence of Arch but prefer a user-friendly desktop. The Arch-based Manjaro on this month's DVD comes with a complete Gnome desktop environment. The developers have also seen to many of the details that are left undone in the minimal yet mighty Arch, resulting in a system that offers ease for beginners yet provides support for key Arch components and packages.

SystemRescueCd 6.1

This steady system rescue disc includes a useful collection of utilities for fixing impaired Linux and Windows computers. Onboard you'll find tools for repairing partitions and filesystems, archiving data, and editing broken config files. SystemRescueCd boots from the DVD and comes with support for all major filesystems (ext3/ext4, xfs, btrfs, reiserfs, jfs, vfat, ntfs), as well as network filesystems such as Samba and NFS.



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] Manjaro Linux: *https://manjaro.org*
[2] Manjaro User Guide: *https://manjaro.org/support/userguide/*
[3] Manjaro Forum: *https://forum.manjaro.org/*
[4] SystemRescueCd: *http://www.system-rescue-cd.org/*
[5] SystemRescueCd Manual: *http://www.system-rescue-cd.org/manual/*

openSUSE + LibreOffice Conference

A Conference for
Open Source Developers
October 13 - 16
Nuremberg, Germany

events.opensuse.org

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

Zorin OS 15.2 Now Available

The latest iteration of Zorin OS has arrived. Version 15.2 is now available, which focuses on software bugs, security, and expanded hardware compatibility. This newest version might also serve as the last release until the developers of Zorin OS unleash their upcoming Zorin Grid, which gives admins the ability to manage massive desktop rollouts from a web-based dashboard.

Since the last release of Zorin OS (Summer of 2019), the OS has been downloaded over 900,000 times and has been included in numerous "best of" lists for desktop distributions. So this latest version should be seen as yet another step forward for the open source operating system.

Zorin OS 15.2 is based on Ubuntu 18.04.4, so it includes the 5.3 version of the Linux kernel. The .2 release also adds support for AMD Navi GPUs (including the Radeon RX 5700), as well as support for 10th gen Intel processors and newer model MacBook and MacBook Pro keypads and touchpads.

Users will also find updated versions of core apps, such as Firefox, LibreOffice, and GIMP.

You can download four different versions of Zorin 15.2:

- Lite – Best suited for older hardware and includes LibreOffice, the XFCE desktop, and the standard desktop layouts.
- Core – Best suited for everyday desktop usage and includes LibreOffice, standard desktop layouts, the GNOME desktop, and Zorin Connect (to sync your phone with your desktop).
- Education – Best suited for education environments and includes LibreOffice, both GNOME and XFCE desktops, Zorin Connect, standard desktop layouts, and educational games and apps.
- Ultimate – Best suited for business desktops and includes LibreOffice, both the GNOME and XFCE desktops, Zorin Connect, standard and advanced desktop layouts, business and media apps, over 20 games, and Zorin installation support.

Download your copy of Zorin OS 15.2 now: *https://zorinos.com/download/*.

Firefox to Get an Additional Sandbox Layer

The Firefox web browser already runs on top of a sandbox which separates the browser from the operating system. But with attack vectors growing more and more sophisticated (and many shared libraries not up to modern security demands), the Mozilla developers decided it was time to take the isolation of the browser further.

With the release of Firefox 74, a new sandbox technology, called RLBox, will be added. RLBox was developed as a joint effort between Mozilla, the University of California San Diego, the University of Texas at Austin, and Stanford University.

According to Bobby Holley, principle engineer with Mozilla, RLBox is a "big deal." With this new sandbox layer, it's easy to isolate existing chunks of code at an unheard of granularity. With RLBox in place, the

Firefox developers are able to separate third-party libraries from the Firefox core engine. By making this separation, bugs and exploits within third-party libraries will be unable to impact other applications that use the same library.

The team involved with RLBox managed to isolate half a dozen libraries. Initially, Firefox will ship with the Graphite font shaping library, which is used to correctly render complex fonts, protected with RLBox. Eventually the same sandboxing technique will be applied more broadly to ensure browsing with the open source tool is as secure as possible.

Original source: *https://hacks.mozilla.org/2020/02/securing-firefox-with-wasm/*

Microsoft Defender ATP is Coming to Linux

Microsoft started out by announcing it would release the new Edge web browser for Linux. Next came MS Teams. Continuing that cross-platform effort, Microsoft is set to release Microsoft Defender ATP for the open source platform. Microsoft's stated goal was to build security solutions "not only for Microsoft, but from Microsoft."

According to many MS customers, they've had to deal with attack vectors across a range of platforms and products. That includes Linux. And with the continued rise of Linux on Azure, it became clear to Microsoft they'd need to offer a security solution for more than just Windows and mac OS.

What is Microsoft Defender ATP? According to Microsoft, it is "...a unified endpoint platform for preventative protection, post-breach detection, automated investigation, and response." In other words, Defender ATP is an enterprise-grade security solution that goes beyond the standard antivirus service. Unlike the standard Windows Defender, ATP works on behavioral analysis to collect usage data and store it on the same system. When Defender ATP notices inconsistent behavior, it sends the data to a service that compares it to collected data, and then offers up advice or solutions.

Microsoft Defender ATP for Linux will be made available to Microsoft customers some time in 2020. However, the public preview should be open soon and will be available to install on RHEL 7+, CentOS Linux 7+, Ubuntu 16 LTS or higher, SLES 12+, Debian 9+, and Oracle EL 7.

Microsoft Defender ATP for Linux will include both command line and GUI tools.

Original source: *https://techcommunity.microsoft.com/t5/microsoft-defender-atp/microsoft-defender-atp-for-linux-is-coming-and-a-sneak-peek-into/ba-p/1192251#*

South Korean Government Considers Move to Linux Desktop

The South Korean Government is on the verge of migrating from Windows 7 to Linux on the desktop. This began back in May 2019, when South Korea's Interior Ministry announced the plans to look into making the switch.

Since that initial date, the South Korean Ministry of Strategy and Planning announced the government is now exploring migrating over three million Windows 7 desktops over to Linux. According to Choi Jang-hyuk (head of the Ministry of Strategy and Finance), South Korea will resolve their dependency on Microsoft while reducing the budget by migrating to an open source operating system.

What is driving this migration? It's primarily a financial decision. The cost of migrating so many desktops from Windows 7 to Windows 10 would reach over $650 million dollars. With the Ministry of National Defense and the National Police Agency already using Harmonica OS 3.0 (based on Ubuntu Linux 18.04), and the Ministry of Public Administration and Security using Gooroom Cloud OS (based on Debian), the choice to make the nation-wide switch to Linux made perfect sense.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

Building and Running Containers
• Jeff Layton
Best practices for building and running Docker and Singularity containers.

Initial Best Practices for Containers
• Jeff Layton
Consider some best practices before diving into containers.

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

Hybrid Public/Private Cloud
• Konstantin Agouros
Extending your data center temporarily into the cloud during a customer rush might not be easy, but it can be done, thanks to Ansible's Playbooks and some AWS scripts.

Transparent SIP communication with NAT
• Mathias Hein
We show you how to secure transparent IP address transitions through NAT firewalls and gateways for Voice over IP.

Prowling AWS
• Chris Binnie
Prowler is an AWS security best practices assessment, auditing, hardening, and forensics readiness tool.

Once the migration officially begins, the Korean Postal Service will be moving to Korean-based TMaxOS, which includes a unique desktop environment and uses ToGate, a Chromium-based web browser.

Although this may be nothing more than a bid to get Microsoft to offer South Korea a significant discount for a Windows 7 to 10 migration, until that comes to fruition, it looks as though the move to Linux is happening.

Source: *https://www.fosslinux.com/29117/south-korea-switching-their-3-3-million-pcs-to-linux.htm*

OpenSSH Now Supports FIDO/U2F Security Keys

OpenSSH is, by far, the single most popular tool for logging into remote servers and desktops. SSH logins are generally considered fairly safe, but not 100 percent. If you're not satisfied with the out of the box security offered by OpenSSH, you can always opt to go with SSH key authentication. If that's not enough, there's also 2 Factor Authentication, which would then require you to enter a PIN generated by an application such as OTPClient (*https://github.com/paolostivanin/OTPClient*) or Authy. (*https://play.google.com/store/apps/details?id=com.authy.authy*)

Image © Kirill Makarov, 123RF.com

As of OpenSSH 8.2, there's a newly supported option, FIDO/U2F security keys. What this means is that you can now use 2FA hardware keys (such as the Yubi Key [*https://www.yubico.com/*]) to authenticate your SSH login attempt.

2FA is often considered the easiest method of adding an additional layer of security to SSH logins. However, for many, Hardware Keys are considered the single most secure means of preventing hackers from brute-forcing your SSH passwords. To make things easy, the OpenSSH developers have made it possible to generate a FIDO token-backed key using the ssh-keygen command. So anyone used to creating SSH keys shouldn't have any problem getting up to speed with integrating hardware keys into SSH.

To gain this feature, make sure you've upgraded to the latest OpenSSH release (8.2 or newer).

Original news release: *http://www.openssh.com/txt/release-8.2*

System76 Launches New AMD Threadripper Machine

The most successful retailer of Linux-based desktops, laptops, and servers has announced a new addition to their popular Thelio desktop lineup (*https://system76.com/desktops*). The new option, part of the Thelio Major model, adds AMD's 64 Core Threadripper 3990X CPU into the mix. This system can compile the Linux kernel in 24 seconds, apply a circular motion blur in 44 seconds, and render a Blender scene in 76 seconds. That's incredibly fast.

The Threadripper Thelio Major has been optimized for the heat produced by the 280 watt, 64-Core CPU, which was a serious undertaking. System76 accomplished the task by using a 5.5" duct that pulls air from inside the system, directs it across a heat sink, and then (drawing the heated air through copper piping) sends it out of the machine through the rear. This method compartmentalized the GPU and CPU heat sources as well as the air that is used to cool the individual chips.

The Thelio Major ships with Pop!_OS and can be customized to best fit your needs (GPU, RAM, storage). The Threadripper version of the Thelio Major starts at $3,798 USD, but can be maxed out to a whopping $14,131 USD.

If the Threadripper version of the Thelio Major is out of your price range, you can always opt for the basic Thelio model, which starts at $899 USD.

Original announcement: *https://system76.com/threadripper*

Get the latest news in your inbox every two weeks

Subscribe FREE to Linux Update

bit.ly/Linux-Update

Southern California's open source and free software expo

SCaLE 18x

With numerous exhibitors and workshops, SCaLE 18x covered a wide range of open source topics. *By Richard Ibbotson*

SCaLE 18x, in Pasadena, California, from March 5-8, 2020, offered over 150 exhibitors and nearly 130 workshops, tutorials, and events.

The event kicked off on Thursday morning with workshops, including tracks on embedded, PostgreSQL, and sponsored topics. Probably the best workshop on Thursday was the introduction to containers and Kubernetes. After being reorganized into a full day event, demand for this workshop was greater than the presenters could have anticipated. In the sponsored track a workshop on working with Jenkins and continuous delivery was offered.

On Friday, topics like Kubernetes, GitLab, PostgreSQL, and related

issues were presented throughout the day. At 2pm, the exhibition hall opened to a large crowd who grabbed any GNU/Linux swag that wasn't nailed down. Aleksey Tsalolikhin, from the Linux Professional Institute, gave a presentation entitled "DevOps Tools Engineer Preparation Session." Other events on the schedule included a GitLab community day, several DevOps sessions, "Getting Started with FreeBSD," and many others.

On Saturday morning, Paul Vixie's keynote speech, "DNS Wars, Episode IV: A New Bypass," covered his work in the DNS field since 1989, including inventing many of the monitoring and filtering capabilities now used by nearly all DNS services. He discussed the web-based DNS over HTTP (DoH) protocol, which is being pushed by Mozilla and others.

If you are familiar with Bradley Kuhn [1], you know how good his public presentations are. Kuhn gave a talk on "What'll We Do When FOSS Licenses Jump the Shark? The Next Season of Copyleft License Drafting and Promulgation." He discussed the current situation where proprietary commercial organizations publish open source software licenses that are not open source, as well as Copyleft-next, an experimental effort to create a new and easier-to-understand copyleft license.

On Sunday, Sha Wallace-Stepter and Jessica McKellar gave a keynote talk entitled "From Prison to Python: What Is the Free Software and Broader Tech Community's Role in Criminal Justice Reform?" The keynote covered how Wallace-Stepter learned to program in prison while serving a life sentence for assault with a firearm.

As a smaller venue, SCaLE offers advantages over a large-scale GNU/Linux event. At SCaLE, it is much easier to attend workshops or presentations, as well as to connect with other people in your area of interest. If you missed SCaLE 18x, you can view the long list of presentations on YouTube [2]. I hope to see you next year at SCaLE. ■■■

Info

[1] Bradley Kuhn: *https://www.socallinuxexpo.org/scale/18x/presentations/whatll-we-do-when-foss-licenses-jump-shark*

[2] SCaLE 18x on YouTube: *https://www.youtube.com/user/socallinuxexpo/videos*

Meet the FreeBSD Foundation

Going Places

As the FreeBSD Foundation approaches its 20th anniversary, we talk to its top brass about the project's growing visibility in the open source ecosystem. *By Mayank Sharma*

There's more to open source than Linux. FreeBSD, a free and open source Unix-based operating system, also plays an important role in the open source ecosystem. We talked to the FreeBSD Foundations's [1] executive director Deb Goodkin and director of project development Ed Maste at the recent Open Source Summit in Lyon.

Linux Magazine: *Tell us about FreeBSD and the FreeBSD Foundation's long history.*

Deb Goodkin: The foundation was founded in March 2000, so we're actually going to be celebrating our 20th anniversary March 2020. The original goal was to be sort of a legal entity that could hold any type of FreeBSD IP, and, at that time, it was the trademarks that we were given, so we've had ownership of trademarks and other IP since then.

LM: *What is the FreeBSD Foundation's role with respect to the FreeBSD Project?*

DG: We're a separate entity. Some open source projects, including NetBSD [2] and OpenBSD [3], are actually underneath their respective foundations, but we're a separate entity. We step in to support the project with critical needs. For example, they needed a release engineering person a few years back, and we were able to bring someone on full time to fulfill that role. It's not always a long term need; we tend to help when they need assistance and they don't have a volunteer who could step in to do it.

We also provide continuous support for software development. Even though we don't direct the project on what to do, we do give them advice on what we hear from end users and commercial users. And then if the need relates to a larger technology or feature, we do have software developers on staff, and that's what Ed oversees. But we also support outside contractors – for instance, an outside project might need a developer, and we find someone.

LM: *So how many people do you have on staff?*

DG: At the Foundation, including the full-time contractors, the number fluctuates, but right now we probably have around eight; and we have eight board members. The staff and board are located all over the world.

LM: *What kind of projects do you get involved with?*

Ed Maste: I guess broadly speaking, the foundation takes on projects in areas that the community (either corporate developers or the general development community) isn't addressing themselves. We step in to do work that has a broad range of interest. Development tools fall into this category. Everyone wants to have a good development tool chain, but it's not to anyone's competitive advantage to develop the tool chain independently.

DG: Yeah, so there might be something that no volunteer is really interested in doing, or it could be just stepping in to fix minor bugs. Our goal is to help get things fixed quickly and keep FreeBSD stable.

LM: *Where do you get the budget to fund this development?*

DG: We receive all of our funding from donations – anywhere from a $10 individual donor to a large corporation.

LM: *Do you have a relationship with iXsystems? Aren't they one of the largest monetary contributors to the FreeBSD project?*

DG: Um, well, not as a donor, but as far as having resources on staff that contribute to the project, I would definitely agree with that. We have a great relationship with iXsystems. We view them as a great FreeBSD advocate – a company that upstreams lots of changes and

helps promote FreeBSD, but just like any other commercial user.

EM: Yeah, they've made donations in the past, sometimes just kind of general donations, and sometimes we've collaborated on a couple of projects where we found some interest in some areas of technology, and then iXsystems contributed some money, because they had an interest in the same kind of project.

LM: *Are there any new projects that you'd like to highlight?*

EM: One we just wrapped up was a big effort to improve FUSE filesystem support [4]. We started off with a GSOC [Google Summer of Code] project many years ago that did the initial port of FUSE to FreeBSD. And then another project updated a little bit and integrated it. But it was never really at the level of being fully production quality and usable. And so the foundation funded a project with a developer in the FreeBSD community recently to bring the FUSE implementation up to the contemporary version that matches what's in Linux. We fixed a lot of bugs, we have a comprehensive test suite now, and all of the outstanding known issues have been resolved. And now, you should be able to choose an arbitrary third-party FUSE filesystem, like NTFS or SSHFS or any sort of FUSE module, and expect it's going to work properly on FreeBSD.

And I think that kind of fits into a goal we've had: to improve the experience of FreeBSD on the desktop or on a laptop for end users. We've been looking at some projects to improve the WiFi situation on FreeBSD, for example.

LM: *So you're focusing on the desktop now?*

DG: Yeah, and one reason why we're doing that is we're looking at recruiting new users and contributors. We're looking at college students. And so we're making sure that, while we're getting into more universities, it's easier for someone to download FreeBSD and run it on their system.

EM: Yes, FreeBSD is perfectly usable for corporate, large-scale server users (people like Netflix). If you're a developer working at somewhere like Netflix, you are already invested in FreeBSD. It's quite easy to just use a virtual machine running on Windows or on Linux or on macOS. And that's a perfectly usable development environment. But for a lot of cases (students in universities, the kind of next generation of FreeBSD users we want to capture), it's not very compelling to say "Yes, you can do your FreeBSD work if you go through all these extra hoops."

LM: *Isn't TrueOS [5] the desktop version of FreeBSD?*

EM: TrueOS has long been a FreeBSD desktop, but there are a few others as well – like GhostBSD [6] and MidnightBSD [7]. I certainly am happy to have those projects package up the operating

Shop the Shop
shop.linuxnewmedia.com

Missed an issue?

You're in luck.

Most back issues are still available. Order now before they're gone!

shop.linuxnewmedia.com

system and produce distributions with preconfigured options and have it usable in a way that is convenient for a new end user.

The FreeBSD base project is more focused on fundamentals, such as improved WiFi support and improved Intel graphics drivers, for example. If other projects come along and repackage it, the real value is in being able to say, "here's a set of packages that we've said work well together, and we've chosen defaults, and you can run the setup through our installer and get a usable system as we envisioned it."

LM: *Will FreeBSD produce its own desktop as well?*

EM: FreeBSD essentially provides a toolkit. We want to support the foundational infrastructure for desktop applications. So the foundation isn't going to produce a FreeBSD-Foundation-branded desktop. We just want to make sure that all of the foundational parts that are needed for a good desktop are in place.

DG: That's sort of the philosophy for all the development. Regardless of whether you're using FreeBSD as a server or for something else, we want the system to be at a state where it's easy to customize for how you want to use it.

LM: *You mentioned Netflix. Can you name some other high-profile users of FreeBSD?*

DG: Yeah, so NetApp, Juniper, we always say Apple, Trivago, WhatsApp, and Groupon.

EM: Companies like Trivago – they use FreeBSD as a server. They're not building their own appliance or anything on top of it. They just use FreeBSD as it exists. And they have a combination of Linux and FreeBSD; that's a good model for using FreeBSD – where its strength lies.

DG: Verisign does that as well. The *LA Times*, which is humongous, uses FreeBSD, and universities like Notre Dame use FreeBSD in their school of engineering. We know about these users because we go to a lot of open source conferences around the world, and people come to our table and they'll tell us how they are using FreeBSD.

One thing that we're really trying to do is publicize our users. We're just starting to work on some case studies. Netflix will be one of the first ones. They've been going around the world giving a Netflix/FreeBSD talk. Actually, the video of the talk from FOSDEM is online, and it's really interesting to watch [8]. We really want to highlight that talk and write up a case study and get more testimonials, so that more companies are aware of what they can do with FreeBSD.

LM: *Is Netflix just a user?*

EM: They are very good citizens in the FreeBSD community. They contribute an awful lot of code into the FreeBSD kernel, and they do a lot of performance work and network stack improvements. And because of their architecture, they're able to consume new versions of FreeBSD very quickly. So they're actually tracking our development branch in their releases. They can do that, because their whole architecture is sort of built on multiple layers of redundancy. They do a lot of large-scale tests, and we get a lot of large-scale test coverage because of Netflix.

DG: And they're really open to sharing their model. Other companies are aware of the Netflix talk now, and they've actually asked for help.

LM: *What happens when other companies ask for help?*

DG: One thing that we do is facilitate collaboration between companies. We may assist with setting up meetings, but most of the time is spent connecting people with each other.

LM: *Do you think the FreeBSD Foundation being at Open Source Summit demonstrates the open source nature of the conference?*

DG: Even though they call it Open Source Summit, it's still very Linux oriented. I say we're a minority voice in the open source world, and we have such a long history that there's a lot of components to the project that are really good models for other open source projects. And it's just really important for us to work with the Linux community.

EM: The kernel code is sort of completely distinct, although there are some cases where we're actually able to share things. But certainly the application stack running on top of FreeBSD and Linux – we collaborate with lots of upstream projects that primarily do their development on Linux and their main user base is on Linux, but they're very happy to work with us to bring up CI testing for FreeBSD or integrate patches to make sure that their software continues to work on FreeBSD. And so I think a conference like this is actually really beneficial for us to come and meet people who are outside of the usual circle of the BSD community.

LM: *Did you just celebrate the FreeBSD operating system's 26th anniversary?*

DG: Yeah, we have a long history. We descended from the original Unix from Berkeley and branched off from there. So it's fun to look at the history.

LM: *To see how far you've come?*

DG: Yeah. But also there's so much innovation that has happened and that is still happening. We still have some of the original Berkeley people on the project. And some of our oldest members are saying "let's make sure we're staying modern, that we're getting the young people involved," which is always great to hear. But there is also that philosophy too of change when there's a reason to change. If there is a compelling reason, let's change, but let's not change just for the sake of changing. ■■■

Info

[1] FreeBSD Foundation: *www.freebsdfoundation.org*
[2] NetBSD: *www.netbsd.org*
[3] OpenBSD: *www.openbsd.org*
[4] FreeBSD's FUSEFS implementation: *https://wiki.freebsd.org/FUSEFS*
[5] TrueOS: *www.trueos.org*
[6] GhostBSD: *https://ghostbsd.org*
[7] MidnightBSD: *www.midnightbsd.org*
[8] Netflix Talk on FreeBSD: *https://archive.fosdem.org/2019/schedule/event/netflix_freebsd*

SHAPING TOMORROW

ISC HIGH PERFORMANCE 2020

LEARN, EXPLORE, CONNECT

TUTORIALS | CONFERENCE | EXHIBITION | WORKSHOPS

HPC | NETWORKING | STORAGE | DATA ANALYTICS | AI

REGISTER BY MAY 5 FOR EARLY BIRD RATES

JUNE 21 – 25
2020
FRANKFURT, GERMANY

ISC High Performance
The HPC Event.

Platinum Sponsors:

Atos
Hewlett Packard Enterprise
HUAWEI
intel

2020 PROGRAM TRACKS

System Architecture | Applications & Algorithms | Emerging Technologies | Parallel Programming Models & Performance Modelling | Machine Learning | Industrial HPC

isc-hpc.com

What is the Edge and why are we all talking about it?

Edge Computing Today

After the cloud came the Edge. We take a look at the Edge computing phenomenon and attempt to assess what all the fuss is about. *By Martin Loschwitz and Joe Casad*

Edge Computing is a popular term in the high tech media, and, like many buzzwords that rise to claim a place in the limelight, the term "Edge" appears to have emerged fully formed before the industry settled on a clear definition for what it is. So what is Edge computing, and how does it differ from other approaches? How is Edge computing related to IoT or other contemporary technologies? We decided it was time for a visit to the Edge.

Beyond the Cloud

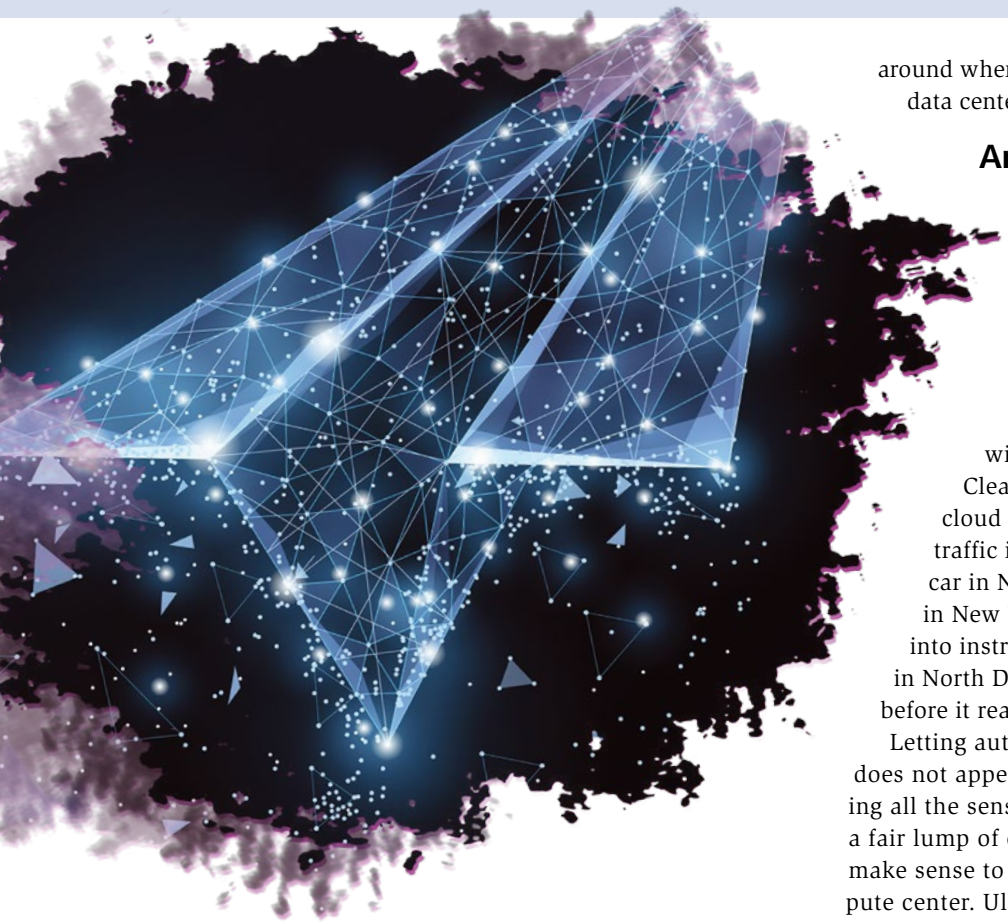
For years now, large cloud providers have attempted to entice customers with the benefits of managing their data and infrastructure from a central, cloud-based location. For many companies, *the cloud* means abandoning their own on-premises data center and instead trusting their data to AWS, Azure, or another cloud company. The goal of the cloud is *centralization*. The data is all in one place, managed by experts with economies of scale. Security, fault tolerance, and other essential tasks are handled from the central facility. In many cases, all the data and processing power for an entire company might be in one or two locations, with branch offices accessing it from all over the world.

Edge computing is the exact opposite of this centralized cloud paradigm. The goal of Edge computing is to provide similar cloud-like services, but to move computing resources to the farthest edge of the environment – geographically close to where the data is gathered and accessed.

From CDNs to the Edge

The idea of placing computing resources at the edge of the network is nothing new. Around 20 years ago, the Web 2.0 era ushered in a new vision for the Internet. The volume of data skyrocketed quickly as new services for social media, images, and video entered into common usage. The industry soon found that it was unable to develop new higher-capacity hardware at the speed the market was asking for it.

As early as the end of the 1990s, the idea of Content Delivery Networks (CDNs) was born. CDNs followed a very simple principle: Instead of keeping the movie on a server in the USA, a company installs infrastructure closer to the customer and keeps a copy of the video closer to where the customer lives. Keeping the connection confined to a small region reduced latency, and it also simplified the



communication path, requiring fewer routers and less overall traffic to deliver the data to the user.

The original CDN systems were primarily designed to offer storage, but today's Edge environment requires a much more elaborate palette of services. New technologies such as robotics, Internet of Things, remote sensing, and realtime monitoring handle lots of data, but they also require lots of computing power.

Home automation is a good example. Classic household appliances are replaced with state-of-art versions that provide Internet access and can be controlled remotely. Decisions occur beyond the device, and the results are transferred back to the device through simple commands. Voice-activated tools like Alexa offer additional complications. When a user talks to Alexa, it fields the command and sends the audio file to a server, where it is analyzed and interpreted. Alexa then receives the command back in machine language so that it can initiate an action. The longer the distance between Alexa and the server that interprets the command, the more sluggishly Alexa behaves.

A home assistant turning up the thermostat can probably afford a little latency, but consider a robotics installation on a factory floor or a set of sensors that monitor environmental data and make complex decisions in real time. These scenarios would potentially benefit from some form of cloud-like consolidation of processing power, but the idea of sending every command and sensor reading to a massive server in another region of the country hundreds of miles away is severely limiting and, in some cases, totally unworkable. The Edge offers a framework for imagining how the same technology would work with lots of mini-data centers scattered around wherever they are needed, instead of a massive data center serving a radius of a thousand miles.

An Edge Example: Autonomous Driving

All the major car manufacturers have been researching autonomous driving for years, and various Silicon Valley companies have also explored the possibilities of cars without drivers. Autonomous driving is a good example of why the experts believe Edge computing will figure so prominently in the future of IT. Clearly, it would not be practical for a central cloud infrastructure to process telemetry data and traffic information for a large region. If data from a car in North Dakota is first sent to the data center in New York, where it is evaluated and converted into instructions, which are then sent back to the car in North Dakota, the information would be out of date before it reaches the vehicle.

Letting autonomous vehicles crunch their own data does not appear to be a meaningful alternative. Evaluating all the sensor data of an autonomous vehicle requires a fair lump of compute power, and it simply does not make sense to turn every car into a small roaming compute center. Ultimately, energy considerations also play a role in ruling out on-vehicle data processing, because the car of the future will be electric.

Consequently, it will be necessary to provide cloud-like, off-vehicle data processing that is close enough to the vehicle to minimize latency – an ideal scenario for Edge computing. This solution could entail dividing a country into regions and then rolling out local islands for compute and storage. Ultimately, every car manufacturer will have to do their own thing, but similarities will undoubtedly exist between the designs.

The challenge will be to provide these small ad hoc mini-data centers wherever they are needed. Big cities would probably have to be divided into several areas – but, especially in city centers, comprehensive IT infrastructure is rarely available. Much of this infrastructure will need to be built from scratch or developed through complex sharing arrangements with existing companies. In the long run, the Edge will require lots of well-ventilated rooms scattered around the world, each with a server cabinet and a capable (possibly redundant) power supply.

Building Blocks

The large number of instances associated with an Edge environment cries out for standardization. At the same time, flexibility is also important, because each location could be slightly different.

Against this backdrop, containerized data centers have emerged as an important solution for Edge applications. In the Edge context, containerized data centers offer the simplest way to roll out a product that is standardized from the first nut down to the last bolt almost anywhere in the world with just a little preparation.

Figure 1: Containerized data centers are perfect for Edge: they can be delivered quickly, are versatile … © Rittal

Figure 2: … and can be set up virtually anywhere. You only need power and network connections. © Rittal

When it comes to containers in this new context, two old acquaintances come to the aid of the vendors. On one hand, OpenStack [1] offers most of the functionality that a decentralized solution requires. A number of additional features, which the developers added to the software in recent years, come in handy, such as support for the Ceph virtual storage solution. Several telco companies are currently building OpenStack into their commercial Edge services.

Another option is to build a distributed setup with local Kubernetes clusters. If the application is really cloud-native and available in containers, Kubernetes appears to be the tool of choice, because it is still far less complex than OpenStack. However, if IaaS is on the list of requirements, Kubernetes will probably not be the ideal solution.

Industrial container systems are available in many sizes. Several hardware vendors provide data center products designed to support containerized environments. The customer receives a mini data center that already solves essential issues such as ventilation and the required racks (Figures 1 and 2). Once the unit is delivered to the site and configured, the only things missing are the grid and the network. Electricity is relatively easy to access in most places. As for the network, the promise of 5G wireless is the final puzzle piece that could lead to widespread migration to the Edge.

Edge and 5G

All the efforts to reduce latency by relocating hardware are of little benefit if networks are not fast enough to meet the challenge. Experts believe 5G wireless is the emerging technology that will meet the simultaneous demands of speed and mobility that will fuel the Edge revolution. With its palette of engineering tricks, 5G radically increases the bit rate for individual clients. 5G will offer the bandwidth to manage the flood of information from hundreds of thousands of autonomous vehicles.

5G also offers beamforming with multi-MIMO antennas: One 5G antenna can support several individual beams assigned to specific customers (Figure 3).

Where Are We?

Most of the talk about Edge technology is still very theoretical, but the major cloud vendors are taking the Edge very seriously. As you could probably guess, much of the activity swirling around Edge computing is about cloud vendors with data center products maneuvering to form partnerships with wireless vendors who will provide the 5G wireless capabilities. Recent reports indicate that Google is in discussions with AT&T to collaborate on developing Edge services [2].

Verizon has already rolled out an ambitious Edge service in partnership with Amazon AWS. The Verizon Multi-Access Edge Computer (MEC) platform [3] is built on AWS Wavelength Edge computing technology. Verizon and other wireless providers have the advantage of being able to locate Edge services at the thousands of Service Access Point (SAP) and C-RAN locations that they currently operate to support their wireless telephone networks.

Rather than teaming with a telco, Microsoft Azure appears focused on keeping the Edge on premises. Azure Stack Edge [4] is an on-premises appliance that brings Edge services to the customer's home network, which would work well for a robotics solution but not for a disparate application like autonomous cars.

Given the excitement currently devoted to Edge computing, it is not surprising that almost every major telco and cloud vendor has some kind of product, service, or initiative with the term "Edge" in the title. All these services share the goal of decentralization, but the details vary widely.

Conclusions

Edge Computing is rightly considered a future driving force in IT. As more and more people use digital services in their everyday lives, it makes sense to operate them closer to the point of use. However, the industry still faces some technical hurdles. A large part of the task facing solution providers will be to integrate platforms that follow the Edge paradigm into existing infrastructures in a meaningful way. ■■■

Info

[1] OpenStack for Edge Clouds: *https://www.openstack.org/edge-computing/cloud-edge-computing-beyond-the-data-center/*

[2] Why Google Cloud and AT&T May Merge Their Telco Edges: *https://www.datacenterknowledge.com/edge-computing/why-google-cloud-and-att-may-merge-their-telco-edges*

[3] The Power of Verizon 5G Edge: *https://enterprise.verizon.com/business/learn/edge-computing/?cmp=paid_search:google:brandcredmec:sem:awareness&gclid=EAIaIQobChMIrKOZn_Wp6AIVCRgMCh-0CmgE-EAAYASAAEgKLqfD_BwE&gclsrc=aw.ds*

[4] Azure Stack Edge: *https://azure.microsoft.com/en-us/products/azure-stack/edge/*

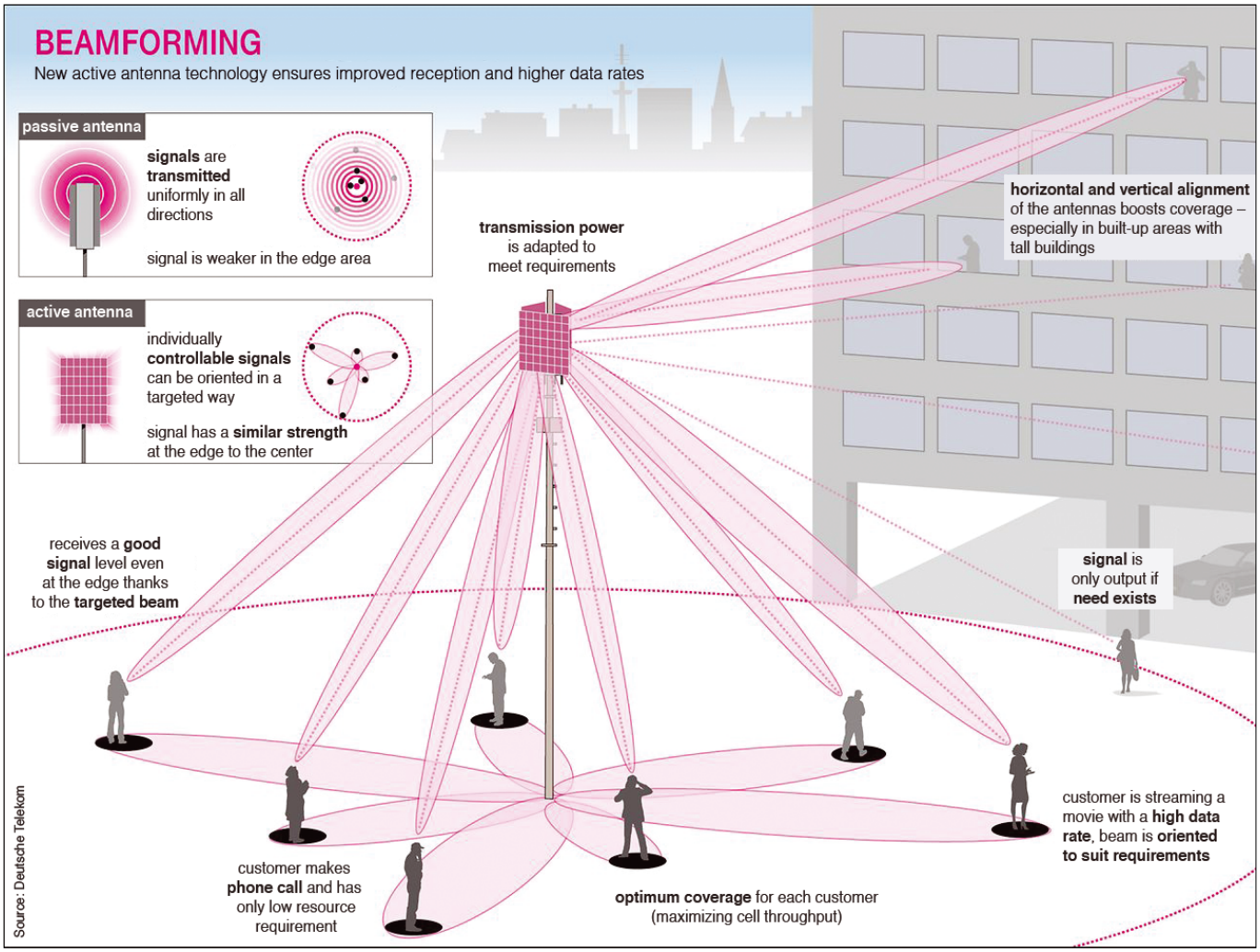


Figure 3: Beamforming in 5G offers the ability to connect Edge locations wirelessly just as easily as with copper cables. © Deutsche Telekom

Hone your skills with special editions!

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

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

Available in print or digital format

Check out the full library!
shop.linuxnewmedia.com


FREE DVD!
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
LINUX SHELL
HANDBOOK
2019 Edition
LINUX Special
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
Keep this comprehe
guide as a permaner
command reference!
WWW.LINUX-MAGAZINE.COM
FREE DVD!
LibreOffice Full Version
Includes full versions for
Windows, macOS, and Linux
2019 Edition
DISCOVER
LibreOffice
Free Office Suite
Find out why 75 million users have stopped paying for office software!
Edit and Save MS Office Files!
Create your own:
Word processing docs
Spreadsheets
Presentations
Databases
Use open file formats that other programs can read and import!
Also inside:
Extensions
Macros
Templates
and mor
WWW.LINUX-MAGAZINE
LINUX Special
101 COOL LINUX HACKS
2020 EDITION
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 Special
WWW.LINUX-MAGAZINE.COM

Matt Trifiro on the state of Edge computing

The View from the Edge

We talk with Matt Trifiro, one of the publishers of the State of the Edge report, on the nature of Edge computing and how it is changing IT. *By Kristian Kißling*

LM: *"Edge computing" is a term with many definitions. What does Edge mean to you and why is your definition more relevant than others?*

Matt Trifiro: For the first State of the Edge in 2018, we identified a need to present a framework in which the industry could have productive discussions around Edge. One of the big challenges at the time (and, to some extent even today), is the proliferation of definitions for "Edge." If you look closely, most Edge definitions are driven by a vendor or pundit's self interest, using language as an attempt to differentiate their products or points-of-view rather than as a way to bring the industry together. For example, what is the "telco Edge"? Or, what is the "IoT Edge"?

We approached the problem like lexicographers and surveyed the market. We realized, of course, that there are lots of Edges, but we also realized we had an opportunity to create clarity and offer a framework that could be independent of any particular vendor's business. Our framework has three principles:

- The Edge is a location, not a thing.
- There are lots of edges, but the Edge we care about is the edge of the last mile network.
- This Edge has two sides: an infrastructure edge and a device edge.

In 2018, we offered these principles as a gift to the industry and the industry responded by welcoming them into the vernacular. In fact, this description of *Edge* has been canonicalized in the Open Glossary of Edge Computing, which is an open source project at The Linux Foundation. Anybody can make comments, suggestions, and improvements, which gives us a corpus of knowledge that can evolve for the whole industry.

LM: *According to estimates, the industry is expected to spend US$700 billion dollars in the next 10 years in the Edge sector. Who is spending the money and for what?*

MT: This is precisely why we have been so meticulous in creating a standardized definition of Edge, so that when we do something as important as discuss a market projection, we do it using a carefully scoped model. The number US$700 billion is what we came up with in our model, given the relatively narrow constraints we put on our model, which is focused only on infrastructure spending – and does include things like economic impact. I've seen Edge projections that trumpet over a trillion US dollars, but they are including everything in that model, such as the jobs and businesses that emerge from Edge technology.

I'm not saying these other models are wrong. My point is only that we need to be clear about what we are and are not including in the projection.

Our US$700 billion projection estimates the total dollars that will be spent between today and 2028 on a very specific component of Edge computing: the infrastructure

required to support Edge use cases. Not devices. Not economic impact. The report forecasts Edge data center and IT infrastructure investments in terms of capital expenditures (measured in US dollars) and the power footprint of the IT infrastructure deployed (measured in Megawatts). The predicted capital expenditures offer insight into the market opportunities for Edge IT equipment and infrastructure providers. It does not, for example, predict the opportunities for device manufacturers or auto makers, or even the economic impact of the Edge. Those are for future reports.

LM: *An example in the white paper "State of the Edge 2020" cites the control of drones as an Edge use case. Isn't that also an example of how far away Edge is at the moment? Even the fastest mobile networks would not be able to control drones today – for reasons of latency, security, and safety. So how far away, in years, is an Edge scenario?*

MT: Autonomous drones are a lot closer than you think. Vapor IO, my company, has a lab in our Chicago deployment that can demonstrate autonomous drones running over a wireless network. The wireless 5G technology exists today; it's just in the early stages of rollout. With drones, you can align the wireless upgrade along specific transport corridors, which is already happening in some areas. Given that the FAA continues to move quickly to support drones without human pilots, we'll see these capabilities being built out this year in certain cities to great economic benefit. Autonomous drones will create billions of dollars of economic value.

Also, Edge is not just a wireless technology. Many of the existing wireline networks, including cable and fiber, already have the latency and security problems solved for the access network (the link between the infrastructure and the devices), so it's just a matter of deploying the local IT infrastructure to support it, at the infrastructure edge, which is starting to happen at scale in 2020.

LM: *We view nuclear power plants with skepticism, and rightly so: Despite high safety standards, several accidents have occurred in the past. Hospitals have to switch to paper and pencil if a Trojan encrypts their infrastructure. If Edge now connects all the daily devices and objects we use and the Internet (partially) fails or is attacked: Won't we end up back in the stone age, with more severe consequences than we face now?*

MT: There are risks inherent in any new technology or infrastructure, and we have to be cautiously optimistic about how we tackle these problems and ensure we minimize or eliminate any single point of failure, but it's clear to me that Edge infrastructure will meaningfully increase the resilience of the Internet and connected devices specifically because it is decentralized, which means you can distribute risk across many different sites and have no single point of failure. For example, with seven Edge micro data center facilities in a metropolitan region, you can deliver something like twelve 9s of reliability using highly-available software failover. Barring some catastrophic event that takes out an entire city, this will greatly increase the resilience of our networks and our applications that depend on them.

The use of infrastructure Edge computing, specifically, can do much to enhance the security and resilience of our entire connected economy. Each infrastructure edge site can host software and hardware solutions dedicated to providing next generation security services at the very edge of the network. Threats that are detected in an edge location can be quarantined in that location, preventing them from penetrating deeper into the infrastructure or to the Internet.

LM: *In the end, doesn't Edge, despite its decentralization tendencies, lead to an overall further centralization of the Internet?*

MT: I don't agree with this conclusion. Edge, by definition, is decentralized and it is massively so, especially when compared to the Internet we have now. Today, for example, in the United States, if you want to provision a workload on Amazon Web Services, you can spawn an EC2 instance in basically two locations: US West and US East. That is extremely centralized. Now imagine a future Edge, where there are thousands (not dozens) of AWS data centers all over the world. In this future, you should be able to provision a workload not just in US West and US East, but also in Chicago West and Shanghai East.

The data paths of the Internet will also continue to decentralize. Today, Internet traffic typically passes through a small handful of large interconnection points that have emerged in hub cities, such as Atlanta and Dallas. Today's Internet passes through a relatively small number of these hubs. For example, in the US, there are maybe a dozen meaningful interconnection and exchange points. As Edge infrastructure gets deployed, a network of smaller edge exchanges will emerge to exchange data between local networks without needing to take a longer route that passes through a hub city. This creates a decentralized Internet beyond what we have seen.

LM: *In an Edge computing scenario, the necessary calculations take place directly on the end devices, which have only limited capacity. At the same time, technologies like AI are very resource-intensive. Aren't these two trends opposing and contradictory?*

MT: No. In a world where our devices are ubiquitously connected to low-latency networks, we can make decisions and tradeoffs about where different services and workloads will run, and this may vary with the application or even the time of day. For example, you can build an expensive sensor with a lot of AI capabilities on board. Or, you could offload that work to a nearby Edge server and deploy lower cost devices. There will be a continuum that stretches from the centralized core all the way out to the device, and developers will have the choice of where to run their workloads along the entire spectrum.

LM: *Edge should work as smoothly as the power grid. But even that fails from time to time – and quite frequently in some areas. How can a much more complex Edge technology, which already requires constant software updates, guarantee a safe and secure infrastructure in the future?*

MT: The secret to resilience in an Edge world is distributed infrastructure and highly available software. A properly deployed Kubernetes service, for example, can be placed in an edge data center, and if that edge data center goes down or there is a fiber cut, the orchestration engine can restart that service in another nearby location. Also, modern software techniques, such as continuous integration, automated testing, blue/green deploys, and rollbacks can all help ensure a highly-reliable system. Moreover, because most edge data centers are remotely operated, they often have sophisticated arrays of sensors that can be remotely analyzed and monitored to perform fine-grained optimizations around energy usage.

LM: *A problem of today's 3G and 4G networks may remain even with Edge: There is little incentive to extend the associated infrastructure to rural areas. Moreover, parts of the necessary infrastructure are hardly profitable. What incentives can there be for looking after middle-mile architecture and edge data centers in rural areas?*

MT: This is a legitimate concern, and I suspect we may want to consider solutions that have worked in the past. In the US, we have this concept of universal service. In 1935, the US established the highly successful Rural Electrification Administration and then, in 1944, extended that model to the Rural Telephone Administration. These entities stimulated rural infrastructure upgrades with programs such as offering long-term, low interest loans.

Perhaps future programs such as the Connect America Fund (CAF) will seek to include Edge.

LM: *Another problem is the different speeds and budgets in nation states, countries, and regions, even cities. Certain mobile Edge applications require a consistently functioning infrastructure, such as cars, ships, and airplanes. Will the autonomous vehicle stop at the national border? Will there be a uniform Edge infrastructure, or will everyone do their own thing, as the electric car manufacturers are doing today?*

MT: My crystal ball doesn't go to that level of granularity, but I will say we've had these sorts of problems in the past, and we've mostly managed to solve them. It's absolutely astounding to me that I can carry a cell phone around the world and have it just work on any foreign network – or that I can dial a seven-digit number with a three-digit country code and reach someone on the other side of the planet. We've seen successful models of compatibility and integration emerge in the past, and we should try to model those.

LM: *If Edge comes in the form it's designed to take: Are there actual projects and practical considerations on how to make Edge's enormous energy footprint environmentally friendly?*

MT: Environmental friendliness is one of the largest trends in data center design today. Modern data centers can often operate at ambient temperatures ("free cooling") in many geographies and for an ever-increasing number of months throughout the year. New cooling technologies use self-contained water and coolant systems, so you don't need to ingest outside water. And so on.

Ultimately, as a world, we are going to need to take alternative power generation more seriously. Fossil fuels are a problem for every industry, not just Internet infrastructure.

LM: *We often talk about security, but rarely about safety. Everyday technologies on which human lives depend often have to pass complicated tests to be considered safe. What role does safety play in the development of Edge technologies?*

MT: Edge applications that place human life at risk will have to go through some of the same rigorous test phases and also be subject to regulation, just like non-edge applications. I don't think there is anything particularly unique about Edge when considering this challenge, but when viewed as part of the core-to-Edge continuum, a collection of Edge data centers can offer real-time collaborative monitoring and processing for Edge devices, and this stands to improve the overall safety of applications and devices that impact our physical world, such as vehicles. ■■■

What?!

I can get my issues SOONER?

Available anywhere, anytime!

Sign up for a digital subscription and enjoy the latest articles on trending topics, reviews, cool projects and more…

shop.linuxnewmedia.com/digisub

Now available on ZINIO: bit.ly/Linux-Pro-ZINIO

A lean distro for 32-bit processors

Shedding Weight

For older computers with 32-bit processors, BunsenLabs Helium offers a lean alternative to popular Linux distributions. Our lab investigates how well the system performs on antiquated hardware. *By Erik Bärwaldt*

With many Linux variants only being released as 64-bit versions, choosing a new distribution for 32-bit systems has become increasingly difficult. BunsenLabs Helium [1], a Debian derivative, offers a distribution with a lightweight, customizable Openbox desktop. Designed to use resources as efficiently as possible, BunsenLabs Helium even runs on older machines.

In addition to an image for 64-bit systems, BunsenLabs also provides two images of the current version of Helium for 32-bit systems: One for 32-bit systems with a PAE extension (enabling the use of more than 4GB of RAM) and one for systems without the extension. The image without PAE support weighs in at 680MB [2]. Regardless of your system, the developers recommend a minimum of 1GB RAM and 10GB free space on the hard disk.

All three hybrid images (64-bit, 32-bit with PAE support, and 32-bit without PAE support) are available on the project's website [1]. The images (each about 1.1GB in size) boot into an unimpressive GRUB boot manager, which supports Live operation as well as direct installation. A graphical installation tool is also available.

The Live version boots very quickly into an Openbox window manager. The developers have updated the appearance with modifications, making it practical for everyday use. In addition, a welcome screen (Figure 1) appears on startup with some simple instructions to get you started.

At the top of the desktop, tint2 is integrated as a taskbar. It accommodates two virtual desktops: a system tray on the right, and some application starters on the left. In the upper right corner, a Conky system monitor displays various PC components' parameters. Below that, the developers have included a list of important keyboard shortcuts for working with the desktop.

A start button and start menu are missing, as are any icons on the desktop. Instead, you can right-click anywhere on the desktop to open the main menu.

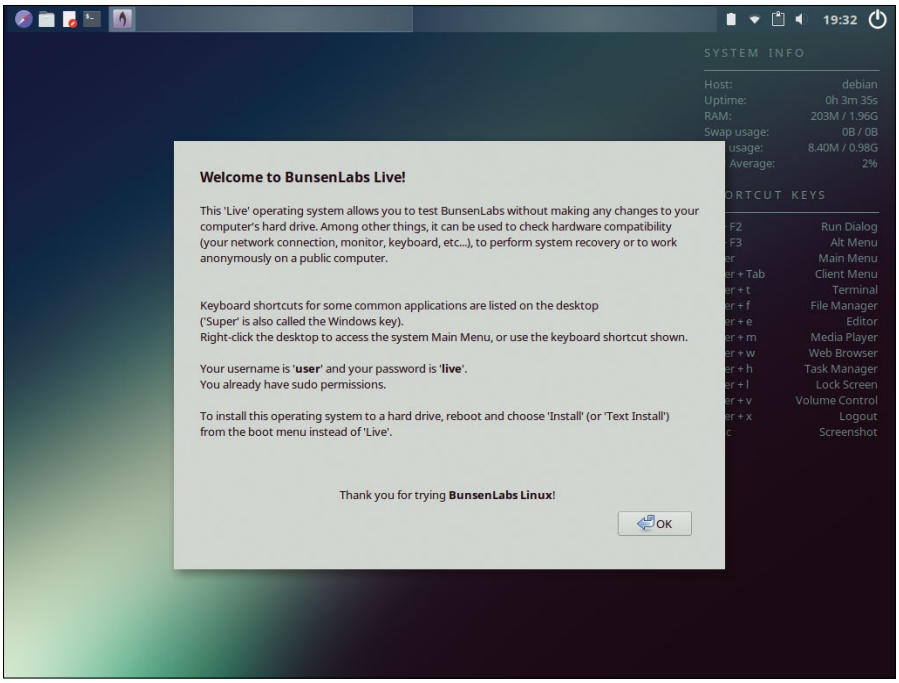


Figure 1: Although visually appealing, the BunsenLabs Helium startup screen takes a little getting used to in terms of its operating concept.

Lead Image © Kirsty Pargeter, Fotolia.com

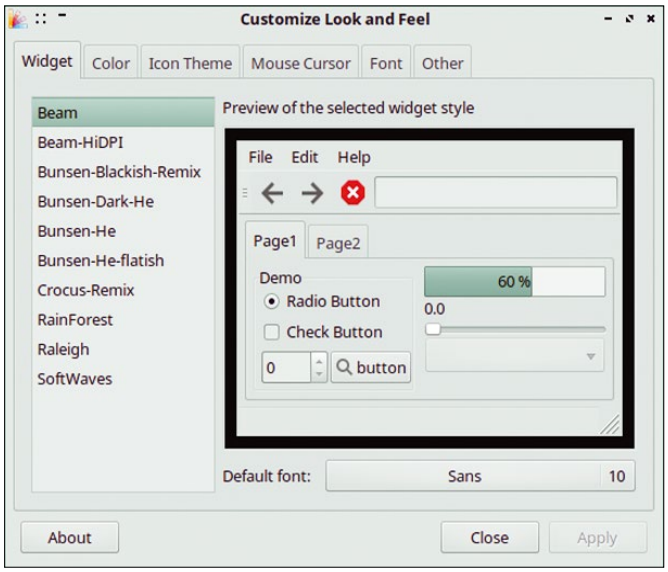


Figure 2: BunsenLabs Helium provides a GUI for customizing its appearance. For many other modifications, you will need to use a text editor.

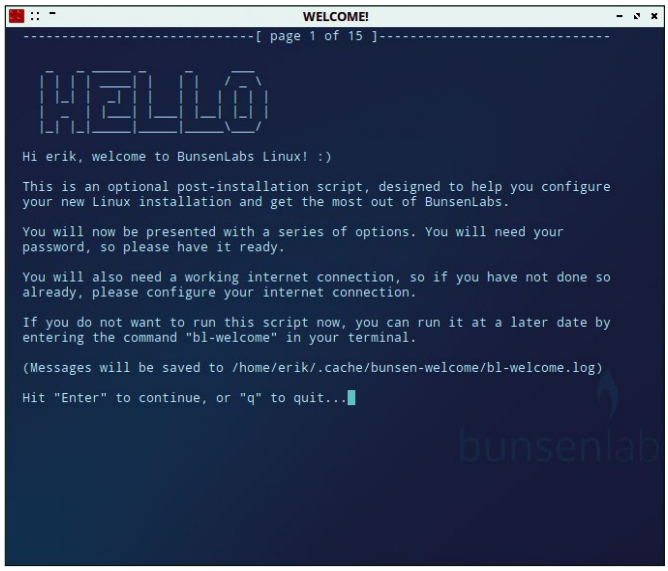


Figure 3: BunsenLabs Helium performs a basic configuration after initial startup, which is essentially based on a shell script.

Software

Even the Live version shows the system's frugality in terms of resources and how fast it still runs. In idle mode, BunsenLabs Helium uses only 200MB RAM.

In the menus, you will find some important standard applications preinstalled, including LibreOffice Writer and the Firefox browser (not exactly a lightweight option). Additionally, VLC, the universal media player, is already integrated into the Live system.

You can install further LibreOffice components directly off the Internet by selecting the corresponding menu entries, as well as the alternative browser, Chromium. Additional office applications come from the Gnome repository, including Gnumeric and the Evince PDF viewer. FileZilla lets you transfer data via FTP if required and supports transmission via various peer-to-peer services.

Somewhat out of the ordinary, you can establish a VNC connection to a remote computer to control other computers with the BunsenLabs Helium system.

Finally, since Debian offers one of the largest software collections in the Linux world, you can also use the preinstalled Synaptic front end to add more applications.

Configuration

In contrast to installing software, the configuration seems less intuitive. BunsenLabs Helium partially uses Openbox's sparse dialogs in addition to applications like Conky and tint2 to customize functions and menus.

Although there are very detailed options available for configuring Openbox, using them requires editing files in a text editor. The graphical settings menus (like the ones in Gnome, KDE, Mate, Xfce, or LXDE) are missing here in places. However, the Settings menu offers some graphical tools for customizing the desktop, as well as a more detailed dialog for modifying the look and feel (Figure 2).

Installation

In the Live system, there is no starter for the setup. To install on a hard disk, you need to reboot and select the option in the GRUB boot manager. The familiar Debian options are available. If you choose the conventional route, you can install the system on a PC in just a few steps using an ncurses wizard. If you choose the graphical route, the corresponding Debian wizard helps you.

BunsenLabs Helium already integrates numerous proprietary firmware blobs that are missing from it's basis Debian. This lets you reliably integrate hardware components that only cooperate with the operating system if these blobs are in place.

During testing in my lab, an annoying breakdown occurred. On my system, the graphical wizard failed to initialize the WLAN card, even though I agreed to the

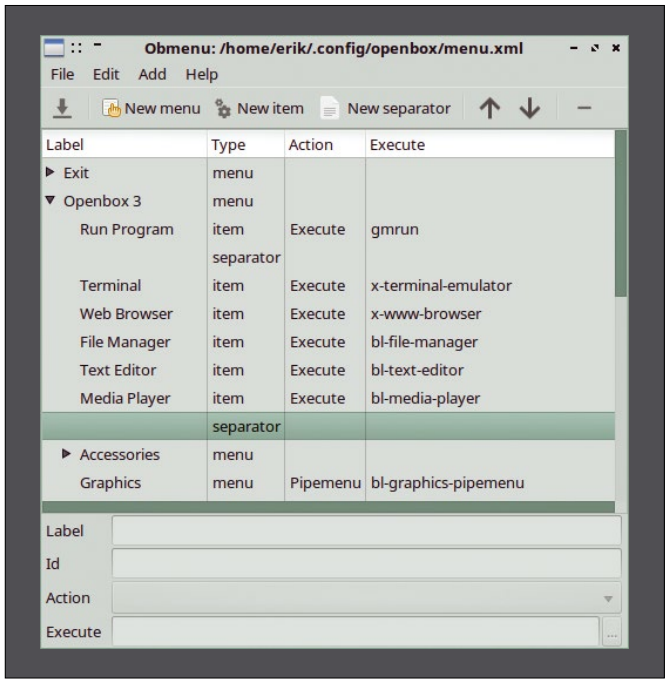


Figure 4: You edit the BunsenLabs Helium menu using a graphical tool.

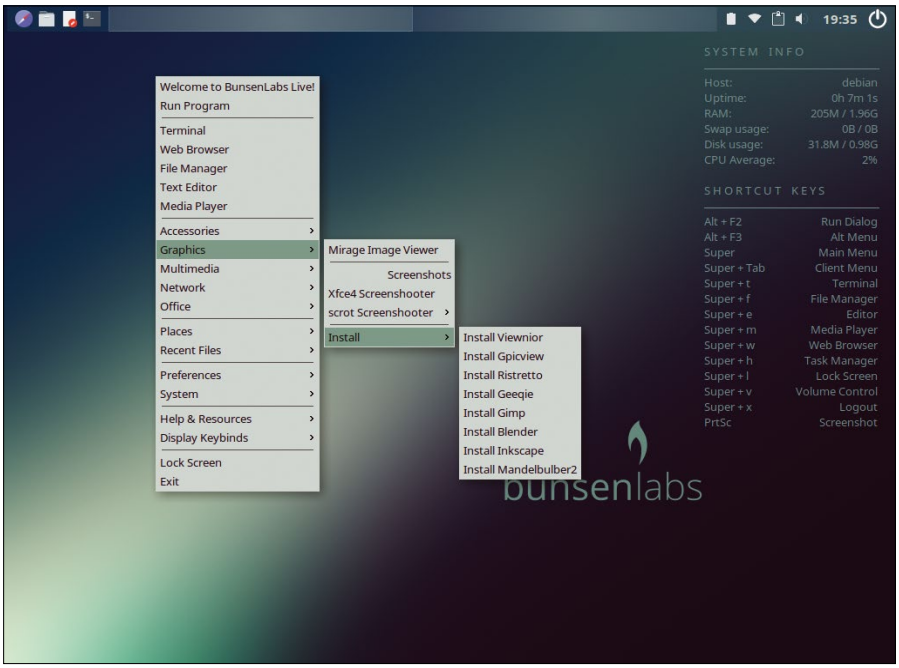


Figure 5: BunsenLabs Helium lets you install numerous applications from the Application menu.

proprietary license and entered the WPA2 key correctly. Consequently, the setup was trapped in an infinite loop whenever it tried to open a connection; in the end, only a reboot helped. However, it turned out that the GRUB boot manager launched from the removable disk was damaged, forcing me to transfer the operating system to a USB stick instead.

When I retried this with the much faster ncurses wizard, the Intel 2200BG WLAN card was detected but, again, not initialized. This time, it was at least possible to carry out the following installation steps in the fallback menu and continue to set up the system.

Welcome!

After completing installation, BunsenLabs Helium starts up with the standard Openbox screen, with no change in the software as compared to the Live version. A welcome screen on the desktop, displayed in a terminal window, runs an optional post-installation script to complete the basic system configuration after initial startup. To run it successfully, though, you must be connected to the Internet (Figure 3).

On my test system, the WLAN card, which caused problems during the installation of the operating system, was easily set up during operation, and the script then immediately updated the system.

Besides updating, the script helps to set up additional repositories. You can also add additional components, such as a Java Runtime Environment, background images, or the Adobe Flash Player.

Desktop Customization

Openbox provides very detailed options that let you customize your desktop. To do this, go to the corresponding entries for Openbox, Conky, tint2, and Compositor in the Preferences menu. In the Openbox submenu, for example, the Menu Editor lets you modify the menu. It lets you create new starters and group entries in submenus (Figure 4).

Additional Applications

In order to retrieve standard applications, such as additional LibreOffice programs, from the Internet and integrate them into the system, you will find various options in the individual menus (Figure 5). Thanks to these options, you can install the desired applications on your hard disk with a mouse click. In some cases, scripts will be started that prompt for additional parameters. If there is still something missing, the only thing left to do is to select Synaptic.

Dependency Errors

Some applications do not integrate directly into the system after installation due to outdated dependencies. Scripts integrated into the menu structure, as well as the classic installation via Synaptic, then abort with error messages. To be safe, open a terminal and enter the following commands to update the system:

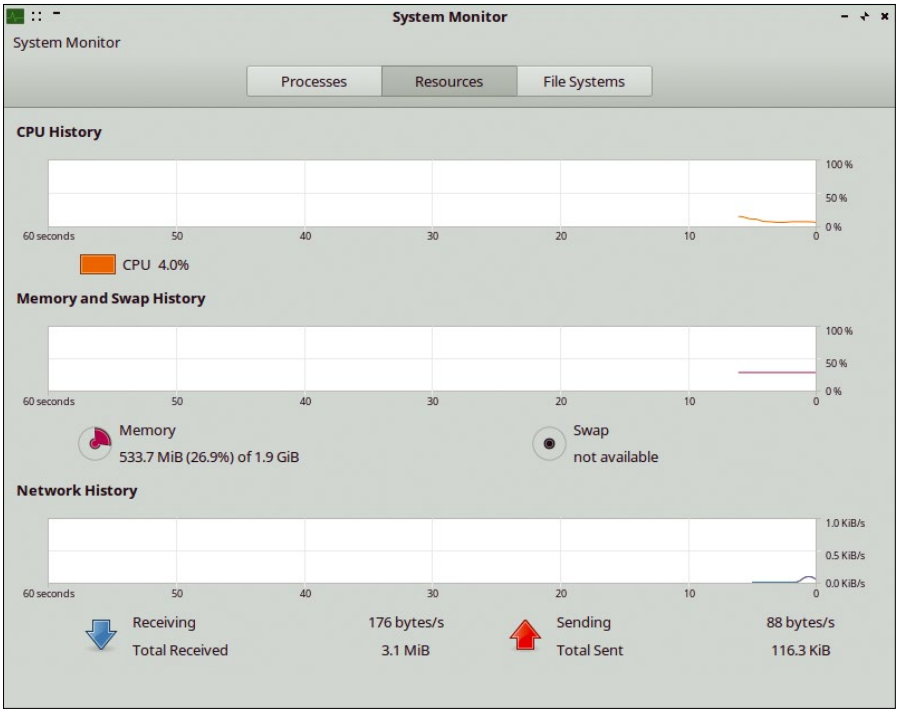


Figure 6: BunsenLabs Helium ensures an acceptable base load even on ancient hardware.

```
$ sudo apt update
$ sudo apt upgrade
```

If you continue to receive error messages due to dependency issues during manual installation of applications or if you cannot find all programs in the Synaptic graphical front end, the repositories may not have been fully loaded.

In this case, open the `/etc/apt/sources.list` and append the following:

```
deb http://httpredir.debian.org/debian
  stretch main non-free contrib
```

After saving the file, run the `update` and `upgrade` commands again to update the repositories and the system. After doing this, the interactive scripts and Synaptic worked fine in my test.

You have access to several thousand packages from the individual repositories. If you use the scripts to integrate individual programs into the system, the corresponding entries then disappear from the menu structure. In their place, you will then find the new program.

Resources

To see if BunsenLabs Helium fulfills its promise of a full-fledged operating system for older hardware, we tested it on a 2005 HP Compaq laptop, with an Intel Pentium M single-core processor that runs at a clock speed of 1.86GHz with 2GB RAM and uses a PATA-SSD as mass storage.

On this setup, BunsenLabs Helium proved that you can still work productively on an antiquated computer. The resource requirements in terms of main memory and CPU load were kept within narrow limits, even when popular standard applications were running (Figure 6).

In contrast, clear latencies were noticeable when starting heavyweights like Firefox or Gimp. The latencies were due not only to these applications' resource requirements, but the Pentium M CPU (built into the test system) was incapable of multithreading. As a result, the system ran at full capacity when launching large programs.

Conclusions

BunsenLabs Helium delivers a respectable interface from the lightweight Openbox window manager that even supports compositing and visual effects. Thanks to Debian, the system is a stable workhorse. In everyday use, it impresses with extremely frugal resource requirements, which in many cases means that you can work smoothly even on older computers.

However, the user interface requires some basic configuration knowledge and is not as easy to modify as KDE, for example. For users who are looking for a lean but cleanly configured system for older hardware, BunsenLabs is definitely worth a try. ■■■

Info

[1] BunsenLabs Helium: *https://www.bunsenlabs.org*

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

Author

Erik Bärwaldt is a self-employed IT-admin and technical author living in Scarborough (United Kingdom). He writes for several IT-magazines.

Exploring the FSF's free distributions for the desktop

Free as in Really Free

The Free Software Foundation maintains a list of GNU/Linux distributions that meet their strict standards for free software – and your distro probably doesn't qualify. Meet the distros that pass the test. *By Bruce Byfield*

Linux is frequently called a free operating system, but that all depends on what you call free – and what you call an operating system. When Richard Stallman launched the Free Software movement in 1983, he had a very specific vision in mind that was codified in what became known as the Four Software Freedoms (see the "Four Freedoms" box) [1]. Although the *Linux kernel* is distributed under a free license and is thus classified as free software, many other components are commonly packaged with the Linux kernel, and some of them aren't so free. Furthermore, a *Linux distribution* (or GNU/Linux distribution in the parlance of the Free Software community) is more than just the kernel and often contains thousands of software packages, including applications, libraries, firmware, drivers, codecs, and other components – all with their own licenses and development goals.

The truth is, since 1996, the Linux kernel has included what are known as proprietary or binary blobs – software or firmware that is free to distribute but is not free-licensed and does not come with source code. These blobs often include drivers for WiFi, sound, or Ethernet. To make matters worse, many distributions come with proprietary tools that require proprietary software, such as VirtualBox.

Most mainstream distros accept binary blobs, because they believe including these components with the distribution makes the system more user-friendly and improves hardware compatibility. The Free Software Foundation (FSF), however, considers binary blobs a violation of the Four Software Freedoms, because these blobs are distributed without source code and therefore cannot be studied or improved. Moreover, since the blobs cannot be studied, no one can be sure what they contain, and they are a potential threat to privacy and security.

Even if a distribution ships without any binary blobs or other non-free software, if the developers maintain a non-free software repository, the distro loses the endorsement of the FSF. For instance, many Linux users consider Debian the quintessential free Linux, but it is not regarded as free by the FSF, because it supports a non-free repository. According to the GNU website, "Debian's Social Contract states the goal of making Debian entirely free software, and Debian conscientiously keeps non-free software out of the official Debian system. However, Debian also maintains a repository of non-free software. According to the project, this software is 'not part of the Debian system', but the repository is hosted on many of the project's main servers, and people can readily find these non-free packages by browsing Debian's online package database and its wiki."

In addition to its objection to binary blobs and non-free repositories, the FSF also withholds the title of "free" from distros that do not meet its standards for:

- Documentation – all documentation included with the project must come with an open license, and, according to the FSF, "it must take care not to recommend non-free software."
- Trademarks – trademarks are another form of intellectual property, and the

Lead Image © skvoor, 123RF.com

FSF does not support including products that place restrictions on distribution of trademarks.

- Name confusion – the common practice of developing parallel versions of a software product (one free "community" version and one non-free proprietary version) and giving them the same or a similar name leads to confusion and promotes non-free software.

In short, the FSF has very strict requirements for what is a free distribution, and very few Linux distros qualify. However, a small but dedicated group of developers and maintainers continue to work to produce functional Linux systems that meet the FSF's Free System Distribution Guidelines [2].

The FSF maintains a page that lists completely free distributions [3]. The list has never been long. Major distributions have not made it a priority to meet the strict FSF requirements for a free system, although some, like Gentoo and Arch, have information on their wikis about how to deblob.

Currently, the FSF list of free distributions includes only nine entries. Of those nine, Ututo is included only for its historic interest as the first deblobbed distribution. Similarly, Dyne:bolic has not had a release for eight years, and gNewSense is on hiatus, although developer Matt Lee says that he plans to revive it. As for Guix, it exists mainly to demonstrate the Guix package manager, rather than as an everyday alternative. In the end, only four distributions are listed that an average user can use on a modern computer: Dragora, Parabola, PureOS, and Trisquel.

Four Freedoms

According to the GNU project and the Free Software Foundation, a program is free software if the program's users have the four essential freedoms:

- The freedom to run the program as you wish, for any purpose (freedom 0).
- The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help others (freedom 2).
- The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

A program is free software if it gives users adequately all of these freedoms. Otherwise, it is non-free. While we can distinguish various non-free distribution schemes in terms of how far they fall short of being free, we consider them all equally unethical.

Dragora GNU/Linux

Like Ututo, the Dragora project [4] is based in Argentina. The distribution is currently in beta, and English documentation is sparse, although a YouTube video can get users started [5]. Even finding basic information like logging into the Live DVD with *root* and *Dragora* can be challenging.

However, those with patience to persist will find much that is original in Dragora. For example, it is built from scratch and uses its own package manager called qi, as well as using `runit` for an init system. The result is a light and highly responsive distribution. Admirers of Slackware should be responsive to Dragora's philosophy – others not so much (Figure 1).

Figure 1: Dragora GNU/Linux takes a do-it-yourself approach reminiscent of Slackware.

Parabola GNU/Linux

Parabola [6] (Figure 2) was originally proposed on the gNewSense IRC and went on to become a distribution in its own right. Drawing on the Arch Linux repositories, it is available for the i686, x86-64, and ARMv7 architectures. An alternative spin called TalkingParabola includes Braille and sound tools for the visually impaired.

Parabola's main difference from Arch is its dedication to a totally free distribution. According to its Wikipedia entry, some 700 packages were removed to reach this goal. The default installation produces an LXDE interface, with a limited set of packages designed to be lightweight.

Parabola can be installed either from a Live DVD or by changing the repositories for an Arch system to Parabola's and then updating. The result is a serviceable but relatively uninspired distribution, best suited to older hardware – and to those already familiar with Arch.

Figure 2: Parabola is a free distribution based on Arch Linux.

Figure 3: PureOS is a distribution used on Purism's line of Librem laptop computers.

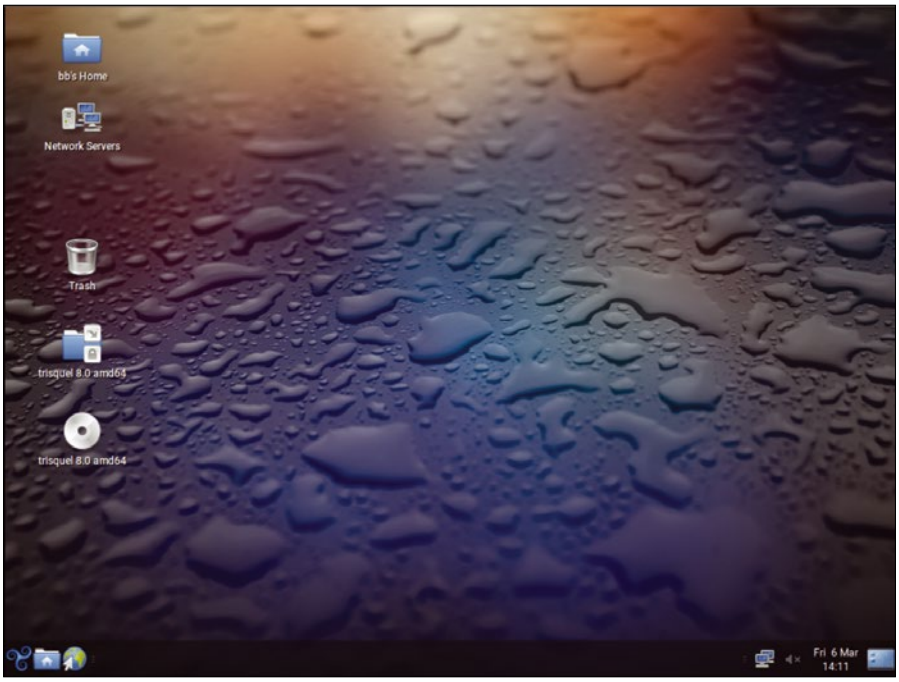


Figure 4: Trisquel is currently the most active free distribution.

PureOS

PureOS [7] is the Debian and Gnome-based operating system used on Purism's line of Librem computers (Figure 3). It is one of the most recent additions to the FSF list, having been added for the first time in 2018, and the Librem laptops have been awarded the Respect Your Freedom certification by the FSF as well. So far as I have been able to determine, PureOS is the only free distribution to run Wayland as a graphical environment.

On the Purism website, PureOS is described as "A user-friendly, secure, and freedom-respecting OS for your daily usage. With PureOS, you are the only one in control of your digital life." This description makes PureOS sound like an ideal choice, except for one thing: No details about the security and privacy have been released. Nor are many details from the selection of software included in the interface. The few that are obvious are small touches, such as including Enigmail for sending encrypted email and defaulting to DuckDuckGo as a search engine. In addition, PureOS gives far more notifications than the average distribution, allowing users to know far more about what is happening.

Possibly more security may lie in configuration defaults, but the references to security presumably refer to features such as the camera and WiFi kill switches that are part of the hardware. For this reason, PureOS seems to best suited to those who run Librem laptops rather than any general hardware.

Trisquel

Trisquel (Figure 4) [8] is by far the most active project on FSF's free distribution list. It has been on the list since 2008 and survives on public donations. Based on Ubuntu, Trisquel is available in four versions: the Standard release, which uses MATE as a desktop environment; Trisquel Mini for lighter or older systems, which uses LXDE; Trisquel Sugar Toast for children; and Trisquel Net Install, which installs over the Internet. KDE Plasma and other desktop environments are available in the repositories, and over 51 localizations are available.

Trisquel installs with a minimum of applications – mostly Gnome Technology. Probably the most innovative feature is Abrowser, a modification of Firefox with the name changed for copyright reasons (Figure 5). Abrowser is notable for its numerous security settings, including options to disable JavaScript, geolocation, and the loading of custom fonts. These options are clearly explained, with the fact clearly stated that some of the options may interfere with normal web browsing. However, Abrowser reflects the fact that the browser is generally the most vulnerable part of the operating system.

As a distribution, Trisquel has a long and well-received history. If you wish to run a free system, try a version of Trisquel first.

Roll Your Own

If none of these alternatives suit you, try converting your distro of choice to a free distribution. Before you start, research your hardware, so you know if it requires proprietary blobs. If you wish to run an all-free system, you'll need to replace any hardware that requires non-free drivers.

Next, edit the repositories and their sections so that only those that contain free software are active. In Debian, that means making sure that the Stable, Testing, and Unstable repositories have only the main section enabled and not the contrib and non-free sections. In Ubuntu, disable the Restricted

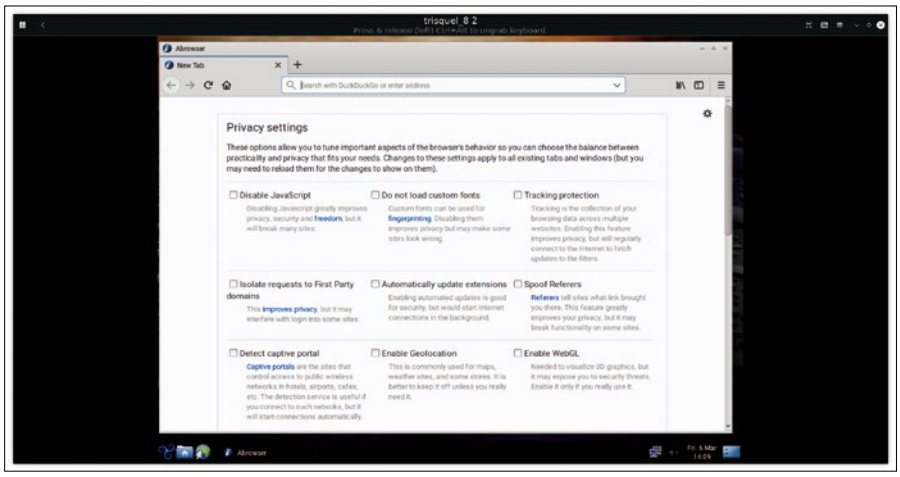


Figure 5: Trisquel features Abrowser, a high-security fork of Firefox.

and Multiverse repositories, while in Fedora, check that only the rawhide and fedora repositories are active. Each distribution organizes its repositories differently, but finding which contain proprietary software is usually a matter of seconds. Once only free repositories remain, you will not be able to install proprietary packages without modifying the list of repositories.

Next, you need to remove any proprietary software already installed. Download and re-run `deblob`, which removes proprietary software [9], and `deblob-check`, which checks kernels, source code, and patches. Some distributions also have their own tools, like Debian's `vrms` (virtual Richard M. Stallman), which detects non-free applications [10] (Figure 6). When you want to upgrade a kernel, download one from Linux-libre [11] or its mirrors. Check your distribution for instructions about compiling and installing kernels. Over the years, the procedure has been improved and documented, so that freeing your Linux installation is mostly a matter of patience.

At the end of the process, your system will not run any faster, although it might be more secure. But is it worth it? Only you can decide. Still, if software freedom matters to you, then the effort is worthwhile, if only so you can honestly say that your operating system is truly free. ■■■

Info

[1] The Four Software Freedoms: *https://www.gnu.org/philosophy/free-sw.en.html*

[2] Free System Distribution Guidelines: *https://www.gnu.org/distros/free-system-distribution-guidelines.html*

[3] FSF list of free distros: *https://www.gnu.org/distros/free-distros.html*

[4] Dragora GNU/Linux: *https://www.dragora.org/en/index.html*

[5] Dragora video: *https://www.youtube.com/watch?v=7IHQ3_HqJcw*

[6] Parabola GNU/Linux: *https://www.parabola.nu/*

[7] PureOS: *https://www.pureos.net/*

[8] Trisquel: *https://trisquel.info/*

[9] Deblobbing scripts: *https://www.fsfla.org/svn/fsfla/software/linux-libre/scripts/*

[10] vrms: *https://packages.debian.org/jessie/vrms*

[11] Linux-libre: *http://www.fsfla.org/ikiwiki/selibre/linux-libre/*

```
root@nanday:~# vrms
              Non-free packages installed on nanday

amd64-microcode                 Processor microcode firmware for AMD CPUs
bluez-firmware                  Firmware for Bluetooth devices
firmware-amd-graphics           Binary firmware for AMD/ATI graphics chips
firmware-linux-nonfree          Binary firmware for various drivers in the Linux k
erne
firmware-misc-nonfree           Binary firmware for various drivers in the Linux k
erne
intel-microcode                 Processor microcode firmware for Intel CPUs

              Contrib packages installed on nanday

alsa-firmware-loaders           ALSA software loaders for specific hardware
flashplugin-nonfree             Adobe Flash Player - browser plugin
iucode-tool                     Intel processor microcode tool
```

Figure 6: Debian includes the tongue-in-cheeked named `vrms` (Virtual Richard M. Stallman), which detects non-free software.

A Bash DIY data extraction tool

Data Collector

With some simple Bash commands, you can gather, parse, and filter text data into CSV files ready for your favorite statistical application. *By Răzvan T. Coloja*

If your research involves pulling large amounts of text data from the Internet, you can gather and process that data from the command line with a few simple Bash commands and turn it into a CSV file for your favorite statistical application, such as SPSS, R, or a MySQL table. In this article, I will show how to accomplish this with a project that examines the Romanian university dropout rate.

The data I need comes from 97 universities. For confidentiality reasons, chances are slim that I can get access to each university's database, but I can obtain that information legally from their website. (However keep in mind that many websites have licenses that prohibit web scraping. This article does not attempt to address copyright and other legal issues related to this practice. See the site's permission page and consult the applicable laws for your jurisdiction.) To gather my data, I could search for the word *abandon* (Romanian for *dropout*) on each of the 97 websites, but that would be tedious. Furthermore, each website may use a different content management system (CMS), so my search might not return the desired results. Instead, an easier option is to download all 97 websites in their entirety and recursively search their text content on my local hard drive. Linux lets you do this with the command shown in Listing 1.

Listing 1: Downloading Websites

```
wget -cv --progress=bar --connect-timeout=30 --force-directories
  --ignore-length -r -l 7 --convert-links --waitretry=61 -R gif,jpg,png,svg,pdf
  http://www.address.ro
```

Retrieving Data

In Listing 1, `wget` is a command-line utility in Linux and other POSIX-compliant operating systems used to download files from servers. It can be used as a mass downloader, and you can specify exactly which type of files you want downloaded and which type of files `wget` should disregard.

In the case of an interruption, the `--continue` attribute (`-c`) allows `wget` to continue where it left off without re-retrieving the data it has already copied when access is granted again. This can save you a lot of time and ensure the process won't loop due to frequent connection glitches.

The `--verbose` attribute (`-v`), when used in conjunction with `--progress=bar`, lets you see what the command does in real time by watching the Linux command-line interface. This is helpful if you want to see `wget`'s downloading progress and look out for possible errors.

The `--connect-timeout` attribute is set to 30 seconds, which means that if no TCP connection can be established with the target server in 30 seconds, `wget` will stop trying. Similarly, `--waitretry` is set to 61 seconds, which means that `wget` will wait for 61 seconds before again trying to retrieve a file it failed to download. The 61 seconds are necessary, because some servers might experience temporary disconnections or might limit the number of downloads to just a few per minute. With 61 seconds, `wget` will wait a little over a minute before trying to download the file again; usually this method works, although it is time-consuming.

The `--force-directories` attribute, which is very important for gathering research data, ensures that you get an exact replica of the website you are downloading, one that maintains the same directory structure as the original.

`--ignore-length` guarantees `wget` will successfully download the target website.

Lead Image © Mike Espenhain, 123RF.com

Some servers send bogus *Content-Length* headers that make `wget` think a file has not fully been retrieved, so `wget` will try and download it again. The `--ignore-length` attribute will ignore the *Content-Length* header and thus circumvent the problem.

`--recursive` (`-r`) forces `wget` to enter each subdirectory and retrieve every file in that subdirectory, not just the main branch. The `-l 7` attribute specifies that `wget` should go up to seven levels deep in the server's directory structure to gather data.

Naturally, you want to store an exact copy of the target web server locally. However, any hyperlink present in a downloaded HTML or PHP file will still point to the online server and not the corresponding copy stored on the local drive. To fix this and make all the links point to their corresponding local counterparts, use `--convert-links`, which will result in a browsable copy of the website that you can read with the help of your favorite web browser, even without an Internet connection.

Excluding Non-Text Content

Websites also contain non-text content, like image files, other file types (MP3, ZIP, RAR, or PPT), and video formats (AVI, MP4, MKV, or VOB). Since I am only interested in text-based information, I want to exclude these types of files, which will have the added benefit of speeding up the entire process and conserving bandwidth. To do this, use the `-R` attribute followed by a comma-delimited list of extensions. For example, if I wish to discard image files, I would use the following:

```
-R gif,jpg,svg,png
```

For Linux and Unix-based operating systems, which are case-sensitive, you will want to modify this by including capitalized extensions as well:

```
-R gif,GIF,jpg,JPG,svg,SVG,png,PNG
```

You can do the same for any other file extension, especially those representing potentially large files:

```
-R avi,AVI,mpg,MPG,mp4,MP4,mkv,MKV,vob,➥
  VOB,iso,ISO,zip,ZIP,rar,RAR,tar,TAR
```

Exclude everything not representing text content including archives, video files, ISO images, pictures, and Microsoft Office documents.

Downloading the Websites

Finally, at the end of the `wget` command, you can add the target website's address to inform `wget` of the address you want cloned. When dealing with several addresses, you can copy and paste each line in a Bash script (one after the other), make the script executable, and execute it. When `wget` finishes with one website, it will pass on to the next.

Since we are dealing with 97 websites, the easiest way is to have all the URLs in one text file, each one on a separate line, and use the small script shown in Listing 2 to show `wget` where to get each address. This way you only have to launch `wget` once instead of 97 times.

The separate file `addresses.txt` should contain only the target server's web addresses, each on its own line, avoiding any special characters, quotes, and/or spaces.

The downloading process may vary depending on your Internet connection's speed, your CPU speed, and the available RAM. Also, be aware that some websites might be down or might even block you. Many websites do try to block scraping, whether because they are protecting their business interests, or because downloading tens of thousands of files with just as many server interrogations might be seen by the server or the server administrator as a potential denial of service (DoS) attack. Consequently, your IP might be blocked at some point. As a reference point, launching 10 such concurrent scripts on a Linux machine running at 800MHz and 256MB RAM memory took six days to download the 97 Romanian university websites – all on a 1GB Internet connection. Better hardware greatly improves the process. In the end, 392,868 files were retrieved with a total 108,447,924,224 characters.

Filtering the Data

Now that I have gathered the data, I now need to recursively search for occurrences of the word *abandon* in the downloaded files. For this, I will use another Linux command-line utility, `grep`. In the directory containing all the folders with the downloaded websites, launch

```
grep -r -A1 -B1 "abandon" ➥
  * > results.txt
```

`-r` tells `grep` to search every file and subfolder in the current folder from where the command was launched. `grep` will search not only for *abandon*, but also for its variations, including suffixes or prefixes.

However, I am only interested in school dropout and how many times it is mentioned on each university's website. For this, I need to see the word's context and eliminate the unrelated instances. The `-A` and `-B` flags are used to verbosely display the line after and before the occurrence of the word. However, the terminal scrolls rapidly in verbose mode. Since I want to take a closer look at the results, I can output everything to a text file named `results.txt`. As a side note, the `*` character specifies that `grep` should search all downloaded files, whether they are in human-readable format or not.

After a bit of processing, the result will be a text file (`results.txt`) that contains the complete path to the file containing the word *abandon* and its variations, plus the surrounding context. The occurrences are delimited by the characters `--`. I need to eliminate these and replace them with something else, because (as you will see later) the next commands tend to interpret `--` as a specifier of command attributes. So you can either open the `results.txt` file in a text editor and do a search and replace or use a Linux command to automatically do this for you. Keep in mind that in doing so you are also replacing the `--` characters that might be present in legitimate URLs in the file. I will fix this later.

I chose to replace `--` with `12345678` to get rid of all the delimiting lines in the file. I did this, because the next command I use lists the first line after this replaced delimiter – the line containing the local HTTP address of the file:

Listing 2: Downloading Multiple Websites

```
wget -cv --progress=bar --connect-timeout=30 --force-directories --ignore-length
  -r -l 7 --convert-links --waitretry=61 -R gif,jpg,png,svg,pdf $(<addresses.txt)
```

```
grep -A 1 -F 12345678 ➥
  results.txt > 1stline.txt
```

The above command displays the line immediately after the one beginning with `12345678` and outputs the result in another text file called `1stline.txt`. Now I have an almost clean file containing just the addresses of the files that contain the word *abandon* and all its variations, ready to be inserted into a statistical program. However, `wget` appends some of the accompanying text at the end of some of the addresses, and I need to get rid of that using the `sed` editor:

```
sed 's/<.*//' 1stline.txt > ➥
  1stlinefiltered.txt
```

This command will delete everything after the < character leaving only the addresses, without the beginning of the context text appearing on the same line. The resulting output is put into a new file called `1stlinefiltered.txt`.

The instances of *abandon* and its variations might appear in multiple files, some of which are duplicate addresses. I don't need these, since I only want to work with distinct instances of the word located in distinct files. To delete duplicate lines from `1stlinefiltered.txt`, I will use `sort` and `uniq`:

```
sort 1stlinefiltered.txt | uniq -u ➥
  > address_filtered.txt
```

The command-line utility `sort` sorts the file's lines, and `uniq -u` prints in verbose mode only the unique lines of the specified text file. Everything is outputted through a new file called `address_filtered.txt`. Because `wget` sometimes appends a - character at the end of each URL, I must further clean `address_filtered.txt`. To do this, I will use `sed` again to delete the last character of each line present in the file:

```
sed 's/.$//' address_filtered.txt ➥
  > list_final_address.txt
```

This time, everything is outputted in `list_final_address.txt` – a list of URLs pointing to pages that contain the word `abandon`. Since I previously replaced `--` with `12345678` in order to be able to correctly display the first line below each delimiter, some of the web addresses that contained `--` in their URL also got their file path changed. All I need to do is take the `list_final_address.txt` file and do a search and replace for `12345678` with `--`.

Now `list_final_address.txt` is clean, and all I have to do is convert it to CSV format for easy importing to a statistical application:

```
cat list_final_address.txt ➥
  > address.csv
```

This final file, `address.csv`, will give you a column of web addresses in a statistical application like SPSS, each representing a location corresponding to a file containing the word `abandon`. In addition to the web address column, I also need a column with *abandon*'s corresponding context. The file `results.txt` still contains this information, and I can use it to extract what I need. You can do this with Bash or a Python script.

Listing 3: Bash Extraction

```
grep -E -o ".{0,50}abandon.{0,50}" results.txt > abandon50.txt
```

Listing 4: Python Extraction Script

```
01 #!/usr/bin/python
02
03 """Custom work for Razvan T. Coloja, placed in the public domain by the author
   - Radu-Eosif Mihailescu.
04 """
05
06 import sys
07
08 MAGIC_WORD = 'abandon'
09
10 def main(argv):
11 with open(argv[1], 'r') as faddr:
12 addresses = set(l.rstrip() for l in faddr)
13 with open(argv[2], 'r') as fres:
14 the_text = set(l.rstrip() for l in fres)
15
16 for address in addresses:
17 for line in the_text:
18 if line.startswith(address):
19 where_found = line.find(MAGIC_WORD)
20 if where_found != -1:
21 if where_found > 50:
22 start_excerpt = where_found - 50
23 else:
24 start_excerpt = 0
25 print '"%s","%s"' % (
26 address,
27 line[start_excerpt:where_found + len(MAGIC_WORD) + 50])
28
29 if __name__ == '__main__':
30 main(sys.argv)
```

Bash Extraction

With Bash, you use `grep` again (Listing 3). Listing 3 takes `results.txt`, searches it for the word `abandon` and all its variations, and displays in verbose mode the 50 characters surrounding the word. One hundred characters with the keyword in the middle should suffice to be able to tell if the context is relevant. Everything is outputted to a new text file called `abandon50.txt`. From this file, I trim out duplicate lines with:

```
sort abandon50.txt | uniq -u ➥
> abandon50_filtered.txt
```

Success: The resulting file, `abandon50_filtered.txt`, contains a column of text corresponding to each address in `address.csv`. The problem with this approach is that each server I initially parsed with `wget` uses a different CMS. Some university servers might use open source solutions, such as Wordpress or Joomla, while others use custom solutions. Consequently, no two sites are the same.

In addition, most sites contain special characters in their URL (e.g., *$* and *%*), and `grep` output has difficulties with these characters. A manual search-and-replace for special characters such as `%20` should fix the problem. Alternatively, you can use another Linux command-line utility, `html-2text`, that trims special HTML tags from files, leaving only clean human-readable text behind. Once this is done, `grep` should have no problem in performing correctly.

Python Extraction

If you want to use Python to extract the occurrences of the word *abandon* surrounded by 50 characters on each side, you can use the Python script in Listing 4. This script will also filter out special characters from URL addresses.

Put the script from Listing 4 in a text file and name it `needleinhaystack.py`. Make it executable in Linux with the following command:

```
chmod +x needleinhaystack.py
```

You need to have Python installed to make this script work. The script will compare the file containing the addresses with the one containing both the addresses and the associated context, trim context to about 100 characters with *abandon* in the middle, and structure everything into two columns that are ready to be imported into a statistics application.

Putting It All Together

You now have the data you need to do your desired analysis. To save typing each command individually, you can put the above commands into a single Bash script as shown in Listing 5.

Then add the addresses to `addresses.txt`, each on one line, and save the file in the same folder as the Bash script in Listing 5. Make the script executable with

```
chmod +x scriptname.sh
```

Then launch it with `./scriptname.sh`.

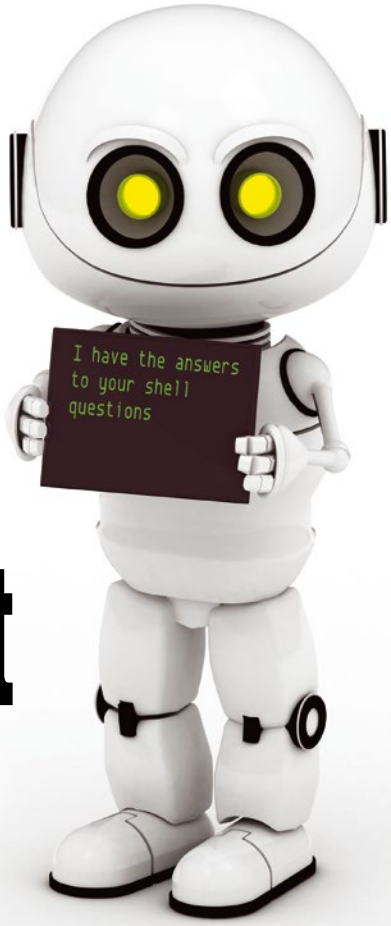
With a few simple Bash commands, you have a DIY text data collection tool that delivers a CSV file for use in your favorite statistical application.

Listing 5: Complete Bash Script

```
01 #!/bin/bash
02 # download the websites specified in addresses.txt one by one
03 wget -cv --progress=bar --connect-timeout=30 --force-directories --ignore-length
   -r -l 7 --convert-links --waitretry=61 -R gif,jpg,png,svg,pdf $(<addresses.txt)
04 # recursively look for the word "abandon" and its variations and print in verbose
     mode the line before and after the keyword so we can take a quick look at the
     context
05 grep -r -A1 -B1 "abandon" * > results.txt
06 # find every line that starts with the "--" delimiter and replace it with
     "12345678" using your favorite text editor
07 # list the first line after "12345678"
08 grep -A 1 -F 12345678 results2.txt > 1stline.txt
09 # delete everything after the "<" character
10 sed 's/<.*//' 1stline.txt > 1stlinefiltered.txt
11 # list every line only once, without its duplicates
12 sort 1stlinefiltered.txt | uniq -u > address_filtered.txt
13 # remove last character form each line (.html-)
14 sed 's/.$//' address_filtered.txt > list_final_address.txt
15 # create a CSV file containing the web addresses
16 cat list_final_address.txt > address.csv
17 # replace "12345678" with "--" in address.csv because "--" might appear in the URL
```

Author

Răzvan T. Coloja is a psychologist currently finishing his Bachelor's degree and PhD candidacy in sociology. He has been a passionate Linux user and OSS supporter since 1998 and has an interest in SBC clusters, CircuitPython, and machine learning.



Cheat sheets for the shell

Interactive Cheat Sheet

When the history function fails and the manpage is too long, navi comes to the rescue with an interactive cheat sheet for the shell.

By Ferdinand Thommes

What exactly was that parameter for creating an archive with `tar`? Command-line aficionados know that if nothing shows up with the Bash `history` command, your only option is to look at the manpage. While manpages are generally useful, some are so extensive that the time spent searching for information is disproportionate to the task at hand. An alternative to this problem is an interactive cheat sheet.

While there are plenty of cheat sheet tools on GitHub, the relatively young navi [1] offers both good functionality and an eye-pleasing design. Navi helps users browse built-in (DIY or downloaded) cheat sheets to display options and arguments for commands or directly execute the commands. Navi (as well as similar tools) lets you interactively learn about newly discovered commands and explore all their possibilities.

Installation

You can quickly install navi from the website (Listing 1). In addition, you can optionally install navi in an arbitrary directory (line 6) instead of in your path.

Navi does have a couple of dependencies. To avoid having navi complain, you must install the fzf string matching algorithm, a search tool that uses fuzzy searches to quickly find specific strings in a longer string or text.

You can find fzf in the Debian, Fedora, and Arch Linux repositories, where you can install it with the standard package manager. Alternatively, Mac OS X users can use Homebrew [2] for the installation. For Z shell [3] users, navi's developer offers a guide for integrating the widely used Oh My ZSH [4] configuration file.

Navi's only other dependency is ncurses, which should be preinstalled on every distribution.

Listing 1: Installing navi

```
01 $ sudo git clone --depth 1 https://github.com/denisidoro/navi /opt/navi
02 $ cd /opt/navi
03 ### Install in $PATH
04 $ sudo make install
05 ### Alternatively
06 $ ./scripts/install <directory>
```

Since work on the relatively young navi is ongoing, you will want to update the software regularly. When a new version appears on GitHub, you can update navi quickly with `git` (Listing 2).

Getting Started

First, you will want to briefly familiarize yourself with navi's arguments and options by calling `navi --help` in a shell. Then start the program by entering `navi`. Navi opens a current list of 242 embedded cheat sheets (Figure 1).

In the three-part view in Figure 1, you can see the application or environment from which the commands originate on the left, the tasks to be performed with them in the middle, and the corresponding command on the right. You can choose to limit the display of commands to one category. For example,

```
navi query git
```

only shows commands for `git`. Alternatively, you can narrow down the results in the window by typing `ip addr`, which

Listing 2: Updating navi

```
$ cd "$(navi home)"
$ sudo git pull
```

Lead Image © Stanislav Volchenkov, 123RF

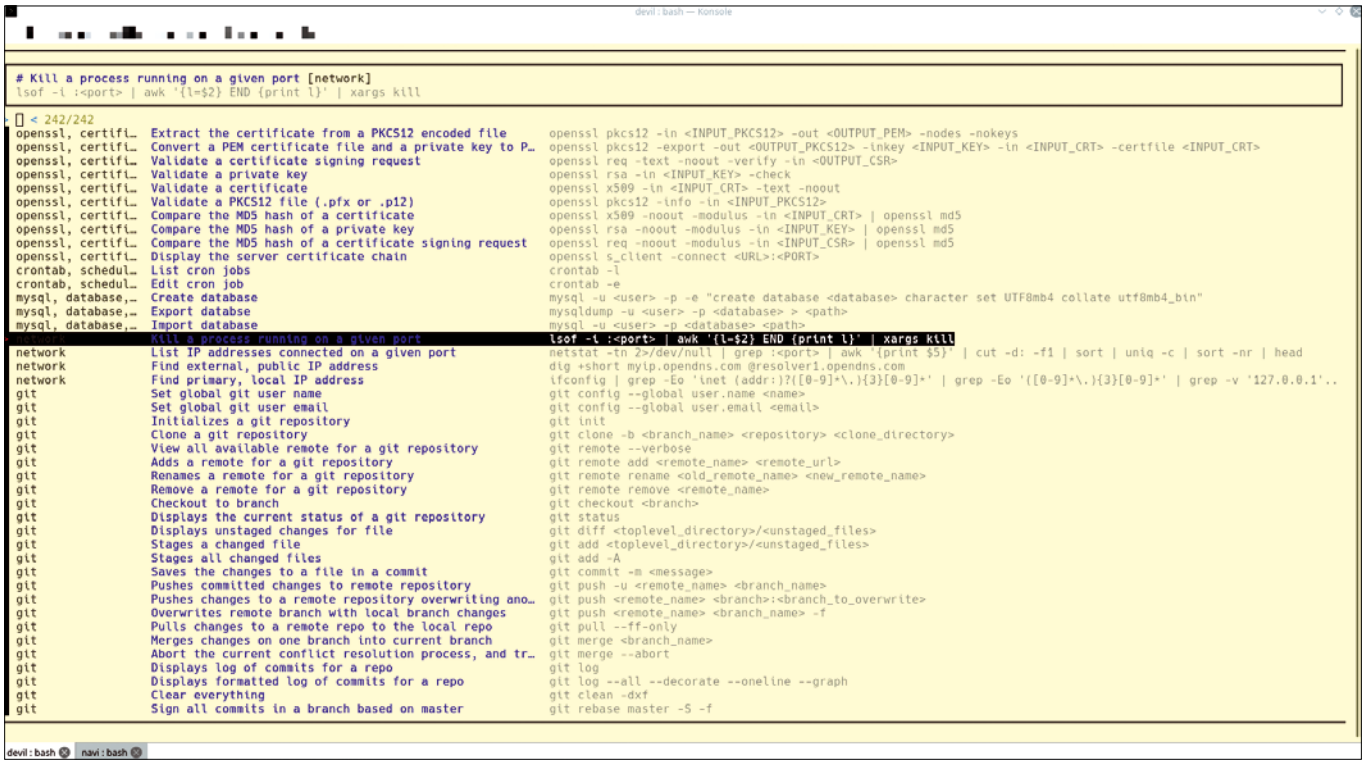


Figure 1: Navi displays the current list of cheat sheets at startup.

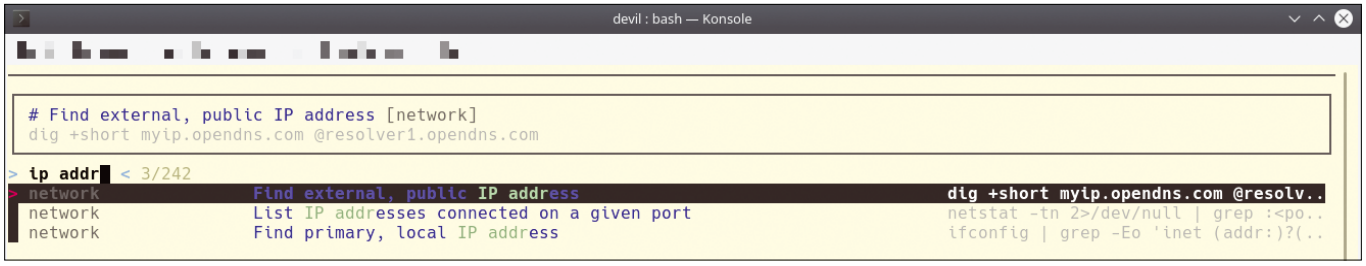


Figure 2: Keywords are used to limit the commands to a specific topic directly in the window.

results in the three commands that deal with IP addresses (Figure 2).

Direct Execution

Navi's interactivity comes into play when you double-click on a selected command. If the command requires no further input, the program passes it directly to the shell. For instance, clicking on the *List running services* task from the *systemctl, service* category takes you directly to a list of running services (Figure 3). In this example, direct execution of the command does not pose a problem, because the command only prints a list and makes no other changes to your system.

Hitting Pause

What if you are not sure that you've chosen the correct command? For instances where the command will change your system, you can add the `--print` option to the command to prevent it from exe-

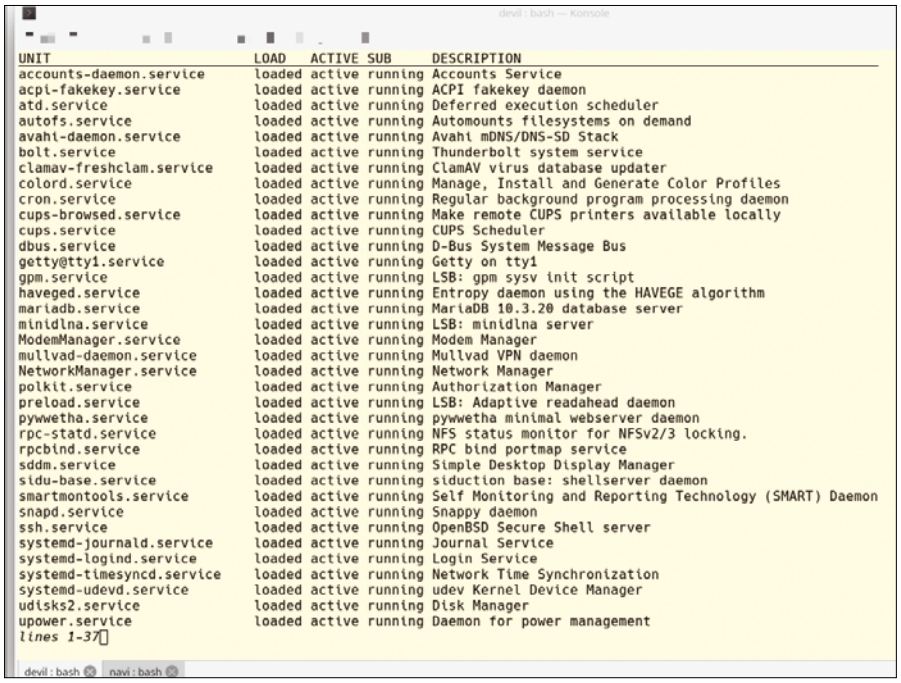


Figure 3: Commands that do not change the system (e.g., listing active services) can be executed directly without any risk.

cuting directly. If you want to use this option as a default, define a corresponding alias, such as

```
alias navi='navi --print'
```

and add it in your `~/.bashrc`.

You also might want to use the `--print` option for commands that require further input before execution. For example, the *Create a tar containing files* task from the *compression* section requires two pieces of input from the user; the software prompts you for them when you click on the command. For `name:`, enter the desired name of the tar archive, while `files:` expects the path to the files you want to compress.

Once you have confirmed both entries, the tool will display the command that creates the corresponding archive (Figure 4). If you have used the `--print` option, navi will not execute this command, giving you the opportunity to determine if the command has the desired effect. If the command does what you expect, then copy the command and run it.

Online Cheat Sheets

Another way to use navi is to enter a command directly as an argument at launch time. For example, entering

```
navi search apt
```

will return eight commands for Debian's `apt` package tool. These commands do not originate from the installed navi package; instead, they come from online sources, such as `cheat.sh` [5], which offers additional cheat sheets (Figure 5).

Figure 4: The command shown here for creating a tarball requires the user to enter a name and a path before the tool shows the command.

Integrating the Widget

You can also add navi to the shell as a widget, which helps keep the history up to date and lets you edit the command to suit your needs before running it.

To use the software as a widget, add the following line

```
source "$(navi widget bash)"
```

to your `~/.bashrc` or `/etc/bash.bashrc` file. This lets you launch navi by pressing Ctrl + G in the future. If the keyboard shortcut is already assigned on your system, or if you want to assign a different shortcut, you can define it in `/opt/navi/navi.plugin.bash`.

DIY

If you are missing an important command, you can create your own cheat sheet for the command and call it in navi. Use a file from `/opt/navi/cheats/` as a template (for a detailed explanation of the syntax, see GitHub [6]). Then call the new cheat sheet with a command like the one shown in line 1 of Listing 3 or include several directories in `$NAVI_PATH`, separated by colons (line 2).

Listing 3: Calling a new cheat sheet

```
01 $ navi --path "/<MyOwn>/<Cheatsheets>"
02 $ export NAVI_PATH="/<MyOwn>/<Cheatsheets>:/<Even>/<more>/<Cheatsheets>"
```

Conclusions

While the built-in cheat sheets for Docker, Kubernetes, and Git focus strongly on developers' needs, they also contain many cheat sheets useful for desktop users. In particular, navi makes it easier for console gamers to handle complex command-line commands.

In addition, navi can be extended by integrating online or DIY cheat sheets. However, navi's developer recommends using the `--print` option to check commands from online cheat sheets.

Finally, Katacoda [7] lets you test navi prior to installation to see if it's a good fit for your needs. If it isn't, you will find other similar tools on GitHub, such as `cheat`, `cmdmenu`, BEAVR, and howdoi. ■■■

Info

[1] navi: *https://github.com/denisidoro/navi*
[2] Homebrew: *https://docs.brew.sh/Homebrew-on-Linux*
[3] Z shell: *http://zsh.sourceforge.net*
[4] Oh My ZSH: *https://ohmyz.sh/*
[5] cheat.sh: *http://cheat.sh/*
[6] navi syntax: *https://github.com/denisidoro/navi#cheatsheet-syntax*
[7] Katacoda: *https://www.katacoda.com/denisidoro/scenarios/navi*

Author

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

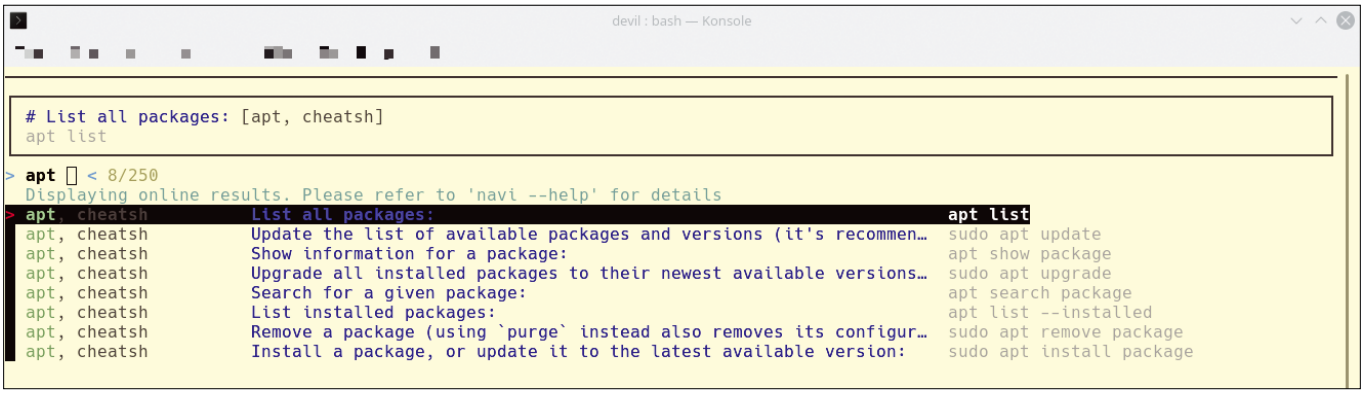


Figure 5: You can use navi to integrate commands from online cheat sheets. However, you should investigate this type of command more thoroughly with `--print` before you take the plunge.

The sys admin's daily grind: darkstat

Spry Methuselah

Thanks to its minimal footprint, 20-year-old darkstat hardly generates any noticeable load even on low-powered systems, making it the perfect monitoring tool for Charly's home utility room.

By Charly Kühnast

Next to our kitchen, there is a small utility room. I don't think its floorspace is even two square meters. In addition to the usual building services, such as fuse box, there are two firewalls, a web and mail server, network attached storage (NAS), and a large switch.

The tiny router supplied by my Internet provider sits a little intimidated in the corner. I downgraded it to something like the IT equivalent of a flow heater. It opens the connection to the provider and passes it to the firewall. I have switched off everything else, like WLAN, telephony, and the DHCP server; I prefer to do that myself, on my own hardware.

You need to monitor what you run. For long-term monitoring of loads and latencies, I use Munin and SmokePing. But if I just want to have a quick look at what currently is happening on the firewall interface, darkstat [1] is the hero of the day.

Darkstat, a true Methuselah at the ripe old age of almost 20, has been under the GPL license since 2002. I had my first contact with the software when I tried pfSense [2]. Thanks to its minimal footprint, the monitoring tool generates so little system load that it even runs unobtrusively on my ancient NAS box with 128MB RAM [3].

Darkstat gets its data via libpcap; the output comes courtesy of a built-in, lean web server. The most important parameters are stored in a small configuration file, which resides in `/etc/darkstat/` on my Ubuntu test system. Using the configuration file is voluntary; I could ignore it and simply start darkstat at the command line.

The only mandatory parameter is `-i <interface>`. The `darkstat --help` command lists all the other parameters. Be careful with `--syslog`. If you enable this option, darkstat suppresses all console messages. It therefore makes sense not to set this parameter until everything else is working to your satisfaction.

Once darkstat is running as desired, a web server on port 667 displays the current traffic data (Figure 1). It is a pity that darkstat displays the data in bytes, not in bits, but it's fine for a quick overview of what's crossing the wire.

More details can be found in the *hosts* tab. This is where darkstat lists the devices in a table; you can sort by the column headers. This is how I found out, for example, that music streaming is very popular today. My eldest child is embarking on a career as an Instagram influencer, or whatever the kids call it nowadays (Figure 2).

Also practical: darkstat not only displays live data, but also visualizes sessions that you record with Wireshark or Tcpdump. Conclusions: Methuselah has aged with dignity and is still very much needed. ■■■

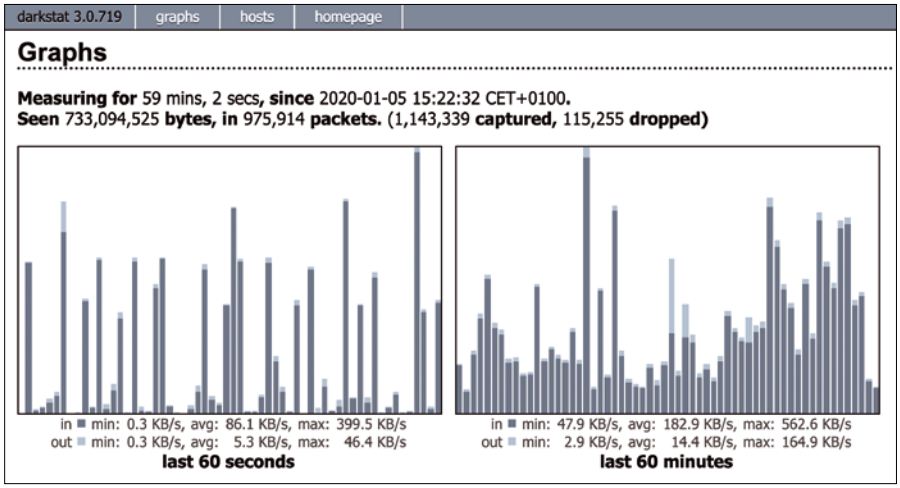
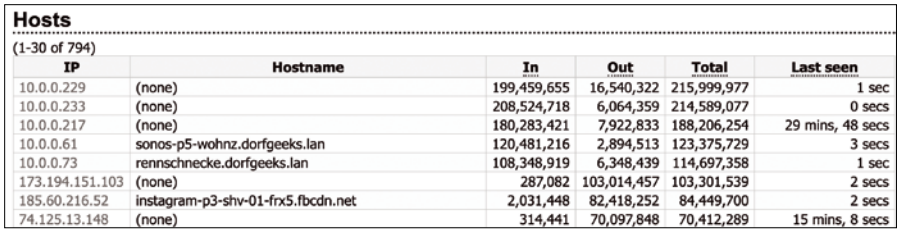


Figure 1: Darkstat returns clear evaluations via a web server on port 667.



Hosts

(1-30 of 794)

IP	Hostname	In	Out	Total	Last seen
10.0.0.229	(none)	199,459,655	16,540,322	215,999,977	1 sec
10.0.0.233	(none)	208,524,718	6,064,359	214,589,077	0 secs
10.0.0.217	(none)	180,283,421	7,922,833	188,206,254	29 mins, 48 secs
10.0.0.61	sonos-p5-wohnz.dorfgeeks.lan	120,481,216	2,894,513	123,375,729	3 secs
10.0.0.73	rennschnecke.dorfgeeks.lan	108,348,919	6,348,439	114,697,358	1 sec
173.194.151.103	(none)	287,082	103,014,457	103,301,539	2 secs
185.60.216.52	instagram-p3-shv-01-frx5.fbcdn.net	2,031,448	82,418,252	84,449,700	2 secs
74.125.13.148	(none)	314,441	70,097,848	70,412,289	15 mins, 8 secs

Figure 2: Darkstat uses the *hosts* tab to list the devices.

Info

[1] darkstat: *https://unix4lyfe.org/darkstat/*

[2] pfSense: *https://www.pfsense.org*

[3] darkstat package for Synology NAS: *https://synocommunity.com/package/darkstat*

Author

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

Scraping highly dynamic websites

Under the Hood

Screen scrapers often fail when confronted with complex web pages. To keep his scraper on task, Mike Schilli remotely controls the Chrome browser using the DevTools protocol to extract data, even from highly dynamic web pages. *By Mike Schilli*

Gone are the days when hobbyists could simply download websites quickly with a `curl` command in order to machine-process their content. The problem is that state-of-the-art websites are teeming with reactive design and dynamic content that only appears when a bona fide, JavaScript-enabled web browser points to it.

For example, if you wanted to write a screen scraper for Gmail, you wouldn't even get through the login process with your script. In fact, even a scraping framework like Colly [1] would fail here, because it does not support JavaScript and does not know the browser's DOM (Document Object Model), upon which the web flow relies. One elegant workaround is for the scraper program to navigate a real browser to the desired web page and to inquire later about the content currently displayed.

For years, developers have been using the Java Selenium suite for fully automated unit tests for Web user interfaces (UIs). The tool speaks the Selenium protocol, which is supported by all standard browsers, to get things moving. Google's Chrome browser additionally implements the DevTools protocol [2], which does similar things, and the chromedp project on GitHub [3] defines a Go library based on it. Go enthusiasts can now write their unit tests and scraper programs natively in their favorite language. I'll take a look at some screen-scraping techniques in this article, but keep in mind that many websites have licenses that prohibit screen scraping. See the site's permission page and consult the applicable laws for your jurisdiction.

Directing Chrome

The Go program in Listing 1 [4] launches the Chrome browser, points it at the *Linux Magazine* web page, and then takes a screenshot of the retrieved content. The whole thing runs at the command line if you type

```
go build screenshot.go
```

followed by `./screenshot`. The user will not see a browser pop up, because `chromedp` normally runs in headless (i.e., invisible) mode, unless otherwise configured. The following command gets the required library code from GitHub and also compiles and installs it:

```
$ go get -u github.com/chromedp/chromedp
```

It takes the compiled program in Listing 1 a few seconds to retrieve the page, depending on your Internet connection and the current server speed; then it saves an image file in PNG format named `screen-`

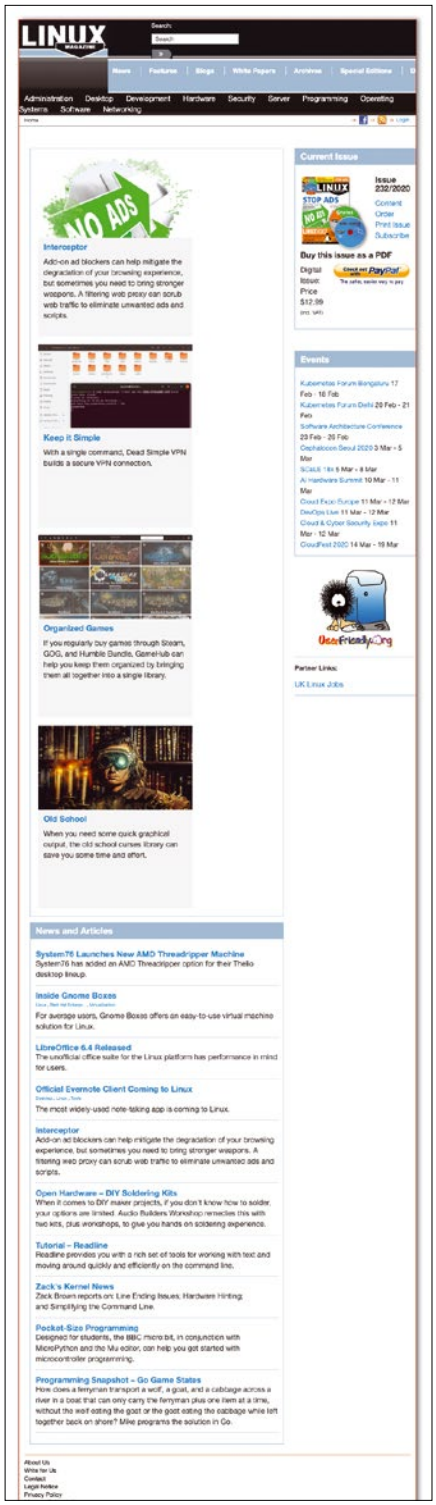


Figure 1: A screenshot of the *Linux Magazine* cover page in a remote-controlled Chrome instance.

Lead Image © Rene Walter, 123RF

shot.png to the hard disk as a result. Since the *Linux Magazine* homepage fills several browser pages in terms of length, giving users a reason to scroll down and explore, the screenshot in Figure 1 is almost 3000 pixels tall.

Listing 1 creates a new chromedp context in line 13 and gives the constructor a standard Go background context, which is an auxiliary construct for controlling Go routines and subroutines. A context constructor in Go returns a `cancel()` function. This function can be called by the main program later to signal to another (maybe deeply) nested part of the program that it is time to clean up, because doors are being closed.

The `Tasks` structure starting on line 17 defines a set of actions that you want the connected Chrome browser to perform, using the DevTools protocol. The `Navigate` task starting on line 18 directs the browser to the *Linux Magazine* website. The second task starting in line 20 is created by the `ActionFunc()` function, a tool to structure new customized tasks in chromedp. In this case, the task creates a screenshot of the web page displayed in the remote browser using the function `CaptureScreenshot()` in line 33.

Wide Open Spaces

Now the question is how far to open the virtual browser, because this setting determines what you see in the screenshot. Is only a fraction of the web page visible or all of it, including the parts that can only be reached by scrolling? If it's the latter, the screenshot needs to capture everything that the user would see if they had an infinitely tall screen with the browser fully extended.

To capture it all, the `GetLayoutMetrics()` function calculates the layout dimensions of the displayed page, and the `viewPortFix()` function (called in line 31 and defined in line 60) uses `SetDeviceMetricsOverride()` to adjust the dimensions of the invisible browser. The `buf` image buffer returned by the `Screenshot` function in line 33 is written to disk in

Listing 1: screenshot.go

```
01 package main
02
03 import (
04   "context"
05   emu "github.com/chromedp/cdproto/emulation"
06   "github.com/chromedp/cdproto/page"
07   cdp "github.com/chromedp/chromedp"
08   "io/ioutil"
09 )
10
11 func main() {
12   ctx, cancel :=
13     cdp.NewContext(context.Background())
14   defer cancel()
15
16   var buf []byte
17   tasks := cdp.Tasks{
18     cdp.Navigate(
19       "http://linux-magazine.com"),
20     cdp.ActionFunc(
21       func(ctx context.Context) error {
22         _, _, contentSize, err :=
23           page.GetLayoutMetrics().Do(ctx)
24         if err != nil {
25           panic(err)
26         }
27
28         w, h := contentSize.Width,
29           contentSize.Height
30
31         viewPortFix(ctx, int64(w), int64(h))
32
33         buf, err = page.CaptureScreenshot().
34           WithQuality(90).
35           WithClip(&page.Viewport{
36             X:      contentSize.X,
37             Y:      contentSize.Y,
38             Width:  w,
39             Height: h,
40             Scale:  1,
41           }).Do(ctx)
42         if err != nil {
43           panic(err)
44         }
45         return nil
46       })}
47
48   err := cdp.Run(ctx, tasks)
49   if err != nil {
50     panic(err)
51   }
52
53   err = ioutil.WriteFile("screenshot.png",
54     buf, 0644)
55   if err != nil {
56     panic(err)
57   }
58 }
59
60 func viewPortFix(
61   ctx context.Context, w, h int64) {
62   err := emu.SetDeviceMetricsOverride(
63     w, h, 1, false).
64     WithScreenOrientation(
65       &emu.ScreenOrientation{
66         Type:
67         emu.OrientationTypePortraitPrimary,
68         Angle: 0,
69       }).
70     Do(ctx)
71
72   if err != nil {
73     panic(err)
74   }
75 }
```

Listing 2: foreground.go

```
01 package main
02
03 import (
04   "context"
05   cdp "github.com/chromedp/chromedp"
06   "time"
07 )
08
09 func main() {
10   pctx, pcancel := cdp.NewExecAllocator(
11     context.Background(), cdp.NoFirstRun)
12   ctx, cancel := cdp.NewContext(pctx)
13   defer cancel()
14   defer pcancel()
15
16   tasks := cdp.Tasks{
17     cdp.Navigate(
18       "https://linux-magazin.de"),
19     cdp.Navigate(
20       "http://linux-magazine.com"),
21     cdp.Sleep(5 * time.Second),
22   }
23
24   err := cdp.Run(ctx, tasks)
25   if err != nil {
26     panic(err)
27   }
28 }
```

PNG format by `WriteFile()`. The sequence of the actions, starting with navigating to the page, followed by taking the screenshot, is processed by the `Run()` function starting in line 48.

The technique of creating screenshots of automatically fetched web pages opens up a number of unheard-of possibilities when testing newly developed web UIs. For example, image recognition can later determine whether the site's various graphic elements are in the right place with different browser sizes, without human test personnel having to click their way through the flow with every release. It could also be used to implement a neat system for archiving websites; in the next century, historians would surely be amused by the advertisements placed on the *Linux Magazine* homepage in 2020.

Complicating Easy Things

For test purposes, it would be quite useful at times to start the remote browser visibly in the foreground instead of hidden in the background. Developers of scraping applications can thus determine if the browser is stepping through or if it gets stuck at some point. Paradoxically, however, setting up foreground mode has become quite complicated since the introduction of default background mode in chromedp some time ago, since using `NewContext()` to create a new browser context configures the browser to run in background mode deep down in the library's engine compartment, which is inaccessible from outside.

This is why Listing 2 creates a new browser controller in the form of `NewExecAllocator()` and passes it the `NoFirstRun` option to make the browser run in the foreground. Back comes a context, but, alas, not a context compatible with the context object that chromedp uses and gives to `Run()` in line 24 of the executing function. Therefore, line 12 creates a compatible context via `NewContext()` and passes it the previously created `Exec` context as a parent context. The new chromedp context also has a `cancel()` function, and the `defer` statements in lines 13 and 14 are both triggered at the end of the program to neatly collapse the remote-controlled browser.

Listing 2 only accesses the homepages of the German and English versions of *Linux Magazine* for this test; it then `Sleep()`s for five seconds and terminates.

Full-Text Search

How can the newly launched browser drone simulate user interactions such as mouse clicks or keyboard strokes to follow more complicated web flows? The *chromedp* library offers functions to select certain form fields or buttons from the displayed web page's DOM via an XPath query and communicates with them via `SendKeys()`, `Submit()`, or `Click()`.

For example, to fire off a full-text search of all repositories on GitHub, you first need to discover the name of the search field there. A quick look in the Chrome browser's developer view (via the *Inspect Elements* menu) reveals that the search field goes by the `name` attribute `q` (Figure 2). Line 15 from Listing 3 defines the associated XPath query `//input[@name="q"]` in the `sel` variable. After accessing the GitHub page, the `WaitVisible()` task in line 19 waits until the search field arrives via the Internet's series of tubes.

The `SendKeys()` function in line 20 then sends the search string (`waaah`) to the input field and terminates it with a newline character. This is enough to prompt the GitHub UI to initiate the search without having to press a *Submit* button. Line 21 then waits with `WaitReady("body", cdp.ByQuery)` until the result is available, before initiating the output of the returned raw HTML with a user-defined `ActionFunc()`.

With the content displayed in Figure 3, it then uses `dom.GetDocument()` to retrieve the root node of the HTML document and `dom.GetOuterHTML()` to get the raw HTML code on the page. For test purposes, the `Printf()` function outputs the displayed page's source code; a scraper application could extract interesting content from this and process it further.

Not Entirely Welcome

Chromedp supports all Chrome and Edge browsers, but not Firefox, which is

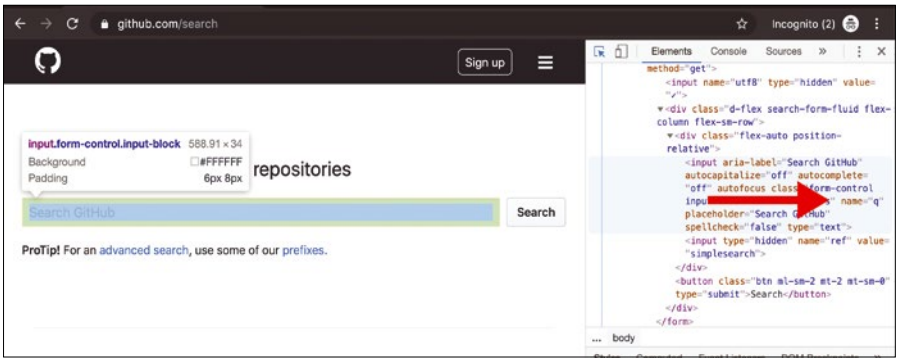


Figure 2: The query field in the GitHub full-text search across all repositories goes by the name of `q`.

Listing 3: github.go

```
01 package main
02
03 import (
04   "context"
05   "fmt"
06   "github.com/chromedp/cdproto/dom"
07   cdp "github.com/chromedp/chromedp"
08 )
09
10 func main() {
11   ctx, cancel :=
12     cdp.NewContext(context.Background())
13   defer cancel()
14
15   sel := `//input[@name="q"]`
16   tasks := cdp.Tasks{
17     cdp.Navigate(
18       "https://github.com/search"),
19     cdp.WaitVisible(sel),
20     cdp.SendKeys(sel, "waaah\n"),
21     cdp.WaitReady("body", cdp.ByQuery),
22     cdp.ActionFunc(
23       func(ctx context.Context) error {
24         node, err := dom.GetDocument().Do(ctx)
25         if err != nil {
26           panic(err)
27         }
28         res, err := dom.GetOuterHTML().
29           WithNodeID(node.NodeID).Do(ctx)
30         if err != nil {
31           panic(err)
32         }
33         fmt.Printf("html=%s\n", res)
34         return nil
35       }),
36   }
37   err := cdp.Run(ctx, tasks)
38   if err != nil {
39     panic(err)
40   }
41 }
```

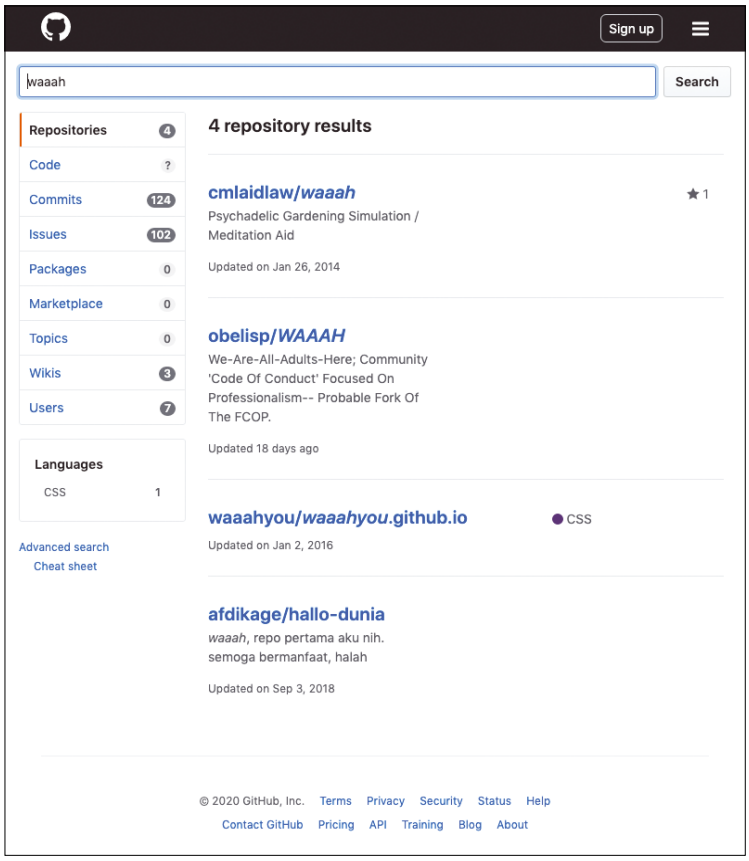


Figure 3: The headless Chrome browser entering a search query on GitHub.

based on a different technology. The chromedp project on GitHub is trying to keep up with new Chrome versions and changes to the DevTools protocol: Sometimes, things don't work as expected, and workarounds have to be found.

There are open issues on the GitHub projects, and the developers are addressing problems with new versions of the library. Besides Google's official documentation [2], there are some tutorials online, but not a thorough and practical publication on the topic. At least, I found a book [5] that contains a huge amount of useless filler material, but also covers various topics in the field of web scraping and briefly discusses Selenium and Chrome DevTools.

Popular websites like Facebook, Twitter, or Google also seem to be interested in making scraping with tools like chromedp as hard as possible. For example, Gmail uses dynamically generated random names for the input fields on the login page. In addition, ongoing changes to the page layout often require a scraper adjustment, so that the whole undertaking remains an arms race between the content provider and the scraper developers. The industry giants want to bind their users to the official pages in order to bombard them with advertising, because, let's face it, somebody has to pay the server bill at the end of the month. ■■■

Info

[1] "Programming Snapshot – Colly" by Mike Schilli, *Linux Magazine*, issue 223, June 2019, pp. 54-57

[2] Chrome DevTools: *https://developers.google.com/web/tools/chrome-devtools/*

[3] chromedp: *https://github.com/chromedp/chromedp*

[4] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/234/*

[5] Smith, Vincent. *Go Web Scraping Quick Start Guide*, Packt Publishing, 2019, *https://learning.oreilly.com/library/view/go-web-scraping/9781789615708/cover.xhtml*

Author

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

An alternative to systemd

Back to the Future

If you find systemd needlessly complicated, runit offers an easy-to-use, minimal init alternative. *By Bruce Byfield*

On Unix-like systems like Linux, init is the first process to run during bootup, and the one that controls all other processes. For years, most distributions have used SysVinit, an init inspired by the one used in Unix System V. However, in 2012-15, the majority of distributions switched to systemd, which not only provides init services, but an administrative overlay of the entire operating system. Since then, a variety of simpler inits have been written by those who find systemd needlessly complicated. One of the best-known of these alternative inits is Gerrit Pape's runit [1], a collection of utilities designed to be a minimal and easy-to-use init system.

Runit boots and shuts down a system in three stages. Like SysVinit, runit uses runlevels, but, by default, it only uses two: `default`, which runs all the services linked in `/var/service`, and `single`, which is used for rescue and recovery. Other runlevels can be added if desired [2].

Runit's structure offers several advantages:

- Navigation is easy, because components are organized by directories. For example, `/var/service` contains links to all active services, while `/etc/sv` contains active services, and `/etc/runit/runsvdir` contains all available runlevels.
- Each service is assigned a unique directory name, which has a symbolic link to `/var/service`. Each directory has a single file called `run`, which has a symbolic link to `/var/service`. For instance, Listing 1 shows the `run` file for `sshd` (borrowed from the runit website [3]). Notice that it is nothing more than a shell script for running the service. Running as a child of the `runsv` utility, each defined process is assumed to be always running, and, by default, it is restarted automatically if necessary. This arrangement makes services easy to locate and eliminates the need for commands like `killall` and `pidof`.
- Each service is started with a clean process. The service always has the same environment, resource limits, open file descriptors, and controlling terminals.
- Using `runsv`, runit can be configured to produce a reliable log.
- Running in parallel, runit boots and shuts down quickly.
- Preconfigured for Debian and BSD systems, runit is easy to port to other Unix-like systems.
- Runit's stage 1 and 3 are governed by shell scripts, making them easy to customize for different distributions.
- A small code size reduces the likelihood of errors and makes runit fast. Runit's utilities vary between 300-500 lines in length.

As a relatively new project, runit is not widely used, although it is carried by many major distributions. You cannot replace systemd with runit, but the online help gives detailed instructions about how to replace SysVinit and gives dozens of sample scripts for starting common services, from simple ones like `atd` to complex ones like Apache2. If you are simply curious about runit, you can run Void Linux [4] in a virtual machine or from a Live CD to explore it. Void is the only Linux distribution that uses runit by default; it is interesting in itself, because it is written largely from scratch for speed and efficiency. Runit is only one of its unusual features.

runit in Action

Runit is actually a collection of half a dozen utilities that work together [5]. The first process that the kernel starts is `runit-inited`. Once the operating system

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 co-founder of Prentice Pieces, a blog about writing and fantasy at *https://prenticepieces.com/*.

Lead Image © Sergey Nivens, 123RF.com

Listing 1: run File for sshd

```
01 #!/bin/sh
02 ssh-keygen -A >/dev/null 2>&1
   # Will generate host keys if they don't already exist
03 [ -r conf ] && . ./conf
04 exec /usr/sbin/sshd -De $OPTS 2>&1
```

```
# sv -v up /etc/sv/sshd
ok: run: /etc/sv/sshd: (pid 1723) 63s
# sv -v down /etc/sv/sshd
ok: down: /etc/sv/sshd: 0s, normally up
# sv -v restart /etc/sv/sshd
ok: run: /etc/sv/sshd: (pid 1772) 0s
```

Figure 1: The basic commands for starting and stopping services.

```
# sv status /var/service/*
run: /var/service/NetworkManager: (pid 958) 241s
run: /var/service/acpid: (pid 959) 241s
run: /var/service/agetty-hvc0: (pid 1382) 1s
run: /var/service/agetty-hvsi0: (pid 1381) 1s
run: /var/service/agetty-tty1: (pid 947) 241s
run: /var/service/agetty-tty2: (pid 944) 241s
run: /var/service/agetty-tty3: (pid 943) 241s
run: /var/service/agetty-tty4: (pid 950) 241s
run: /var/service/agetty-tty5: (pid 948) 241s
run: /var/service/agetty-tty6: (pid 951) 241s
run: /var/service/dbus: (pid 960) 241s
run: /var/service/elogind: (pid 954) 241s
run: /var/service/lxdm: (pid 1034) 234s
run: /var/service/polkitd: (pid 949) 241s
run: /var/service/rtkit: (pid 991) 240s
run: /var/service/sshd: (pid 953) 241s
run: /var/service/udevd: (pid 955) 241s
run: /var/service/uuidd: (pid 956) 241s
```

Figure 2: The list of running services in Void Linux.

is ready for use, `runit-init` replaces itself with `runit-inited`. You can use the command `init 0` to shutdown the system and `init 6` to reboot the system.

Probably, the most important utility is `sv`. `sv` follows the classic Unix nomenclature:

```
sv up /etc/sv/SERVICE
```

starts a service, while

```
sv down /etc/sv/SERVICE
```

stops it. To restart a service (Figure 1), use

```
sv restart /etc/sv/SERVICE
```

The command

```
sv status SERVICE
```

shows its current status, while

```
sv status/var/service/*
```

returns the status of all services (Figure 2). No confirmation message is given unless the `-v` option is used, and even then it is the briefest of status reports.

In addition, `sv` can be used to add new services. After setting up the service and its run file, add a symbolic link with:

```
ln -s /etc/sv/SERVICE /var/service/
```

```
# touch /etc/sv/sshd down
# cd /etc/sv/sshd
# ls
run  supervise
# cd /etc/sv/sshd
# ls
run  supervise
# ls
control  lock  ok  pid  stat  status
```

Figure 3: You can create a file in the service's directory that sets it to not run automatically.

To prevent a service from starting automatically, create a file in the service's directory with:

```
touch /etc/sv/SERVICE/down
```

Rather than creating a file named `down`, as you might expect, the command signals runit to create a `supervise` subdirectory, a collection of binary files that define how the service runs (Figure 3). If you no longer want to prevent the service from starting automatically, delete the `supervise` directory.

As you can see, `sv` works with already existing Linux commands, making the command easier to learn.

`svlogd` is used to set up an optional log in a preexisting directory that rotates when the current log reaches a certain size. Its syntax is:

```
svlogd OPTIONS LOG-PATH
```

`svlogd`'s options include a timestamp (`-t`), a human readable timestamp in *YYYY-MM-DD_HH:MM:SS.xxxxx* format (`-tt`), and a line length for entries (`-l NUMBERS`) (Figure 4).

A log can be configured by placing a config file in the log directory. Table 1 shows some of the most important configuration options.

When necessary, you can monitor a specific service (`s`) with `runsvdir`. The structure of the command is:

```
runsvdir /etc/sv/SERVICE LOG-PATH
```

Up to 1,000 services can be monitored. Every five seconds, `runsvdir` checks whether the time of the last modification, the inode, or the device of a listed service has changed. If so, `runsvdir` rescans, and, if a service is stopped, stops monitoring it. Similarly, if the service has been restarted, then a new process is started. In addition, all activity in runit-related directories is recorded. `runsvdir` is not needed for runit to function so much as it is an administrative tool to help keep track of selected services. Closely related to `runsvdir` are `runsvchdir` (`runsvchdir DIRECTORY`), which changes the logfile for `runsvdir`, and `runsv` (`runsv SERVICE`). `runsv` starts, stops, or pauses the specified service and can interrupt a service or send an alarm if the service is running.

Runit's last utilities are `chpset`, which can run a specialized version of a service for one user or group, and `utmpset`, which modifies `utmp` and/or `wtmp` when a user logs out. However, neither `chpset` or `utmpset` are essential to runit for many users. Like other init systems, once set up, runit operates largely in the background and should rarely require such specialist tools on a small network or a lone workstation.

```
# mkdir /etc/runit-log
# svlogd -ttv /etc/runit-log
svlogd: info: new: /etc/runit-log/current
cd /etc/runit-log
ls
current  lock
# less current
# cat current
2020-03-02_20:14:34.57773 cd /etc/runit-log
2020-03-02_20:14:37.11941 ls
```

Figure 4: The steps in the creation of a log directory. The log records all runit-related activity, including actions taken in any runit directory.

Table 1: svlogd Options

size	The maximum number of bytes in the current log. The default is 1,000,000. Set the size to zero to turn off log rotation.
nnum	The number of old logfiles to keep. The default is 10, and zero prevents old logs from being removed. When the number of logs exceeds this number, the oldest log is deleted.
Nmin	Sets the minimum number of old logfiles to keep.
ttimeout	The maximum age of the current logfile. When this age is exceeded, the current log is rotated.
ua.b.c.d[:port]	Transmits log messages through the IP address *a.b.c.d*, port number *port*. The default port is *540*. Messages are still written to the local file.
Ua.b.c.d[:port]	Transmits log messages through the IP address *a.b.c.d*, port number *port*. The default port is *540*. Messages are *not* written to the local file.

Reconsidering init Choices

Runit is notably smaller and more limited in scope than the omnipresent systemd. If you hold to the Unix philosophy that an app should do one thing and do it well, you are likely to warm to runit to a degree that you have not to systemd.

However, that is not runit's only virtue. Runit is economically and consistently written and easy to learn. After exploring runit for several hours, I felt competent to use it in a way I have yet to feel with systemd despite several years of use. Although runit might seem a return to a simpler age before systemd, I am nearly convinced that it would not be a bad thing.

Support for multiple init systems would be difficult for any distribution, and perhaps impractical. For this reason, it may be a while before runit is an alternative in the current systemd-dominated world. Still, if runit was compatible with KDE's Plasma and practical to set up in Debian, I would be seriously tempted to use it on my main workstation. ■■■

Info

[1] runit: *http://smarden.org/runit/*
[2] Runlevels: *http://smarden.org/runit/runlevels.html*
[3] Runlevels: *http://smarden.org/runit/runscripts.html*
[4] Void Linux: *https://voidlinux.org/*
[5] runit utilities: *http://smarden.org/runit/*

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

MakerSpace

CircuitPython for Raspberry Pi and MCUs

One for All

The CircuitPython run-time environment runs on almost all microcomputers and microcontrollers, making it perfect for cross-platform programming. *By Bernhard Bablok*

CircuitPython is an Adafruit fork for MicroPython. Both are run-time environments for microcontrollers. Thanks to a compatibility layer, CircuitPython now also runs on various single-board computers (SBCs) like the Raspberry Pi.

MCU Scripting Languages

Traditionally, microcontroller units (MCUs), because of their extremely limited resources, where every byte can count, run either Assembler or C/C++. programs. Over the years, however, manufacturers have been improving their products, and powerful SBCs are now available for very little money that have sufficient reserves for interpreted languages.

Moreover, many typical applications do not exhaust an MCU (e.g., in education or home projects). The obvious approach then is to use an easier to learn scripting language instead of C/C++. Luckily, some processor families have Python, Lua, or JavaScript support (Figure 1).

Even small SBCs like Pi Zero have enough RAM and storage available to use scripting languages without problem. Python in particular impresses with innumerable ready-to-use modules for every imaginable purpose.

Like many other distributions, Raspbian is currently in the final transition phase from Python 2 to Python 3. The developers will soon stop maintaining version 2 because parallel support of two Python variants takes up too much time, raising the question as to why another Python dialect seems necessary or useful.

On the Raspberry Pi and other SBCs, CircuitPython does not use its own interpreter but requires Python 3. CircuitPython is not only an interpreter

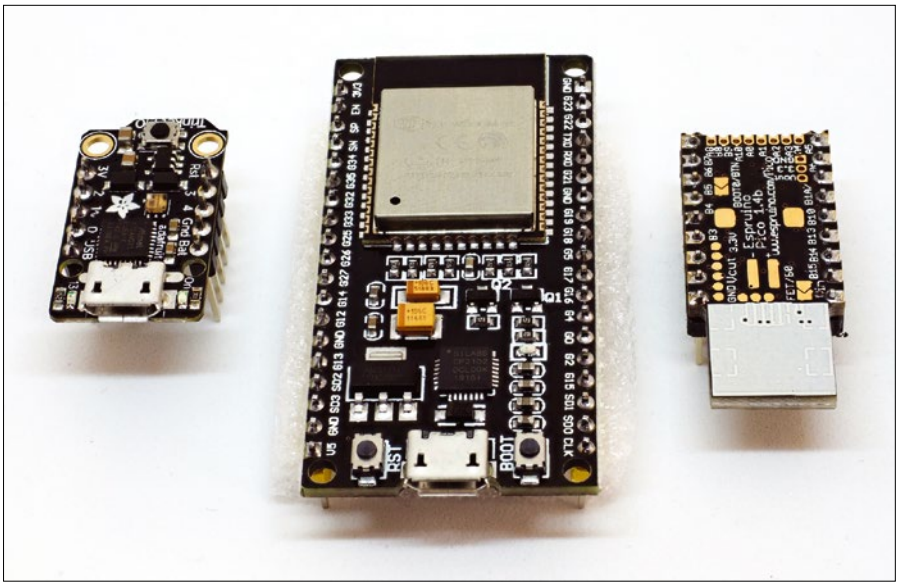


Figure 1: Various MCUs with support for scripting languages. From left to right: Trinket M0, ESP32, and Espruino Pico.

Listing 1: Installing Blinka Library

```
#!/bin/bash

apt-get update
apt-get -y install python3-pip
pip3 install RPI.GPIO adafruit-blinka

# Activate SPI and I2C (as required)
echo "dtparam=spi=on"      >> /boot/config.txt
echo "dtparam=i2c_arm=on" >> /boot/config.txt
echo "i2c-dev"             >> /etc/modules

# Install application libraries
pip3 install adafruit-circuitpython-bme280
```

Lead Image © donatas1205, 123rf.com

(on MCUs), but also a hardware abstraction layer in the form of a collection of system libraries. Although the Raspberry Pi already has available various libraries that abstract the hardware, they were designed to be Pi-centric so that the corresponding programs run only on Raspberry Pi without modifications. CircuitPython, on the other hand, promises compatibility across hardware boundaries.

Programming Models

The programming model of any microcomputer, no matter how minimal, is fundamentally different from that of a microcontroller. The computer runs an operating system that, simply put, manages the hardware, programs, and users.

The operating system allocates the available CPU(s) in sequence to all active programs. In this way, several programs run virtually in parallel, even if only one processor is available. However, only on real-time operating systems can a program rely on getting computing time reliably within a defined period. On conventional operating systems it simply has to wait its turn.

An MCU, on the other hand, has no operating system; the single-application program also controls all the resources, which makes it easy to implement time-critical applications that need to transmit bits across the wire within microseconds. Typical application programs here consist of three parts. The first initializes the MCU hardware, the second is an infinite loop, and the third deals with interrupts (i.e., interruptions of the normal process, for example by incoming data).

Apart from reliably allocating computing time and other resources, this programming model also exists for many Raspbian programs. Sensors are read and data is written in a regular cycle, which means it is also possible to use CircuitPython with its simpler programming model for these cases.

Blinka

If you want to prepare your Raspberry Pi for CircuitPython, you need the *Blinka* library, named after CircuitPython's mascot. Its installation, including all dependencies, is handled by the script in Listing 1. Raspbian also has many standard Python packages in the normal repository. However, the script uses the cross-platform Python package installer `pip3`. This tool downloads the packages, builds them from the sources, if required, and stores them in the `/usr/lib/python3.6/dist-packages/` directory.

The script also activates the I2C and SPI interfaces immediately, because reading sensors is one of the tailor-made applications for CircuitPython. After a reboot, the interfaces are then available for your applications. If you have already enabled them on the system with `raspi-config`, then comment out the lines under the *Activate SPI and I2C (as required)* comment in Listing 1. You can see from the installation script itself that Blinka uses RPI.GPIO internally – but this only applies to the Raspberry Pi.

Listing 2: Data Collection

```
01 #!/usr/bin/python3
02
03 import time
04 import board
05 import busio
06 import digitalio
07 import adafruit_bme280
08 import hd44780
09
10 # Create objects
11 i2c     = busio.I2C(board.SCL, board.SDA)
12 bme280  = adafruit_bme280.Adafruit_BME280_I2C(i2c,address=0x76)
13 display = hd44780.HD44780(i2c,trans_map={"°": 223})
14
15 btn           = digitalio.DigitalInOut(board.D4)
16 btn.direction = digitalio.Direction.INPUT
17 btn.pull      = digitalio.Pull.UP
18
19 # Convert air pressure to sea level for a given altitude
20 sealevel = 520
21 fac      = pow(1.0-sealevel/44330.0, 5.255)
22
23 # Main loop
24 bl_on = True
25 while True:
26   try:
27     if bl_on:
28       line1 = "T: %3.1f°C %2d%%" % (bme280.temperature,bme280.humidity)
29       line2 = "P: %4d hPa" % int(bme280.pressure/fac)
30       display.write(line1,1)
31       display.write(line2,2)
32
33     # Query button
34     if not btn.value:
35       bl_on = not bl_on
36       display.backlight(bl_on)
37       time.sleep(0.5)
38
39     time.sleep(0.1)
40   except Exception as ex:
41     print("ex: %r" % ex)
42     break
43
44 display.clear()
45 display.backlight(False)
```

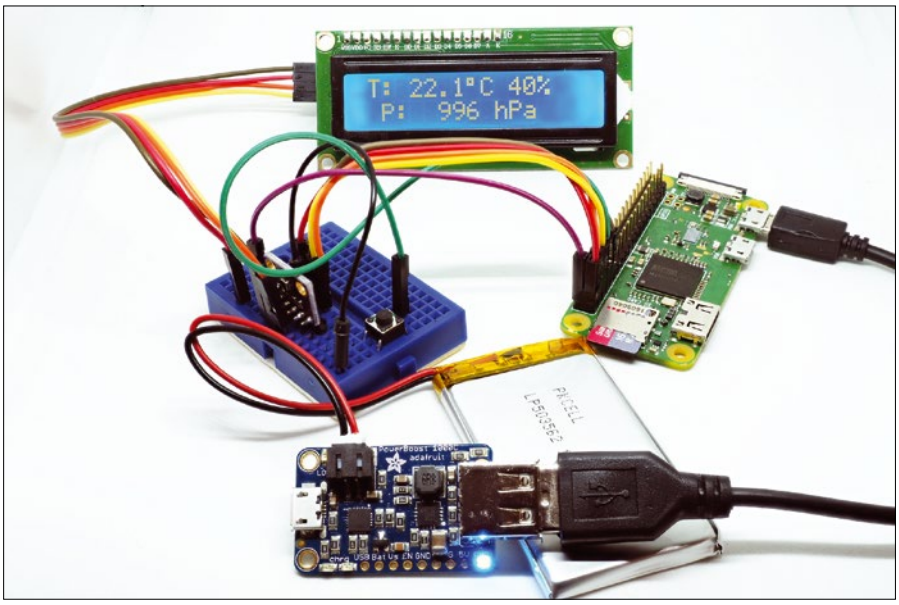


Figure 2: The test setup with a sensor and a display.

Depending on your application, you will need several libraries besides the Blinka layer. The example program presented here is intended to read the BME280 [1] temperature, humidity, and pressure sensor in an infinite loop and output the results on an old-fashioned display with two lines of 16 characters each. For this sensor, the last line of Listing 1 installs another CircuitPython library from Adafruit.

Although Adafruit has published many libraries, they do not cover every electronic component. The cheap LCDs with two to four display lines – you will find them on Ebay or Amazon – often work with the PCF8574T serial converter chip. However, Adafruit uses the MCP230xx for its displays. For this reason, the Adafruit LCD library is not used for the application example, but rather the library from my GitHub site [2]. To proceed, manually download the Python file `lib/hd44780.py` in the repository and copy it into the directory of your main program.

On MCUs, the process is slightly different because `pip` is missing. Adafruit provides library packages in ZIP format for each platform. After downloading, unpack the archive and copy the relevant libraries to the `lib/` subdirectory.

Data Collector

The data collection program (Listing 2) addresses the BME280 over the I2C interface of the Raspberry Pi. SDA is located on header pin 3 and SCL on pin 5, which only plays a role for the wiring; the program itself uses the platform-specific *board* package (line 11), which makes porting to other platforms easier. With `board.<xxxx>` you can access a whole range of symbolic constants for various hardware features. To reach this data, `busio.I2C()` creates an interface to access the standard I2C device and `busio.SPI()` the SPI interface.

In addition to the data lines, the sensor in this example requires 3.3V and a ground connection (although some sensors of this type are 5V; check your sensor specs). Thanks to the serial adapter, the LC display is controlled by the same data lines – but the display requires 5V operating voltage. Careful users will want to solder out the pullups of the PCF8574T, which boost the I2C bus to 5V. Because the Raspberry Pi has internal pullups on SDA and SCL, they are not necessary (Figure 2).

A breadboard allows both components to be connected simultaneously to the normal I2C connector of the Raspberry Pi. Both components usually use the addresses 0x27 and 0x76 on the I2C bus. Depending on the module, however, the addresses may differ. The `i2cdetect` program from the *i2c-tools* library provides this information (Figure 3).

The program in Listing 2 creates the Python objects for the sensor, the display, and a button in lines 11 to 17. The sensor delivers the temperature, humidity, and either the air pressure or the altitude. The last two parameters depend on each other: If you specify the altitude, the sensor outputs the air pressure. If you specify the air pressure, the construct calculates the altitude. For a stationary system, the first option is more useful.

After initializing the components, the program runs in an infinite loop starting in line 25. The measurement is taken automatically by accessing attributes of the BME280 object. Lines 30 and 31 format the output to fit on the display being used, which has two lines of 16 characters each. At this point, the data could still be stored in a CSV file for later processing in LibreOffice Calc or for saving in a database. However, if you are looking for maximum compatibility, you will want to avoid this path because the program will then no longer run on a microcontroller unless modified.

Restrictions

For didactic reasons, the program from Listing 2 uses an optional button that switches the backlight of the display on or off. The button is connected to GPIO4, and the program accesses it

```
[root@pi0w:~] # i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- 27 -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- 76 --
```

Figure 3: The output from `i2cdetect` shows the two components at the addresses 0x27 and 0x76.

with `board.D4` (line 15). The query is quite simple with the `value` attribute (line 34).

The button, however, also shows what is currently CircuitPython's biggest limitation. The run-time environment does not support interrupts or any kind of asynchronous code processing. Therefore, the application program needs to constantly query the button. The more switches it has to manage, the higher the chance of missing a button. This approach unnecessarily inflates the rest of the application logic, as well. The sample program does not care and reads the sensor 10 times per second, which is not always possible or useful.

If you want to query the sensor at fixed intervals, you have to save the last reading time and regularly check whether enough time has elapsed. Because CircuitPython does not have its own interpreter on the Raspberry Pi, on this platform you could simply choose to ignore the limitation and work with threads; however, then the programs cannot be ported without customization.

Another restriction concerns the available libraries. The SAMD21/SAMD51 MCUs now have the *displayio* library for output on (small) displays. This library lets you display images, text, and graphics comfortably. The library has not yet been ported for the Raspberry Pi. The only option is to use libraries such as *Pillow* from the normal Python collection. Also, the UART interface works a bit differently with Raspberry Pi and uses the normal Python serial module.

Porting

Porting existing libraries to CircuitPython is comparatively simple. I created a MicroPython library for the small DFPlayer Mini MP3 Player chip and the LCDs used in the example. In the latter case, I left out almost half of the code lines, because the I2C abstraction of the CircuitPython libraries replaced the equivalent DIY functions.

The situation is different when porting Blinka to other platforms. Adafruit has a manual for installing on the Orange Pi R1, an even more compact SBC than the Pi Zero. The module is based on the Allwinner H2 chip and, thanks to its four CPU cores, is even better than the Pi Zero for many CPU-intensive tasks. The biggest disadvantage of these still fairly exotic SBCs is the lack of support for the various hardware add-ons available for the Raspberry Pi family. The future looks far rosier for Blinka. At least all boards that use standard interfaces like I2C, SPI, and UART are immediately ready for use with an available Blinka port.

A test port to the Orange Pi One Plus SBC, which uses the Allwinner H6, turned out to be more complex than expected. Additionally, a lack of documentation from the SBC manufacturer further complicates the work. To make things worse, Adafruit also does not have a porting guide, which adds complications.

Conclusions

Genuine cross-platform programming still does not work perfectly – even with CircuitPython, partly because of the lack of ports for individual libraries and partly because every platform works differently when you get into the details. However, porting overhead is reduced in all cases. If you look at the example in this article, you only need to change the way you define the button, and then the program will also run on the Trinket M0.

The biggest advantage of CircuitPython for the Raspberry Pi is easy access to many hardware components, regardless of whether you adopt the simpler programming model. Prototyping for MCUs is also often easier on the Raspberry Pi than directly on the corresponding hardware. Porting existing Python libraries is also far easier, because CircuitPython provides a stable API for accessing the classic I2C and SPI bus systems.

Programming fun is also crucial, though, and it is here that CircuitPython makes sure to lower the obstacles to getting started on a growing number of platforms. ■■■

Info

[1] BME280: *https://www.amazon.com/ACROBOTIC-Temperature-Barometric-Raspberry-Altimeter/dp/B07HVFB9M9/ref=sr_1_29*

[2] HD44780 library for CircuitPython: *https://github.com/bablokb/circuitpython-hd44780*

Author

Bernhard Bablok works at Allianz Technology SE as an SAP HR developer. When he's not listening to music, cycling or walking, he deals with topics related to Linux, programming and small computers. He can be reached at *mail@bablokb.de*.

shop.linuxnewmedia.com

REAL SOLUTIONS *for* REAL NETWORKS

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

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

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

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

Improve your admin skills with practical articles on:

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

SUBSCRIBE NOW
SAVE 30%

ADMIN
Network & Security

shop.linuxnewmedia.com

MakerSpace

Pi Zero as a universal USB stick

Multitool

In just a few simple steps, you can turn a Pi Zero into a universal USB flash drive that emulates storage, a serial port, Ethernet, and more. *By Bernhard Bablok*

Instead of taking along a separate USB gadget for every task, you can turn a Pi Zero into a universal device that provides storage, a network interface, and additional functions.

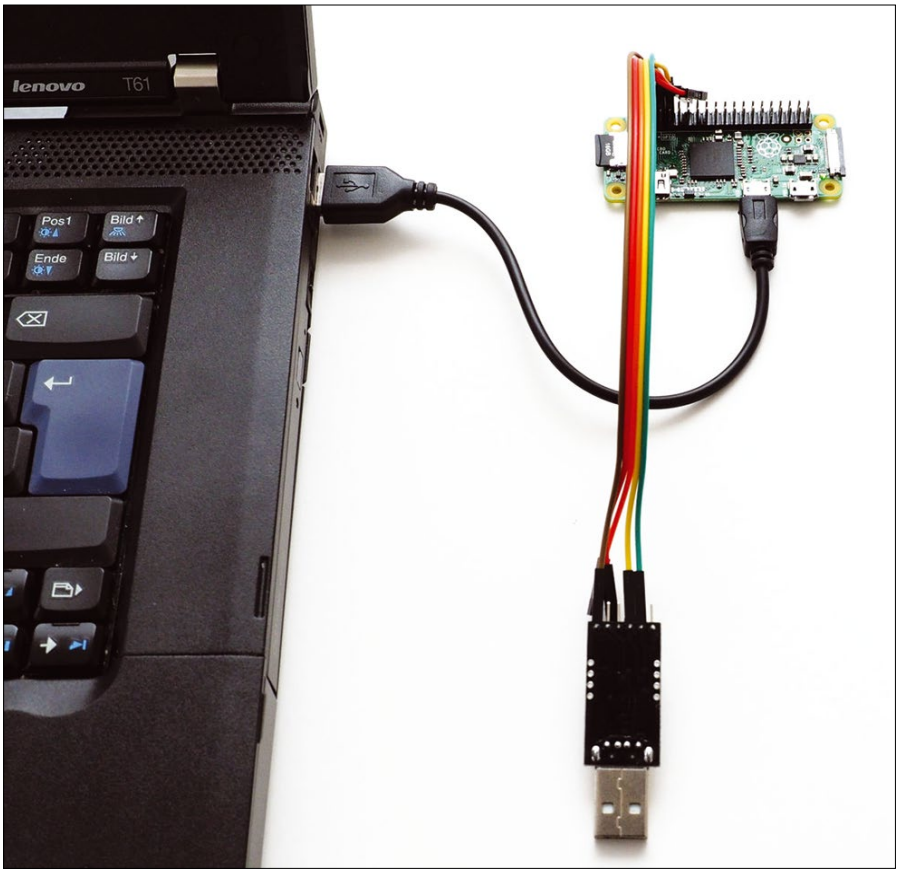


Figure 1: If you connect the Pi Zero to a laptop with a normal cable, the SBC receives power and data.

This project relies on a rarely used feature of the Pi Zero: Its USB port supports USB On-The-Go (OTG). If you connect the small-board computer (SBC) to another computer, the Raspberry Pi logs in as a lower-level device (gadget).

Normally gadget mode [1] is undesirable. If you want to connect a memory stick to the Pi Zero, you need a special OTG cable; otherwise, it will not work (see the "How OTG Works" box). Conversely, plugging a normal cable into the USB port triggers the desired behavior.

In Figure 1, the USB cable is plugged into the left-hand USB port and is used both to supply power and exchange data. If you want to reproduce the project without much effort, just from the software point of view, you can skip the next steps.

However, the cable solution is not elegant when you're on the road. On a laptop, especially, the Pi Zero will dangle in the air. For this reason, you need to do some tinkering with the hardware before you tackle the software.

Hands On

If you have basic experience with a soldering iron, attach a USB connector to the bottom of the Pi Zero (Figure 2), making sure the pins on the connector are assigned correctly; if in doubt, consult diagrams found on the Internet.

Lead Image © besjunior; 123rf.com

The most difficult part might not be the soldering itself, but finding the optimal length for the wires. USB plugs for self-soldering are available for a few cents from the usual mail order companies. Because the project basically only needs a Pi Zero (< $10/CA$13/£10/AU$25/EUR6), the risk of financial loss is not very high. If you have a Zero W, everything works the same way.

Another alternatives does not require any soldering. From Geekworm [2], you can pick up a ready-to-use board (Figure 3) for about $14 (CA$19/£11/AU$22/EUR13). Here, pogo (spring-loaded) pins ensure contact with the solder joints from below.

The solution is elegant and very compact but has the twin disadvantages that the Pi Zero remains unprotected and the USB plug is exposed. The big advantage is that the conversion from a Pi Zero to a USB stick is non-destructive.

The second alternative is the Zero Stem for about $6 (CA$8/£5/AU$11/EUR6), which is sold by various vendors worldwide [3] (Figure 4). The small board requires some soldering, but the USB connector is then firmly attached to the SBC, thanks to the screw connection.

Makers have 3D-printed housings for both the Geekworm board [4] and the Zero Stem [5]. When traveling, it certainly makes sense to swap protection for compactness. However, these housings do not solve one problem: If a stick juts out of the laptop by more than a few centimeters, jostling by people in public spaces will tear it off – it's not a matter of if, but when. That's why I decided to go for the DIY soldering approach and designed a suitable housing with an angled connector (Figure 5).

If you want to 3D-print the case, you will find it on Tinkercad [6]. The holder for the connector is optimized for the components I used. The very tight tolerances might have to be adapted to the specific printer or material used.

How OTG Works

Normally, the role of a device in communication between two connected partners is defined. It either acts as a host or it doesn't. The host supplies power to the other device (slave, gadget, or peripheral) and controls its functions. In this way, accessories such as sticks, mice, or keyboards require minimal electronics.

In the first smartphones, the roles were still clearly defined. A smartphone connected to a PC advertised itself as a mass storage device, and the PC was able to read and write files. Later, the manufacturers decided that it would be practical to plug a USB stick into the smartphone and write to the stick from the device. To do this, however, the smartphone had to act as the host and also be intelligent enough to switch between roles, depending on the context. This was the impetus for the OTG extension of the USB specification.

The technical implementation was simple: In addition to the four existing pins at USB 2 (5V, ground, two data pins), a fifth ID pin was added. If it is connected to ground, a smartphone or a Pi Zero will switch to the active host role; otherwise (pin *high* or *floating*), it acts as a slave.

Normal USB cables simply lack the appropriate coding, so devices connected to them assume their default roles: For the Pi Zero, the default means the typically unused slave role. OTG adapters and cables, on the other hand, only connect the pin to ground on one side and thus also define the roles.

Raspbian Tweaks

A normal Raspbian is not configured to be used as a peripheral, but it has all the

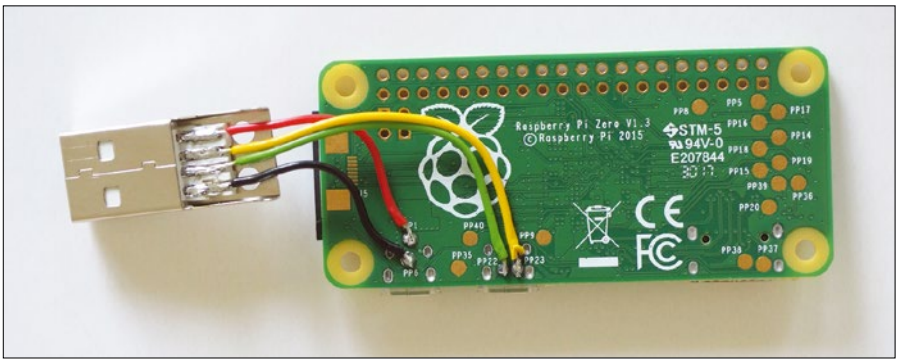


Figure 2: With a self-soldered USB connector, figuring out the right length for the wires might be more difficult than the soldering.

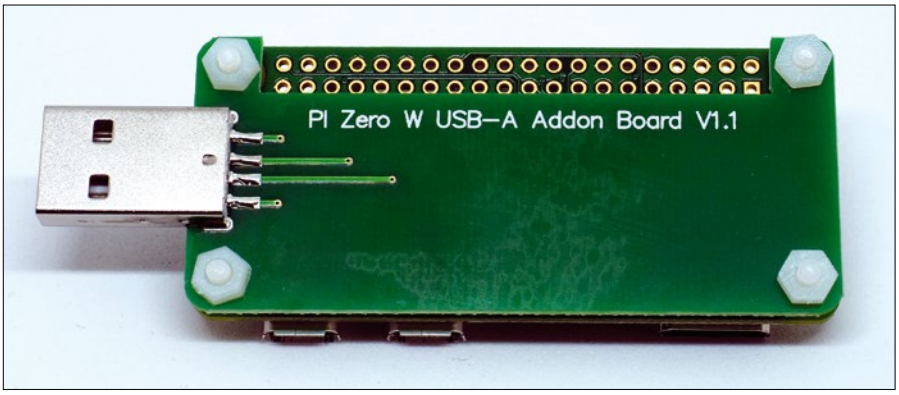


Figure 3: With a USB board from Geekworm, you can convert the Pi Zero without having to resort to your trusty soldering iron.

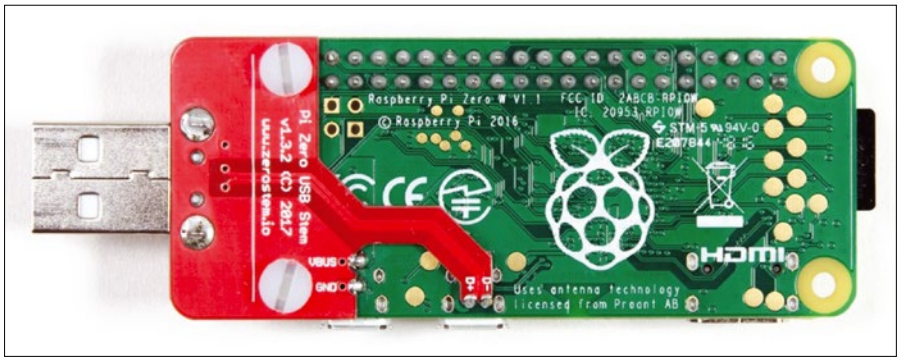


Figure 4: Although the Zero Stem requires the use of a soldering iron, it offers stable and clear connections (source: Pimoroni).

necessary features. Even the tasks you have to complete for this project do not require any special knowledge.

The Linux kernel has now reached the second iteration of gadget support. Originally, it had specialized drivers for this purpose, such as `g_ether` for emulating an Ethernet adapter. The configuration was very simple: The line

```
dtoverlay=dwc2
```

was added to the `/boot/config.txt` file, and you had to add the line

```
modules-load=dwc2,g_ether
```

to the `/boot/cmdline.txt` file, which is normally a one-liner.

Although this method still works, these drivers will eventually be removed from the kernel. Another disadvantage is that a simultaneous configuration with several device types (e.g., Ethernet and mass storage) never worked properly.

In the next sections, therefore, I describe the currently applicable configuration procedure. Although the `dtoverlay` line in `/boot/config.txt` remains, the Raspberry Pi can provide all the necessary information dynamically.

The procedure now is: (1) plug the Pi Zero into another computer to supply power and boot; (2) during the boot process, a script configures the computer as a gadget; (3) the computer then transmits all the appropriate information to its counterpart.

USB Configurations

When booting, a Linux system automatically creates an entry in the `/sys/bus/usb/` virtual filesystem (Figure 6) for all connected USB devices, including the internal hubs. Among other things, the content of all files with *bInterface* in their names is important: They describe the kind of USB device you are using.

To structure this information, the Linux kernel (like its Windows counterpart) contains an internal table with corresponding information. When a device is plugged in, the kernel and device exchange information, and the Linux kernel automatically creates the sysfs files.

However, because the Pi Zero is now assuming the role of the peripheral device, the process is reversed. The Linux kernel now provides the information to the other side with the configfs virtual filesystem (`/config`). Like sysfs, configfs only exists at run time, with very similar content. Creating the file is not difficult, but it is tedious: You have to create various files, directories, and links – which looks like the kind of task ripe for scripting.

Listing 1 shows excerpts from such a script (complete file are available from the GitHub project [7] or FTP [8]) and dynamically creates the required entries in configfs at boot time. The files differ depending on the role you assign to the stick. Because the USB specification allows a stick to provide multiple logical devices, you do not need to decide.

The script uses the `init_start()` function (lines 23-46) to create the basic files, including things like information about the vendor and type and maximum power consumption of the gadget. The `create_storage()` and `create_serial()` functions, on the other hand, create specific files, directories, and links for a mass storage device and the serial interface.

You install the script, including the appropriate system service, from the GitHub project. The system service is started at boot time and applies the configuration once. The example is based on a current Raspbian Lite; the Pixel version also works, but the Pi Zero has problems with the graphical interface.

The commands

```
$ git clone https://github.com/➥
  bablokb/raspi2go.git
$ cd raspi2go
$ sudo tools/install
```

install the software and adjust `/boot/config.txt`. Afterward, you need to configure various settings in the file `/etc/raspi2go.conf` (e.g., which role the Pi assumes or what name the "product" goes by). Rebooting enables the settings.

For the installation itself, as well as for later updates, you connect the Pi Zero to the local network in the usual way with an OTG adapter or access it over a wireless network. As long as you avoid using the soldered USB port, everything stays the same. Once the service is installed, the Pi Zero will display some error messages in the system log in host mode, but you can safely ignore them.

Figure 5: A USB gadget housing with an angled connector ensures that the Pi Zero does not protrude too far beyond the host after plugging in.

In gadget mode, the Pi Zero only suffers one problem: Simply unplugging it does not shut the Pi down properly. The solution is to run the Pi Zero in read-only mode. You can find detailed instructions on how to do this online [9].

Connected

The default settings in `/etc/raspi2go.conf` let the Pi Zero assume three roles: It advertises itself as a USB drive, USB Ethernet adapter, and serial USB adapter. Depending on the host settings, a Windows or Linux system then mounts the drive automatically.

It wouldn't make much sense to go through all this trouble for a normal USB stick, but functioning as a USB

```
[bablokb@pi0w:~] > ls -l /sys/bus/usb/devices/usb1/1-0\:1.0/
total 0
-rw-r--r-- 1 root root 4096 Oct  5 11:07 authorized
-r--r--r-- 1 root root 4096 Oct  5 11:07 bAlternateSetting
-r--r--r-- 1 root root 4096 Oct  5 11:07 bInterfaceClass
-r--r--r-- 1 root root 4096 Oct  5 11:07 bInterfaceNumber
-r--r--r-- 1 root root 4096 Oct  5 11:07 bInterfaceProtocol
-r--r--r-- 1 root root 4096 Oct  5 11:07 bInterfaceSubClass
-r--r--r-- 1 root root 4096 Oct  5 11:07 bNumEndpoints
lrwxrwxrwx 1 root root    0 Oct  5 11:07 driver -> ../../../../../../bus/usb/dr
vers/hub
drwxr-xr-x 3 root root    0 Oct  5 11:07 ep_81
-r--r--r-- 1 root root 4096 Oct  5 11:07 modalias
lrwxrwxrwx 1 root root    0 Oct  5 11:07 of_node -> ../../../../../../firmware/
evicetree/base/soc/usb@7e980000
drwxr-xr-x 2 root root    0 Oct  5 11:07 power
lrwxrwxrwx 1 root root    0 Oct  5 11:07 subsystem -> ../../../../../../bus/usb
-r--r--r-- 1 root root 4096 Oct  5 11:07 supports_autosuspend
-rw-r--r-- 1 root root 4096 Oct  5 11:07 uevent
drwxr-xr-x 3 root root    0 Oct  5 11:07 usb1-port1
```

Figure 6: Files on the sysfs virtual filesystem (`/sys`) tell you about the computer's USB devices.

Listing 1: raspi2go.sh (Excerpts)

```
015 USBCONF_DIR="/sys/kernel/config/usb_gadget/g1"
016 NODE="USB0"
017 C=1
018
021 init_start() {
022
023   echo 0x1d6b > idVendor  # Linux Foundation
024   echo 0x0104 > idProduct # Multifunction Composite Gadget
025   echo 0x0100 > bcdDevice # v1.0.0
026   echo 0x0200 > bcdUSB    # USB2
027   echo 0xEF   > bDeviceClass
028   echo 0x02   > bDeviceSubClass
029   echo 0x01   > bDeviceProtocol
030
031   # OS descriptors
032   mkdir -p os_desc
033   echo 1       > os_desc/use
034   echo 0xcd    > os_desc/b_vendor_code
035   echo MSFT100 > os_desc/qw_sign
036
037   mkdir -p strings/"$LANG_ID"
038   echo "$SERIAL_NO" > strings/"$LANG_ID"/serialnumber
039   echo "$MANUFACTURER" > strings/"$LANG_ID"/manufacturer
040   echo "$PRODUCT" > strings/"$LANG_ID"/product
041
042   mkdir -p "configs/c.$C/strings/$LANG_ID"
043   echo "$DESCRIPTION" > "configs/c.$C/strings/$LANG_ID/
                           configuration"
044   ln -s "configs/c.$C" os_desc
045   echo 250 > "configs/c.$C/MaxPower"
046 }
047 [...]
056 create_storage() {
065   [...]
066   # configure gadget
067   mkdir -p "functions/mass_storage.$NODE"
068   echo 1 > "functions/mass_storage.$NODE/stall"
069   echo 0 > "functions/mass_storage.$NODE/lun.0/cdrom"
070   echo 0 > "functions/mass_storage.$NODE/lun.0/ro"
071   echo 0 > "functions/mass_storage.$NODE/lun.0/nofua"
072   echo "$USB_FILE" > "functions/mass_storage.$NODE/
                        lun.0/file"
073
074   ln -s "functions/mass_storage.$NODE" "configs/c.$C/"
075 }
076 [...]
079 create_serial() {
080   mkdir -p "functions/acm.$NODE"
081   ln -s "functions/acm.$NODE" "configs/c.$C/"
082 }
083 [...]
108 # ---- main program   ---------------------------------
109
110 # source configuration file
111 . /etc/raspi2go.conf
112
113 # load modules
114 modprobe libcomposite
115 sleep 1
116
117 # initialize gadget
118 mkdir -p "$USBCONF_DIR"
119 cd "$USBCONF_DIR"
120 init_start
121
122 # create configuration
123 [ "$USB_MASS_STORAGE" = 1 ] && create_storage
124 [ "$USB_SERIAL" = 1 ]       && create_serial
125 [ "$USB_ETHERNET" = 1 ]     && create_network
126 [ "$USB_HID" = 1 ]          && create_hid
127
128 init_end
129
130 # start additional services
131 [ "$USB_SERIAL" = 1 ] && systemctl start
      serial-getty@ttyGS0.service
```

drive is ideal for installing drivers and application software. For example, Windows needs a suitable program to access the serial port, and it would be possible to provide Putty on the emulated drive of the Pi Zero. This procedure is familiar from commercial UMTS/LTE sticks: When first plugged in, they register as a CD drive with the driver software. After installing the driver, Windows then identifies the stick as a WiFi dongle.

Physically, the mass storage emulated by the Pi Zero is an image file. Do not access it simultaneously on the host and Raspberry Pi – it is not a network drive. However, the Pi Zero can also serve up this kind of drive by NFS or Samba, because it plays the role of an Ethernet adapter with two ends.

Mini-Network

A normal USB Ethernet adapter has a USB plug on one side and a socket for a network cable on the other. The wire usually leads to a router or switch but could also be plugged directly into another computer. The Pi Zero as a network gadget does exactly that – the only difference being that the cable only exists virtually and the computer on the other side is the Zero itself.

Windows and Linux detect the adapter as a network interface. For communication with the Pi Zero to work, both computers need to have addresses on the same subnet. The easiest way to do this is to use Zeroconf/Avahi. If the service is running on both sides, the participating computers negotiate the technical parameters independently; otherwise, manual configuration will do the trick. Simply assign both computers a fixed IP address in your address range.

Once configured, both devices form a mini-network, and you can launch servers (e.g., a storage or web server) on the Pi Zero and make them available to the host. The transfer rates are in the Fast Ethernet range; depending on the direction, `iperf` indicates 80-120Mbps as throughput.

Consoles and More

In the third emulation, the Pi Zero pretends to be a serial interface. If the Raspberry Pi starts a virtual console on the port (the `raspi2go.sh` script does this automatically), you can access the Raspberry Pi on the host with a terminal emulator (Putty on Windows; Screen or Tio on Linux), which is especially useful if the network configuration is not working correctly.

Another possible application is the simulation of sensor data (for sensors with a UART interface). In this case, a program on the Pi Zero simulates the data and sends it to the host over the serial interface. This scenario is ideal for software development because it not only makes it very easy to process data, but also avoid errors.

Attack Tool

The emulation of a keyboard (HID, or human interface device) is prepared but not activated by default, because keyboards disguised as USB sticks are a popular tool for hackers. Lying around, apparently lost in the company car park, curious finders guilelessly plug the stick into a PC and don't notice that it is sending commands over the keyboard interface.

Keyboard mode, as well as the emulated network, makes the mobile Pi Zero the ideal tool for pen testers, because this kind of access to PCs bypasses all the firewalls and security mechanisms. Admins in companies therefore disable the USB ports on the employees' PCs (either in the BIOS or in software) as a precaution. A complete implementation of a toolbox is provided by the GitHub project [10]. If you use it, you need to know what you are doing and be sure you have permission, wherever you may be.

Conclusions

The Pi Zero again demonstrates its universal utility value in various fields of application. With a mobile PC you can carry with you the server environment of your choice, use the Pi for demonstration purposes, or simply avoid overloading a host with unnecessary software.

Programmers, on the other hand, appreciate the genuine environment with all the physical interfaces that the Raspberry Pi offers, in contrast to an emulation (e.g., with Qemu). The overhead for modding the mini-computer, including the software installation, is definitely manageable. ■■■

Info

[1] Linux USB gadget mode API: *https://www.kernel.org/doc/htmldocs/gadget/intro.html*

[2] Geekworm USB adapter: *https://geekworm.com/products/raspberry-pi-zero-w-badusb-usb-a-addon-board-usb-connector-case-kit*

[3] Zero Stem for North America, Europe, and Australia: *https://zerostem.io/sellers/*

[4] Case for Geekworm adapter: *https://www.thingiverse.com/search?sort=relevant&q=Pi+Zero+USB-A+Addon+Case&type=things&dwh=935e6a88da3df73*

[5] Case for Zero Stem: *https://www.thingiverse.com/thing:2516525*

[6] My angled housing: *https://www.tinkercad.com/things/kOFOeVrnuFP-pi-zero-with-90-usb-dongle*

[7] raspi2go on GitHub: *https://github.com/bablokb/raspi2go*

[8] Files for this project: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/234/*

[9] Pi Zero in read-only mode: *https://learn.adafruit.com/read-only-raspberry-pi/*

[10] Pi Zero as an attack tool: *https://github.com/mame82/P4wnP1_aloa*

Author

Bernhard Bablok works at Allianz Technology SE as an SAP HR developer. When he's not listening to music, cycling, or walking, he deals with topics related to Linux, programming, and small computers. He can be reached at *mail@bablokb.de*.

If you survived the early PC era, you will certainly remember a simple (but stable, for its time) operating system known as MS-DOS. Microsoft's old command-line OS later became the foundation for the first flaky generations of Windows, but then, as now, the purists were perfectly happy with clear and simple text commands.

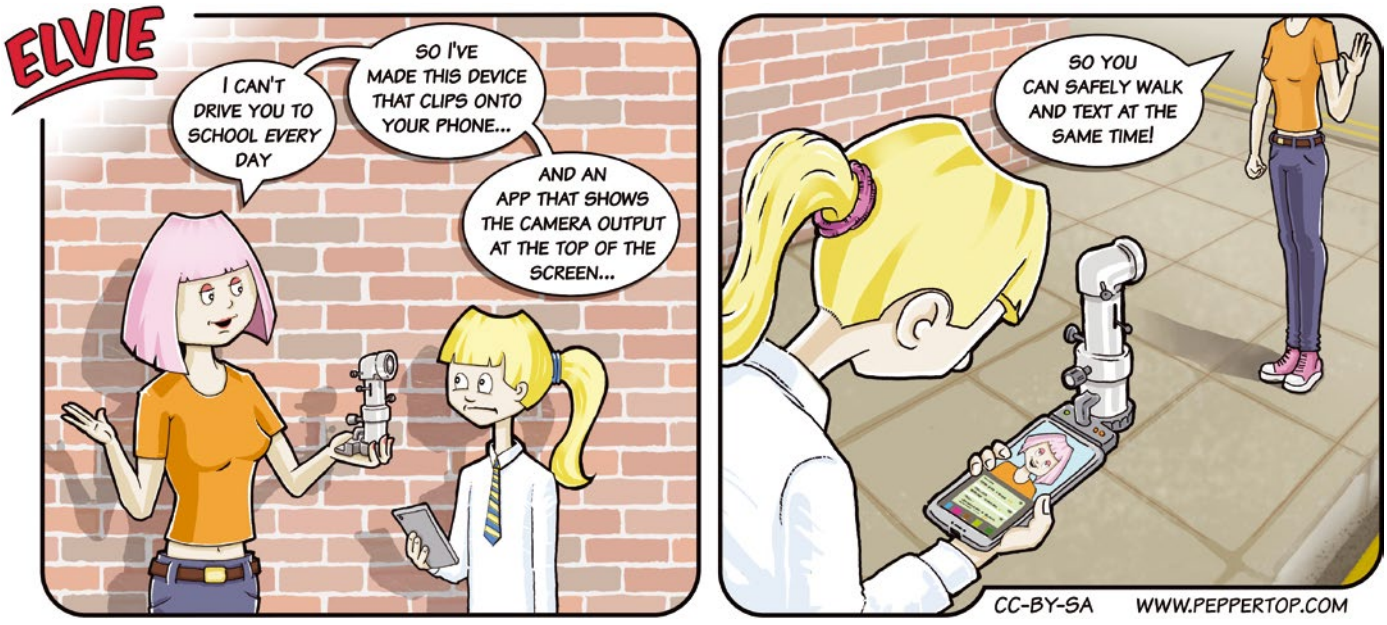
One of the most memorable things about MS-DOS was the vast quantity of software written for it – games, word processors, spreadsheets, and other tools that provided millions of families with their first exciting steps into the world of personal computing.

MS-DOS was all closed source, but that didn't stop the FOSS community from developing an equivalent version that would allow the world to keep running legacy DOS software. The FreeDOS project turned 25 this year, and we're proud to include an article from founder Jim Hall on how you can get started with FreeDOS and the gallery of iconic DOS games and applications.

Also, inside, we check in with the FlightGear flight simulator and roll out tutorials on LÖVE animation and printing in the shell.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE



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

MADDOG'S DOGHOUSE

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

maddog calls for courage and imagination to overcome our prejudices and address the world's perplexing problems.

BY JON "MADDOG" HALL

Shoulders to Stand On

Today I received news that Katherine Johnson, a NASA mathematician who performed many of the calculations used by the space program to put people on the moon, has died at the age of 101.

For those of you who did not see the movie *Hidden Figures*, Ms. Johnson had to fight against both racial and sexual discrimination to have her work accepted in the white male-dominated climate of that day.

I was born in Baltimore, Maryland, in 1950. I saw the "white persons water fountain," the "white persons lunch counter," and the signs that told black people to go to the back of the bus. Even many of our religious leaders told us that the white race needed to "take care of our black brothers and sisters." With all the prejudices that were present at that time, we can appreciate Katherine Johnson's contribution all the more knowing the obstacles she had to overcome. We can also see how the space program benefited from her presence.

In FOSS, many of us work remotely, so we often do not know the race, age, nationality, or (in a lot of cases) sexual identity of the people we communicate with every day. Many FOSS people have names like "rasterman," "maddog," and "IT Guy," which masks a lot of the things that make us different.

The phrase "show me the code" was an early cry that said, "I really do not care about our differences, only the code you create and how good it is." The FOSS community is at its best when we hold true to this ideal.

I have traveled the world, more than 100 countries, and visited most of them more than one time. For many of them, I have spent not just a day or two in the country, but weeks or months.

Every country I have visited has large and modern capital cities, most with rich and poor people inside of the city. Some countries I have visited do indeed have less infrastructure than other countries and fewer universities, but the Internet and modern technologies can bring education to places where it has never been before. Free and Open Source Software and Hardware (FOSSH) allows people with less capital and fewer resources to do more.

When I started programming (around the same era as the events described in *Hidden Figures*), networking was carrying boxes of cards down the hall, security was locking the computer room door at night, and graphics were ASCII art on a line printer (I question how many people even remember a "line printer").

Even the smallest computers cost hundreds of thousands of dollars (and USD$1,000 was a lot of money back then). Not having access to a computer at a company or university meant you could not learn computing. Today I can simulate those "huge" computers using SIMH running on a Raspberry Pi, and my programs run even faster than they did on the big iron.

We now have the tools; what we need is courage and imagination.

Courage to believe that if we work together we can fix some of the problems that we see. Courage to throw out our built-in prejudices. Imagination to think outside the box and create solutions for all, and courage again to implement those solutions.

I would like to see computer science and engineering mentors who would encourage young people to enter the field no matter what their race, gender, sexual orientation, or nationality. Help them learn the things that we learned over the years. Answer their questions. Spend time with them.

If you believe in the ways of FOSSH, then petition your schools, governments, and employers to use FOSSH and to help build the infrastructure in your town, state, or country to create local jobs. Yes, you are tired at the end of the day, but if we all work together, then "many hands make light work," as we know in FOSSH.

I have heard many times in my life the words "It cannot be done." I am sick of those words. There are many things that "cannot be done." I cannot jump to the moon. I can not hold my breath for 50 hours. However many things that people say "cannot be done" are simply difficult.

Years ago, an engineer told me that Unix could not configure itself the way that VMS did. In two hours, I developed a method of having our installation configure itself. I showed it to him and said, "I did the impossible in two hours; what if it were merely 'hard'?"

Some of you will question why, in a technical magazine, I write about courage and imagination. It is because, at this time, we need it more than ever.

Carpe Diem ■■■

Reviving old tools and games with FreeDOS

What's Old Is New

The FreeDOS Project turned 25 years old this year. We'll show you why a free version of DOS is still cool in 2020.

BY JIM HALL

Throughout the 1980s and into the 1990s, I was a great fan of MS-DOS. Our family had a PC at home, and I grew up writing my own programs and tapping out commands on the DOS command line. I considered myself a DOS "power user." I even wrote my own programs and utilities to enhance and expand the DOS command line.

In Spring 1994, I was finishing my junior year in physics at the University of Wisconsin-River Falls. It was around this time that I started to read articles in trade magazines where Microsoft announced that the next version of Windows would do away with MS-DOS. Have you used Windows 3? I found it to be clunky and slow, no match for MS-DOS and the stable of mature DOS applications and utilities. I decided that if Windows 4 would be anything like Windows 3 (Figure 1), I wanted nothing to do with it.

In contrast, MS-DOS was fast and stable and offered a mature set of applications that ranged from spreadsheets to word processors to communications systems to games (see the box, "Memorable DOS Applications and Games"). I used them all: Lotus 1-2-3 and As-Easy-As spreadsheets to analyze physics lab data, WordPerfect and Galaxy word processors to write class papers, ProComm and Telix to dial into the university's computer lab, and Doom and TIE Fighter to relax when not doing work.

I preferred DOS over Windows. And in 1994, Linux wasn't ready as a full operating system replacement for me.

I'd already installed Linux by this point. I found the Softlanding Linux System (SLS) the year before, on a friend's recommendation. Linux sported a powerful command line with the same tools that I had already used in the university's Unix computer lab. And Linux came with the source code, so self-taught programmers like me could study the source code and learn from it.

I looked at Linux as a model. If programmers from around the world could come together to create an advanced system like Linux to replace Unix, then surely we could do the same with DOS. If Microsoft was abandoning MS-DOS, it was up to us to create our own version of DOS to replace it.

Launching an Open Source Software Project

On June 29, 1994, I made a small announcement on *comp.os.msdos.apps* about my idea.

"Announcement of PD-DOS Project:

"A few months ago, I posted articles relating to starting a public domain version of DOS. The general support for this at the time was strong, and many people agreed with the statement, 'start writing!'

"So, I have...

"Announcing the first effort to produce a PD-DOS. I have written up a 'manifest' describing the goals of such a project and an outline of the work, as well as a 'task list' that shows exactly what needs to be written. I'll post those here, and let discussion follow."

I thought "PD-DOS" was a good name at the time. As I wrote in a

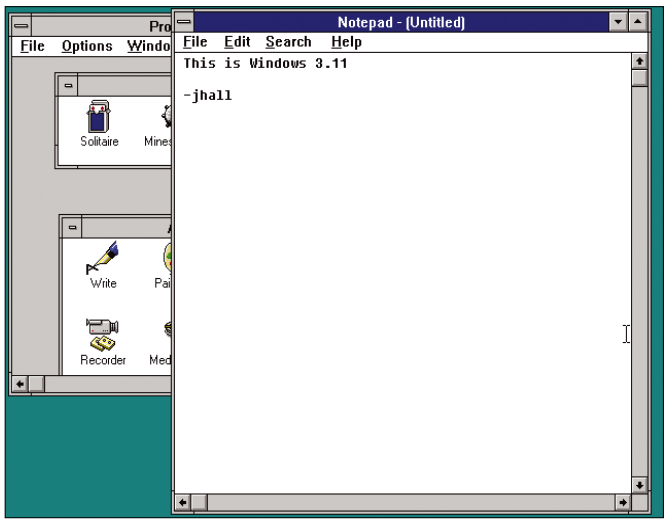


Figure 1: The author found Windows 3 clunky and slow.

manifesto posted to the same group, PD-DOS was short for "Public Domain DOS," because we would create our own implementation of DOS.

"I would like to form a group that will, eventually, create another implementation of MS-DOS. DOS appears to be a popular system, and there is plenty of hardware already available that is ready to support it. Microsoft will not develop DOS forever, and one cannot count on 'other' commercial programming firms such as IBM or Digital to continue enhancing DOS. I feel it is then up to those on the Internet to develop their own 'Public Domain DOS' (hereafter, PD-DOS) and I feel there is a lot of support for this from people on the 'net'."

People thought this free DOS project was a pretty neat idea, and several people contacted me right away to volunteer their time. Like me, they had also written their own extensions to MS-DOS. Soon, we had collected utilities that reproduced much of the MS-DOS commands, and others that introduced improved features.

But it didn't take long to realize that Public Domain DOS was the wrong name. Our manifesto set the goal that "Any effort that goes into writing a PD-DOS would … be released under the GNU General Public License." This made our project not "public domain" software, but "Free" software.

Memorable DOS Applications and Games

The DOS environment was home to many classic applications and games. A few of my favorites are…

WordStar

For a while in the early to mid 1980s, WordStar (Figure 2) was the most popular word processor program for DOS. Its remarkably simple user interface makes it easy for beginners to get started, but its powerful word processing features continue to support expert users.

Author George R.R. Martin famously remarked that he writes his books using WordStar on DOS. And it's hard to deny the appeal of distraction-free writing with WordStar.

VisiCalc

DOS was popular in office environments, and it's hard to imagine an application better suited for the office than a spreadsheet. Written by Dan Bricklin and Bob Frankston, VisiCalc (Figure 3) was the first spreadsheet for personal computers. While it lacks features common in modern spreadsheet programs, VisiCalc still feels familiar.

If you're interested in trying VisiCalc for yourself, Dan Bricklin released the original DOS program for free via his website [1].

Doom

DOS had a ton of games, but none is so iconic as the original Doom (Figure 4). While not the first shooter, Doom quickly became the metric against which other games measured themselves. Pick up your shotgun and prepare to gun down some imps and demons.

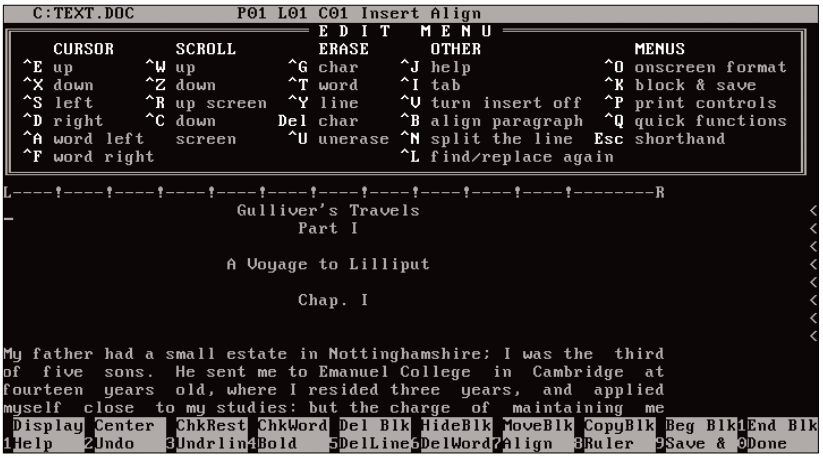


Figure 2: Editing files in WordStar was pretty simple, but plain.

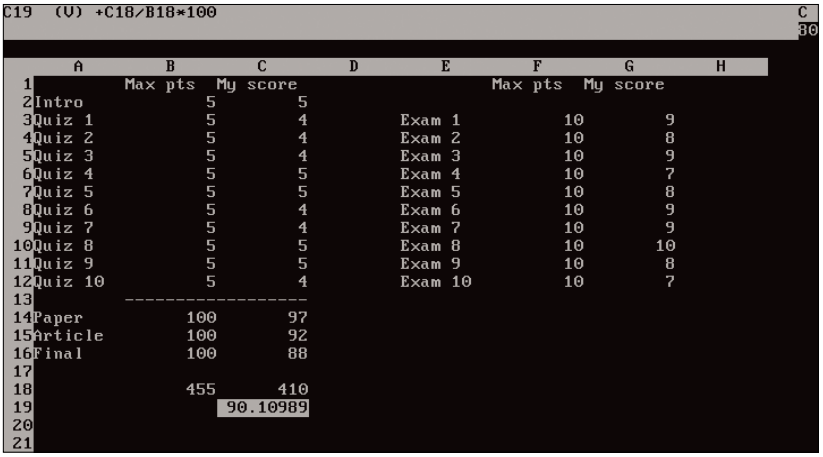


Figure 3: Managing class grades in VisiCalc, the first spreadsheet program for personal computers.

Figure 4: Playing Doom: You got the shotgun!

We changed our name to Free-DOS about a week after the PD-DOS announcement. We later dropped the hyphen to just FreeDOS.

We made our first Alpha release in September 1994, just a few months after the announcement. And we released FreeDOS Alpha 2 a few

How To Get Started With FreeDOS

Installing an operating system may seem daunting, but FreeDOS makes it easy. Like modern Linux distributions, the install program in FreeDOS 1.2 guides you through each of the steps to set up your hard drive and install everything. Here is a quick breakdown to installing FreeDOS on your computer.

1. How do you want to run FreeDOS? Most people use FreeDOS to play classic DOS games or to run the occasional legacy DOS application. If that's you, then you probably don't want to replace your current operating system. Rather, you will likely want to install FreeDOS inside a PC emulator or a virtual machine. FreeDOS runs well in all the major PC emulators, such as VirtualBox, Qemu, or Gnome Boxes. So to get started, you'll want to have one of these virtual machines ready to go.
2. Booting the CD-ROM installer gives you a menu (Figure 5). You can choose to install FreeDOS or boot from the system hard disk or from a diskette.
3. Select your preferred language; then welcome to the FreeDOS 1.2 install program.
4. If your `C:` drive isn't partitioned for DOS, the installer detects that. To partition your hard drive, the installer jumps to the FDISK program (Figure 6).
5. Select *1* to create a DOS partition as your `C:` drive. At the next screen, select *1* to create the primary DOS partition. FDISK creates your `C:` drive partition and marks it as "Active." After you partition your hard drive, you need to reboot for FreeDOS to see the changes.
6. After rebooting, the installer starts up again automatically. Select your preferred language and continue with the installation. Since you just created your DOS partition, the installer can format it for you (Figure 7).
7. The FreeDOS 1.2 installer has two default install modes: install only those packages that reproduce the functionality of classic DOS (*Base*) or install everything (*Full*). Because FreeDOS is open source software, we give you the option to install source code, too.
8. Then sit back and let FreeDOS install itself. This may take several minutes, especially if you installed everything (*Full*).
9. Now reboot your system to begin using FreeDOS.
10. That's it! You are now using FreeDOS! (Figure 8)

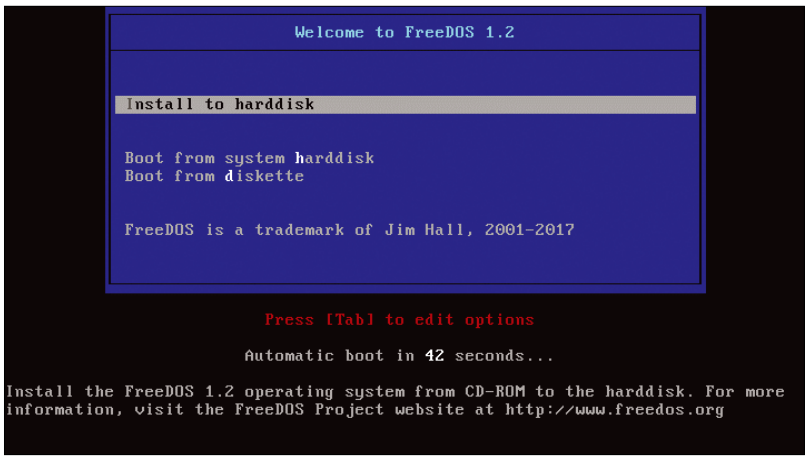


Figure 5: Booting the FreeDOS install CD-ROM.

Figure 6: Partition your hard drive with the FDISK program.

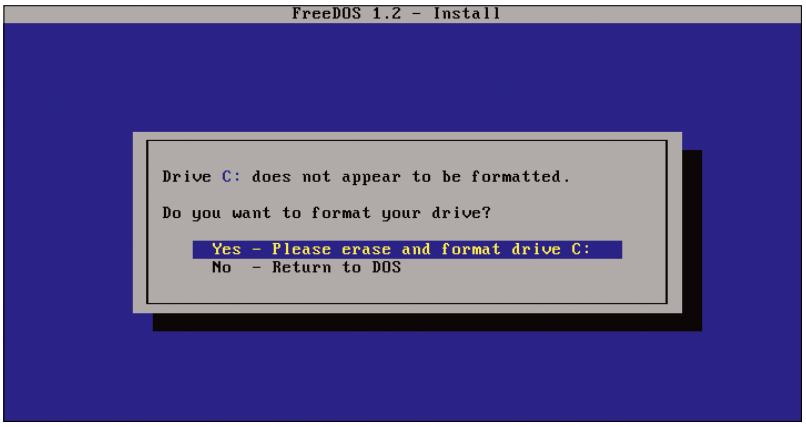


Figure 7: You'll be prompted to format and erase your new FreeDOS partition.

months later, in December 1994. We were off to a great start.

You may be familiar with our other milestones. We released Alpha 3 through Alpha 5 over the next few years, finally moving on to Beta releases starting in 1998. Over the next several years, we slowly made our way from Beta 1 to Beta 9 SR2. In 2006, we finally released FreeDOS 1.0. Things slowed down after that, with FreeDOS 1.1 in January 2012, and FreeDOS 1.2 in December 2016 [2] (See the box, "How to Get Started with FreeDOS"). MS-DOS stopped being a moving target long ago, so we didn't need to update as frequently after the 1.0 release.

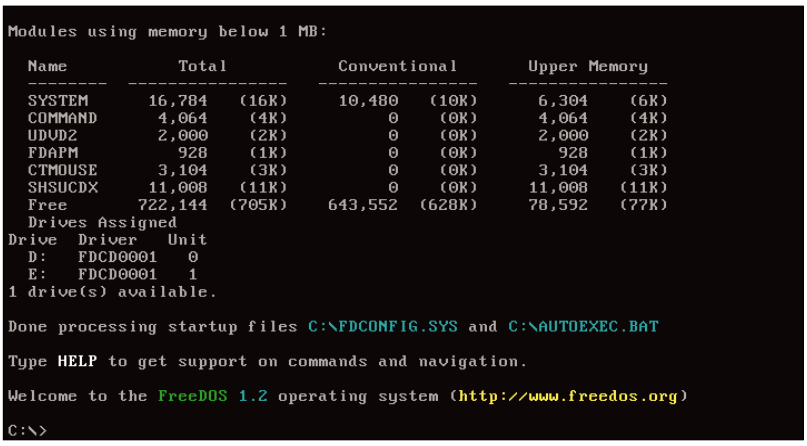


Figure 8: The FreeDOS 1.2 command prompt, after booting.

It's About the Community

The FreeDOS Project has been around for over 25 years. We passed our milestone anniversary on June 29, 2019.

FreeDOS got to where it is because developers worked together to create something. In the spirit of open source software, we always contribute to each other's work by fixing bugs and adding new features. We treat our users as codeveloper; we try to find ways to include people according to their talents, by writing code or writing documentation. And we make decisions through consensus based on merit. If that sounds familiar, it's because those are the core values of open source software: transparency, collaboration, release early and often, meritocracy, and community.

I think FreeDOS remains active due to the number of developers who continue to work on it. While many of our original developers have dropped off, we make it easy for new developers to join. You can download the FreeDOS 1.2 distribution and immediately start coding in C, Assembly, Pascal, BASIC, or a number of other programming languages.

Shop the Shop ⇒ shop.linuxnewmedia.com

Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

Want to subscribe?

Searching for that back issue you really wish you'd picked up at the newsstand?

shop.linuxnewmedia.com

DIGITAL & PRINT SUBSCRIPTIONS

SPECIAL EDITIONS

Today, many people around the world install and use FreeDOS to play classic DOS games, to run legacy business software, to develop embedded systems, or to install a BIOS update. These days, I think that still represents most of the usage of FreeDOS. Although I'll admit most people probably run FreeDOS to play DOS games, and that's okay with me. DOS had a lot of great games.

FreeDOS has grown into a modern DOS. We've moved beyond "classic DOS" and now FreeDOS features a selection of open source development tools including compilers, assemblers, and debuggers. We have lots of editors beyond the plain FreeDOS EDIT editor, including Fed, Pico, TDE, and versions of emacs and vi. FreeDOS supports networking and provides a simple but functional graphical web browser (Dillo, Figure 9). And we have a collection of Unix-like utilities, which should make Linux users feel at home on the command line. (See box, "How to Install DOS Applications.")

But under the hood, FreeDOS is still DOS. And that comes with a certain set of assumptions and limitations. Like any DOS, FreeDOS will remain 16-bit and will retain focus on a single-user command-line environment. While we borrow certain Linux utilities for the FreeDOS command line, we don't want to turn FreeDOS into a watered-down Unix. Twenty-five years after its inception, FreeDOS is still just DOS. And that's cool with us. ■■■

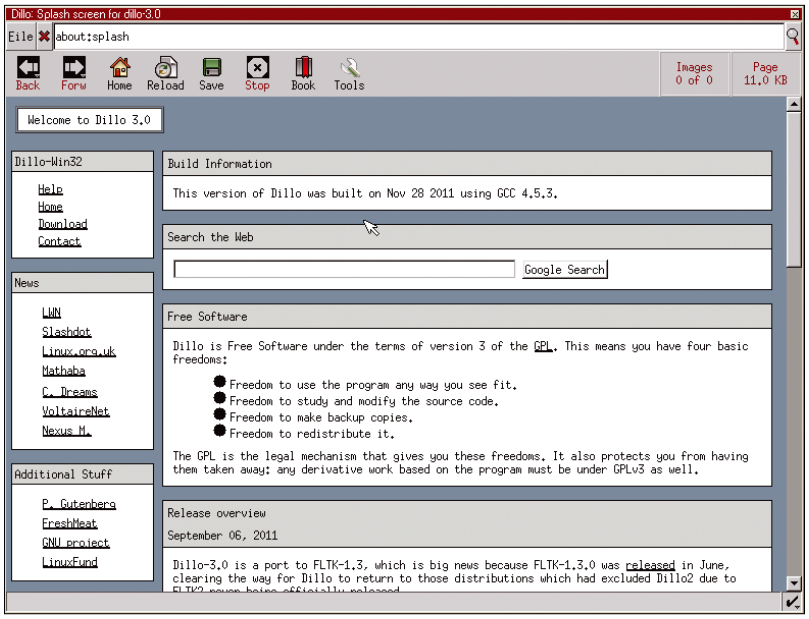


Figure 9: The Dillo web browser is a simple yet functional graphical web browser. It is included with FreeDOS 1.2.

```
QUAKE               <DIR>  11-30-2019  9:07p
STARW               <DIR>  12-18-2019  5:36p
WORD                <DIR>  12-01-2019  9:33p
DOOM19S  ZIP     2,450,688  12-24-2019 10:16p
        1 file(s)      2,450,688 bytes
        8 dir(s)     243,081,216 bytes free
C:\TEMP>unzip -l doom19s.zip
Archive:  doom19s.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
      184  02-03-1995 18:13   INSTALL.BAT
    17303  02-02-1995 01:23   DEICE.EXE
      226  02-02-1995 01:23   DOOMS_19.DAT
  1439232  02-02-1995 01:24   DOOMS_19.1
   994588  02-02-1995 01:24   DOOMS_19.2
---------                     -------
  2451533                     5 files
C:\TEMP>unzip doom19s.zip -d doom19s
Archive:  doom19s.zip
  inflating: doom19s/INSTALL.BAT
  inflating: doom19s/DEICE.EXE
  inflating: doom19s/DOOMS_19.DAT
 extracting: doom19s/DOOMS_19.1
  inflating: doom19s/DOOMS_19.2
C:\TEMP>_
```

Figure 10: Unzipping the Doom installer.

How To Install DOS Applications

On Linux, you usually install new applications by finding a package in your distribution's repository. If your preferred Linux distribution doesn't include that application, you might have to recompile the program from source code.

But on DOS, things were much simpler. The original DOS didn't support a package management concept. Instead, you installed applications by downloading a ZIP file archive, and extracting it into a new directory (Figure 10). Other applications required you to run an install program, usually included in the ZIP file.

Don't be afraid of trying new DOS applications. Install them and try them out. If you don't like them, you can easily remove them by deleting the program's directory.

Info

[1] VisiCalc: *http://danbricklin.com/visicalc.htm*

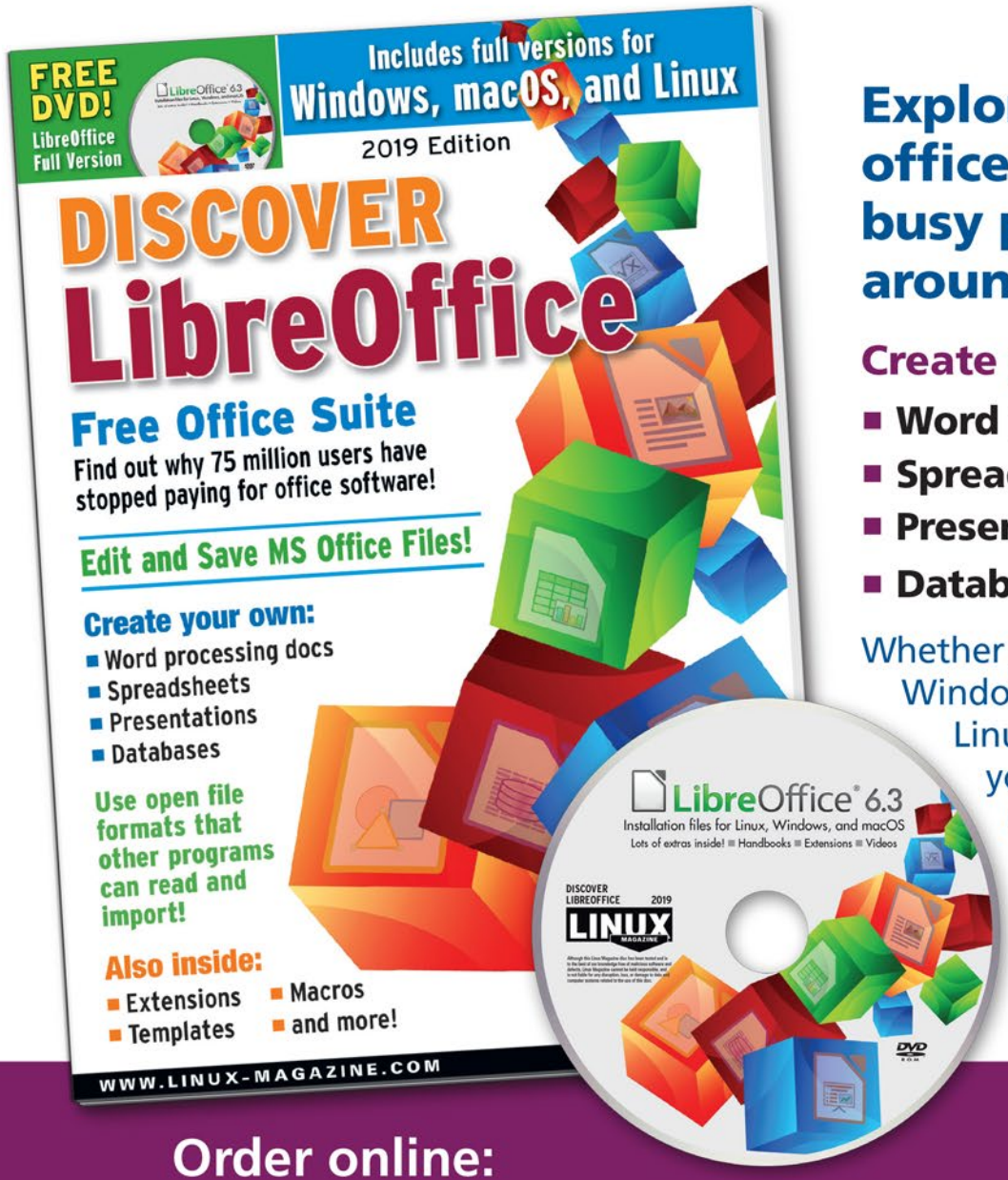
[2] FreeDOS: *https://www.freedos.org*

The Author

Jim Hall is an open source software advocate and developer, probably best known as the founder of FreeDOS. Jim is also very active in usability testing for open source software projects like Gnome.

Shop the Shop shop.linuxnewmedia.com

DISCOVER LibreOffice



Explore the FREE office suite used by busy professionals around the world!

Create your own:

- Word processing docs
- Spreadsheets
- Presentations
- Databases

Whether you work on a Windows PC, a Mac, or a Linux system, you have all you need to get started with LibreOffice today. This single-volume special edition will serve as your guide!

Order online:
shop.linuxnewmedia.com/specials

For Windows, macOS, and Linux users!

Free FlightGear flight simulator

Above the Clouds

The free flight simulator FlightGear has improved in terms of stability and realism in recent years and offers joystick pilots massive potential for fun.

BY PETER KREUßEL

The FlightGear flight simulator [1] is one of the veterans of the Linux gaming world. More than 20 years after its initial release, the program has matured into complex simulation software that brings many aspects of flying to the screen in a realistic way. For this article, we used FlightGear v2018.3.

The images in this article were taken with the best possible graphic settings (see box, "Graphics Settings"). At a screen resolution of 2540x1440 pixels, this only utilized about half of an AMD RX 580 card's capacity (Figure 1). Even with slower cards, FlightGear remains playable, because after switching off rendering options like *3D clouds* the simulator's resource requirements drop significantly.

Artificial World

FlightGear features numerous aircraft types whose cockpits and exterior views the program renders as 3D models. This means that the aircraft appear realistic from different angles and in different lighting conditions. A huge amount of work was also put into representing the landscape. It is based on a model of the entire Earth's surface that processes data from the space shuttle program, the global coastline database GSHHG [2], and data from the European Environment Agency and OpenStreetMap. In addition, around 180 contributors have designed local features, such as airports or cities.

FlightGear renders landscapes around the world with natural-looking vegetation, including impressive mountain scenes from the North American Blue Ridge Mountains to the Himalayas (Figure 4). Compared to the brilliant nature scenes, low overflights over cities are somewhat disappointing. For example, in Nuremberg, Germany, the castle looks a bit different in reality than on the screen (Figure 5), which is only to be expected. While landscapes automatically generated from elevation data look quite realistic, the scenery developers have to design recognizable buildings by hand. From a normal overflight altitude of about 6,000 to 7,000 feet, the city views generated from OpenStreetMap data look at least halfway realistic.

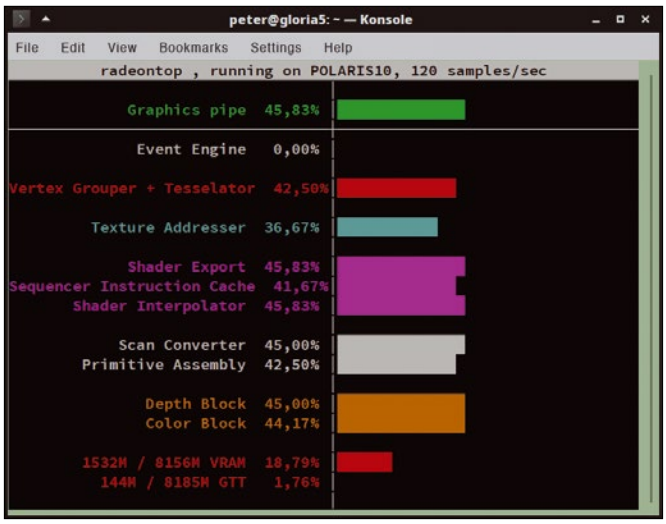


Figure 1: The RadeonTop graphics performance monitor shows that with an AMD RX 580 graphics card, even at maximum rendering quality in FlightGear, there is still plenty of capacity.

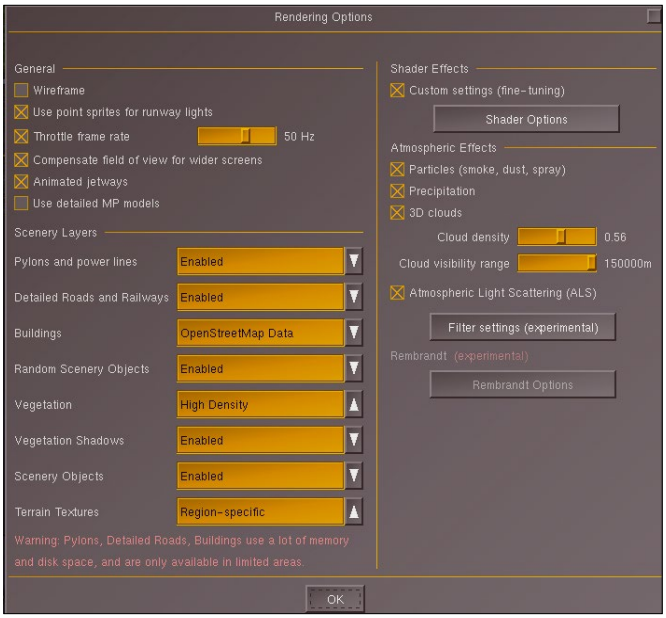


Figure 2: The *Rendering Options* allow you to adapt FlightGear's resource consumption to an acceptable frame rate on your system.

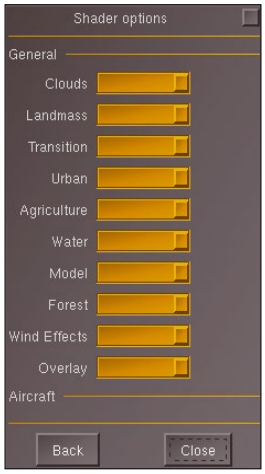


Figure 3: After selecting *Custom Settings*, all *Shader options* were set to the maximum for the best possible graphics here.

Graphics Settings

In the graphics settings below *View* | *Rendering Options* (Figure 2), you can adapt the graphics to suit your graphics adapter's performance. *Use point sprites for runway lights* is enabled in *General*, which means you have three-dimensional rendered runway lights. *Throttle frame rate* restricts the frame rate to a maximum of 50 frames per second, helping the game to run more evenly. *Compensate field of view for wider screens* is another helpful setting.

In the Scenery Layers section you can decide whether FlightGear shows you *Pylons and Power lines* and objects such as barns (*Scenery Objects*), based on a database, and also adds *Random Scenery Objects* to the landscape. The dialog warns you about the memory consumption, which can climb to 8GB. You can also choose the vegetation density, and decide whether or not trees and bushes should throw shadows.

The *Region-specific* option for *Terrain Textures* colors the terrain and vegetation to match the region, for example, the Blue Ridge Mountains actually look blue instead of green.

The Shader Effects have a noticeable influence on the frame rate. You would normally see a simple slide control here, but if the *Custom Settings* option is enabled, then the *Shader options* button opens a dialog for detailed settings (Figure 3) that govern the rendering quality of scenery display. Experience shows that the *3D clouds* option consumes the most graphics capacity. Without 3D clouds, the FlightGear world appears far less vivid, but it is often sufficient to just reduce the density of the clouds or the distance at which they become visible.

Atmospheric Light Scattering computes the light color on the basis of the light scatter in the atmosphere. Without it, the FlightGear world is far paler at typical altitudes.

There are also differences in the degree of realism in the interiors of the aircraft models: The cockpit surfaces of the Cessna 172P almost look as if you could touch them. The soft reflections in the instrument glasses may impair readability, but this is no different in reality. The cockpits of the Boeing 777-300 (Figure 6) and MDD F-15C do not look quite as fresh, although they are still above-average FlightGear models that make flying fun.

Figure 4: You can fly above the Himalayas in FlightGear, for example, on a flight through Nepal from the Tenzing-Hillary Airport in Lukla (VNLK) to Kathmandu (VNKT).

Flight Machine

FlightGear not only simulates the landscape and the appearance of the aircraft, but also many technical conditions (Figure 7). Navigation instruments like the VHF Omnidirectional Range (VOR) actually work. (This is a rotary radio beacon that emits angle information. In combination with a radio-based rangefinder, the aircraft position can be precisely determined.) You are confronted with a dreaded stall scenario, where the aircraft stalls

Figure 5: This is the degree of realism you can expect in a low-level flight (which would be prohibited in real life) over a major city in Germany. If you are familiar with Nuremberg, you will not recognize it in FlightGear.

Figure 6: The cockpit of a Boeing 777 (first flight in 1995) is now based almost entirely on screens. The analog dials are only backup instruments.

after dropping below the minimum speed with the resulting need to permanently monitor the airspeed, just like a real pilot.

Time to launch the flight simulator and a Cessna 172P: This veteran aircraft has been one of the best maintained models in FlightGear from the outset. In recent versions, the developers have once again polished its cockpit. The single-engine high wing aircraft is not only relatively easy to fly in reality, but also in FlightGear. It is a good model with which to learn flying skills.

If you run FlightGear with the `--launcher` parameter, a friendly dialog appears to let you select the aircraft, the starting airport, the weather, and other FlightGear settings (Figure 8). In the *Aircraft* section, click on the *Cessna 172P Skyhawk (1982)* that I mentioned earlier.

How about an afternoon flight over the beautiful scenery of the Blue Ridge Mountains in South Carolina? Enter ICAO Identifier 33A in the *Location* category and select the picturesque *Fairview* airfield. ICAO stands for The International Civil Aviation Organization, which – among other things – assigns globally unique codes to airports. In the next screen, select *Runway 32*. This runway points, as its name indicates, in the direction of 320 degrees (northwest) that is straight to the Blue Ridge Mountains.

You can choose the weather in FlightGear. We ordered a summer afternoon in the Environment section below Time & Date (*Time of day*: *Afternoon*, *Season*: *Summer*). In Weather, owners of newer computers can enable *Advanced weather modeling*. We disabled the *Real-world weather* option – it evaluates the weather data of a METAR report, a standardized weather report for airports. Instead we chose *fair weather* as the *Weather scenario*, to avoid cloud cover blocking our view of the landscape.

If you have an Internet connection with at least one megabit bandwidth, you can select the default *Download scenery automatically* [3] setting in the *Settings* section. In *Multiplayer*, you can also enable a connection to a FlightGear multiplayer server, which integrates other players' planes into your simulated world.

The simulator is normally launched in full screen mode (*Start full screen*). The *Renderer* I would recommend is *Atmospheric Light Scattering*. This method calculates the light scattering in the atmosphere, which makes the simulation look far more realistic for the large visibility ranges of high-altitude flights. Owners of more-or-less up-to-date graphics cards can treat themselves to *Anti-Aliasing 4x*, which completely irons out the ugly jagged diagonal lines that otherwise appear.

If you enter the `--http=8080` option in *Additional Settings*, you can retrieve an OpenStreetMap-based moving map from the IP address of the FlightGear computer on port 8080 (*http://<FlightGear-IP>:8080*) via web browser (Figure 9). This lets you turn a tablet into a GPS device with a map display, like the one many amateur pilots take with them to facilitate navigation in their Cessnas.

Getting Airborne

Now press *Fly!* This launches the simulation, which will take a number of seconds. The first task is to start the aircraft's engine. Fortunately, a Cessna 172 is easy to start. First zoom out (Shift+X) until you can see the whole of the Cessna cockpit. Ctrl+X switches back to the default zoom level later on. Pressing Ctrl+C highlights all the mouse-controllable cockpit elements in yellow and also shows you the names of the instruments (Figure 10). Pressing the same shortcut again disables this

Figure 7: Navigation instruments in the Cessna model look and work just like the real thing.

Figure 8: The `--launcher` start option opens a convenient wizard that lets you select the airport, aircraft, and many other FlightGear settings.

help feature. Press *Y* to hide the yoke, which is just annoying in this view. You can press the same key again later, if so desired, to display it again.

Then click on the two red toggle switches bottom left (*MASTER*) and additionally the shift lever right next to the ignition key (*AVIONICS POWER*). Then click six times on the *Primer* pull to the left below the red toggle switches to pump some fuel into the carburetor. Next, set the black throttle lever (*THROT PUSH OPEN*) bottom center to 25 percent. To do this, mouse over it and adjust the lever with the mouse wheel; a small pop-up shows you the percentage. Then push the adjacent red lever for *MIX PULL LEAN* all the way in (100 percent). Finally, press *S* for the starter for a few seconds to wake up the engine. In the menu with the aircraft name, there is usually an *Autostart* entry that handles all of these steps, but experienced pilots would never use it.

Theoretically, FlightGear can be controlled completely by keyboard, but serious flying is only possible with a joystick (see "Joystick" box). For control functions for which you cannot use the joystick buttons, there are also the FlightGear keyboard shortcuts, which you will find in the *Help* menu. An introduction to flying itself is given in the official FlightGear manual [4], chapter 8, *A Basic Flight Simulator Tutorial*.

All you have to do now is accelerate to take off. But will you find your way back to the airport of departure? Although our simple airfield does not have its own radio beacon for the approach to land, an antenna station nearby can be used to navigate to a position within sight of the runway. *Equipment* | *Map* lets you open the flight map, without which a pilot would never take off.

Joystick

For FlightGear, we recommend a three-axis stick, whose lever can not only be tilted, but also rotated around a third axis. In this way you can control the elevator, aileron, and rudder. In addition, the joystick should have a slider for thrust; push buttons for other aircraft functions are usually available anyway. A joystick like this is okay for convenient control, even if it does not realistically reproduce the aircraft yoke and rudder pedals. In the *Joystick Configuration* dialog, you will ideally want to assign axes 0 to 2 to the elevator, aileron, and rudder, and axis 3 to the thrust controller. In addition, you will want to have the *Elevator Trim* and *Rudder Trim* easily available; this is the zero position correction to compensate for air currents. For the Cessna, the remaining two pairs of keys are used to adjust the fuel mixture and flaps.

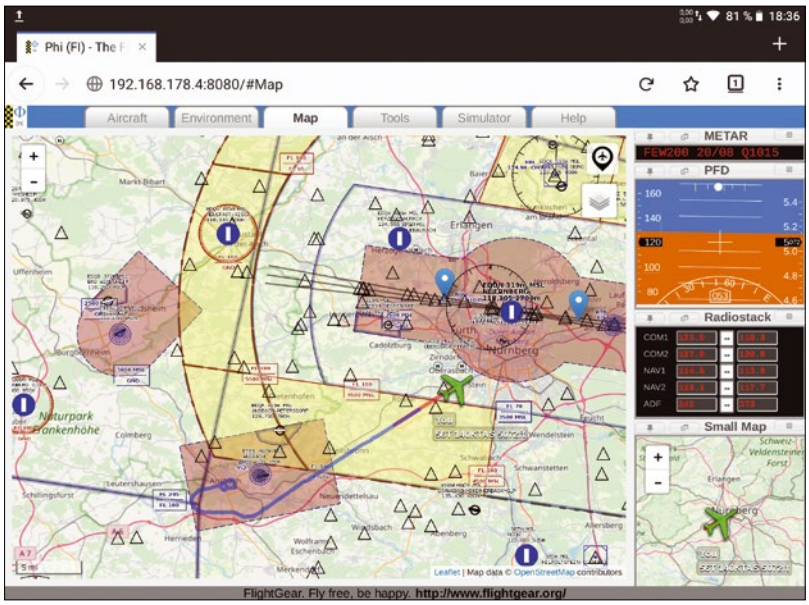


Figure 9: As a web server, FlightGear delivers an HTTP-based map view that you can open on a laptop or a tablet.

But thanks to FlightGear's *Map* feature, you don't have to have it on paper. The shortcut for this is Ctrl+J, even if the Equipment menu tells you it is Ctrl+M.

If you check the *Navaids* (navigation aids) box on the left side of the map, then the *SPA* VOR in Spartanburg appears on the right side below the aircraft. If you also enable the *Data* option, then you will see its frequency *115.7 MHz* on the map. F12 opens the radio settings, where you can enter this frequency in *Selected* for *NAV1*, the first navigation device. The SPA station will then turn light blue on the map. A light blue line also appears through the aircraft location, the line is known as the radial.

Now mouse over the dial of the *NAV 1 display*, and adjust it with the mouse wheel until the horizontal pointer moves to the center and the display *TO* appears to the right of it at the same time. The pointer may first move to the center in the case of an *FR* display ("from"). Then turn 180 degrees further, in our example up to 130 degrees.

A VOR always guides an aircraft along the set radial line, which now also runs through the current

Figure 10: Pressing Ctrl+C highlights all the clickable points in the cockpit in yellow and (in the Cessna) additionally shows you the names of the elements.

aircraft location on the map. To determine the position, combine the radial with the Distance Measuring Equipment (DME) display. This is a radio-based distance measuring device, often used in combination with a VOR (VOR/DME). In Fairview, the DME shows a distance of 12 miles to the Spartanburg station. VOR and DME together define a traceable position. If you center the needle of the VOR instrument at Radial 130 and move 12 miles away from SPA on Radial, you will fly over Fairview.

Now you can press Shift+B to release the parking brake, open up the throttle, and fly off to the northwest into the Blue Ridge Mountains.

Finding Your Way Home

Once you have had your fill of the scenery, or if the tank fill level (*Equipment* | *Fuel and Payload*) starts to drop towards the 50 percent mark, it's time to turn the plane southeast. What now follows is the procedure which is sometimes jokingly referred to as "Chasing the needle": If the vertical needle in the VOR instrument points to the right, then steer the plane past the 130 degree angle on the compass on the right. If it points to the left, then you steer to the left (Figure 11).

You may want to watch the autopilot first to see how it does this. A Cessna 172P from 1982 only comes with a simple version that cannot be programmed with a route consisting of waypoints. But the KAP-140 autopilot by Bendix [5] can fly along the radial of the active VOR. In the Cessna model, the autopilot cannot be activated using the FlightGear autopilot menu, but only by clicking on its control buttons in the cockpit.

If you switch it on by selecting *AP*, then it first holds the course and pitch of the aircraft. Click on *UP* or *DN* until the number on the right in the autopilot reads *0000* – this stands for a climb rate of zero feet per minute. In other words, the aircraft will maintain its current altitude. Then, using the left-hand dial (marked red in Figure 11), turn the heading indicator to the 130 degree radial of the home course determined before takeoff, and click on the *HDG* (heading) autopilot button.

Figure 11: If the vertical VOR needle is to the left of the center, then you fly slightly to the left of the 130 degree course you actually planned (pink). If it deflects to the right, turn in the same direction (blue). If the needle is in the middle, simply follow the planned course (white).

The Cessna now turns to 130 degrees, but does not yet compensate for the probable lateral shift in the course (vertical pointer in the VOR, also visible on the chart display). This only happens when you click *NAV* on the autopilot. Now the autopilot will take you back to the starting airport, but will stubbornly maintain altitude. You have to initiate the descent yourself, at the right time, by pressing the *UP* and *DN* buttons.

You also have to manually keep the speed at around 100 knots with the throttle stick. At a ground speed of 120 knots, the aircraft will travel two nautical miles per minute. You can read this off on the DME unit as long as you follow the radial. The mile is the unit commonly used in aviation, which is why the DME also measures in this unit.

If you are flying at 7,000 feet, it will take you 10 minutes to descend to roughly 1,000 feet at a sink rate of 600 feet per minute. This is equivalent to a flight distance of 20 miles. The distance to the VOR is shown by the DME. In our case, you still have to compensate for the 12 miles that the VOR is located behind our home airfield. In other words, you need to start descending 32 miles before the VOR by setting a sink rate of *-600* feet on the autopilot with the *UP* and *DN* buttons.

As soon as you see the runway (on the map in the browser or on the map display in the FlightGear window), you need to approach manually at about 60 knots. As you will now see, a successful manual landing requires some practice.

Jet Power

Once you have mastered the art of manual flying, the good old Cessna may seem too boring for you. Why not fly a real airliner like the Boeing 777-300, another quality FlightGear model? Its wiki page [6] explains how to launch the triple-seven if you do not want to use the shortcut via the *Autostart* menu entry. Aircraft-specific operating details can also always be accessed in FlightGear using the Shift+? shortcut.

It is clear that such a large aircraft's response to the controls is a little slower than that of the small Cessna. On the whole though, it's not a massive difference when using a three-axis joystick. The fact that flying the 777 still feels completely different is more due to the screen-based instruments

and the digital flight management computer that can automatically fly the aircraft along the entire route – if necessary, up to a few meters before touchdown.

You have already seen the cockpit of the Boeing 777 in Figure 6. The left screen combines airspeed indicator (left), altimeter (right), and compass (below). In the middle is the artificial horizon, which shows the horizontal and vertical position of the airplane even without a view to the outside. The navigation screen to the right with the flight management computer display (*Autopilot* | *Route Manager*) shows the planned course.

If the Boeing 777 with a top speed of 513 knots (950km/h) is not fast enough for you, try a Mach 2 fighter. The McDonnell Douglas F-15C "Eagle" (Figure 12) is recommended in FlightGear. The jet, designed as an air superiority fighter, will knock you clean out of your gaming chair even without G-force simulation.

Technical Problems

However, the first test with the F-15C was followed by disillusionment. This had nothing to do with the virtual aircraft, but with the technology of the PC on which the simulation was running: The game only managed disappointingly low frame rates of around 16 frames per second, and the rate also fluctuated strongly. This made it impossible to play FlightGear meaningfully.

Troubleshooting revealed that the load on the graphics card, like with other flight models, hardly exceeded 50 percent. However, one of the four CPU cores was running almost permanently at 100 percent load. Even switching off CPU-intensive simulation elements like the *Advanced Weather* system or *AI Traffic* hardly improved the situation. Was it really impossible to play FlightGear on a powerful AMD Ryzen 1800X CPU without any restrictions?

The solution was to add the sparsely documented and experimental startup option:

```
--prop:/sim/rendering/
  multithreading-mode=DrawThreadPerContext
```

It distributed the load over all four cores, keeping them well away from a full load scenario that would slow down the simulation. The simulation then ran smoothly at frame rates of around 50 fps. Only the head-up display of the F-15C did not move as fast as the aircraft itself.

Figure 12: The F-15C "Eagle" – shown with full afterburner here – is an agile fighter that can make you dizzy even in the simulator.

Conclusions

FlightGear is an impressive simulation game that allows computer pilots to experience the feeling of flying over a beautiful landscape. Many technical features, such as VOR navigation or – in commercial aircraft – even an instrument-based approach, can captivate geeks for hours. But you should not expect models, especially jets, to feel exactly like they do in reality, since the flight dynamics model is obviously too simple for that. The commercial X-Plane [7] flight simulator, which is also available for Linux, offers an alternative for this purpose. ■■■

Info

[1] FlightGear: *http://flightgear.org*

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

[3] Manual scenery download: *http://ns334561.ip-5-196-65.eu/~fgscenery/WS2.0/scenery-2.0.1.html*

[4] FlightGear manual: *http://flightgear.sourceforge.net/getstart-en/getstart-en.html*

[5] Bendix King autopilot: *http://wiki.FlightGear.org/Bendix/King_KAP140_Autopilot*

[6] Boeing 777 in the FlightGear wiki: *http://wiki.flightgear.org/Boeing_777*

[7] X-Plane: *https://www.x-plane.com*

FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software

Graham recently found the perfect use for his old Nintendo DS Lite. Thanks to having exactly the same screen resolution, it now runs the brilliant ZXDS Sinclair ZX Spectrum emulator. BY GRAHAM MORRISON

Video editor

OpenShot 2.5

OpenShot is one of those applications that has been around for such a long time you forget how far it's come. Way back in 2008, when it was first released, video editing on Linux was almost nonexistent. Back then, it was good enough to simply have something that could stitch a couple of clips together without running out of memory. Over a decade later, open source video editing has come a long way, and so too has OpenShot. Whether it's from your phone or an online gaming session, video has become ubiquitous, and an application that helps you get the most from your recordings is now a cornerstone of any operating system. OpenShot has become one of those cornerstone applications and is a strong candidate for being the Linux equivalent to Apple's iMovie, with elements from the truly professional FinalCut Pro thrown in for good measure. It's a nonlinear video editor that lets you drop various clips and recordings into a pool before dragging them onto a visual timeline where they can be cut against one another, blended, and otherwise made into a coherent whole.

1. Dynamic UI: The Qt-based interface can be modified to disable, shrink, detach, and reorder its various panels. **2. Clip view:** Collate everything you might need into a single location from which you clip elements into the time line. **3. Preview:** See the potential output of your video rendered as you make changes. **4. Time line:** Layer, cut, and blend between clips. **5. Audio:** Detach the audio tracks from the video to add different sounds, or to resync the audio you recorded. **6. Effects:** Drag and drop effects onto clips and see the results in real time. **7. Properties:** Use keyframes to move between different sets of property values.

OpenShot 2.5 has been described by the OpenShot team as "our largest release yet," and it's amazing how much progress has been made in the two years between this and the last major update. The two flagship features for this release, however, both deal with performance. This might not seem that exciting, but if you've ever tried to process and edit large video files, you'll know the difference between how often you use a powerful but slow application, versus one that gets the job done with as little delay as possible. OpenShot started off as the former and is quickly becoming the latter. The first performance feature is hardware encoding and decoding support. This does have the "experimental" caveat, but it means that if you're using any of the supported graphics hardware, you can benefit from a performance increase of 30-40 percent without putting any more strain on your CPU. Similarly, keyframe performance has also improved. Keyframes are a snapshot of values at a specific frame, and these values are interpolated between one keyframe and the next to create smooth transitions, blending, transformations, and many other kinds of animation within your videos. Calculating these values as you skip between clips has been slow, but this release promises to make them "magnitudes" faster.

Another significant new feature that will help OpenShot take over the world is its new ability to import and export EDL and XML files, as used by Adobe Premiere and Apple's Final Cut Pro. Both of these formats are close to being industry standards on Windows and macOS, synonymous with Word documents in the world of video editing. EDL, in particular, is used across a huge range of broadcast products, and its support in OpenShot means you can work with different people on the same project across different software platforms, which is what often happens in video production. Finally, the Blender support for title rendering has been aligned with the major 2.8 update to Blender. This allows animated titles to be rendered directly from OpenShot for the best possible quality and flexibility.

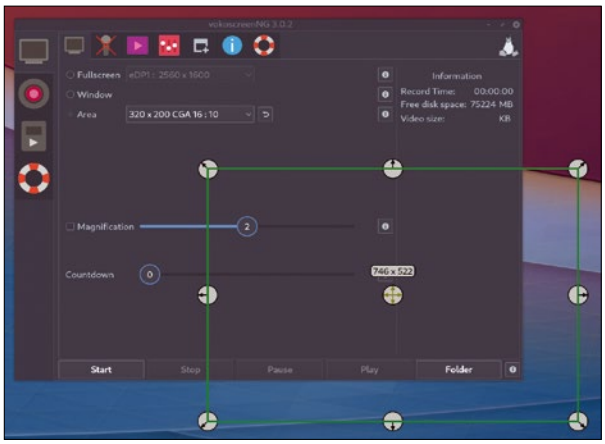
Project Website
https://www.openshot.org

Screen recorder

vokoscreenNG

The mighty OBS Studio is the master of live desktop and game streaming, having become a huge part of the culture driving screencast popularity on YouTube. But that shouldn't stop alternatives from being developed, especially for those of us who don't have the ambition of becoming super-influencers. SimpleScreenRecorder is a great choice for situations where you don't need a huge application to make a simple recording. So too is vokoscreenNG, especially if you need a little more control. VokoscreenNG uses Qt and GStreamer and looks very much like a standard Qt or KDE desktop application, but it actually hides an impressive number of features for such a young project.

One of its best features is apparent from the start, and that's the elastic-edged window you use to select which part of the screen you want to capture. This is clear and accurate, and ideal for only grabbing a section of your display. You can also capture either a single window or the entire screen, with or without magnification, and with a set duration if you want to limit recording time. There's even a movie-themed countdown to help ease you into the recording. Another brilliant feature is the built-in webcam viewer. This opens a window showing the view from an embedded or connected camera so that you can manually move it into the capture area to show some live input. It's not the multi-input you get on OBS, but



If you're looking for a lightweight but powerful alternative to OBS for screen recording, vokoscreenNG is a great option.

it's quick and easy to use. Frame rate is dependent on your hardware, but can theoretically go right up to 144fps. x264 and the MKV container are the default codec and container formats for whatever resolution you choose. Despite the application being in the early stages of (re)development, it already works well and can feel like an integrated part of your desktop – which can't be said for the sometimes overwhelming OBS.

Project Website
https://github.com/vkohaupt/vokoscreenNG

Music tracker

ft2-clone

There's more to the amazing popularity of retro computing than pure nostalgia. While it's certainly fun to revisit your misspent youth playing old 8- and 16-bit games, there's another reason why many of those old genres live on. The limitations in that old hardware meant game developers had to be more imaginative, whether it was color palette hacks, implementing music like code, or repeating tiles in the gameplay mechanic. Many of those techniques have survived the limitations that created them, simply because they made games more enjoyable. The same can also be said for certain types of old applications, and in particular, old applications that created music like code.

The most infamous of these was Ultimate Soundtracker on the Commodore Amiga. It allowed a composer to create four-channel music by entering note data into a kind of scrolling spreadsheet. Each row would represent important parameters such as note, instrument, amplitude, and effect. This data could often be used directly within a game and stored alongside all other game data. The repetitive and algorithmic nature of working like this created a very distinctive sound that you can still hear today, not just in retro gaming titles, but in wider electronic music, IDM, and dance genres. Soundtracker begat ProTracker on the Amiga, and then FastTracker on the

While Amiga ProTracker clones often grab the limelight, the PC FastTracker became equally dominant and is worthy of its own modern interpretation.

quickly dominating PC in the early '90s. Fasttracker II clone (ft2-clone) is a highly accurate recreation of this specific tracker, without needing you to worry about IRQ assignments or DMA access. It comes complete with chunky DOS graphics, pumping visualizers, and an awesome low-fi sound. It will load most of the thousands of XM and MOD music files you can find and let you create music like it's 1999.

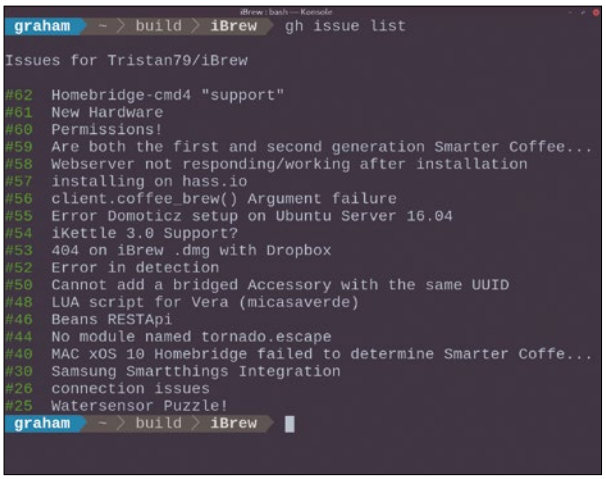
Project Website
https://github.com/8bitbubsy/ft2-clone

GitHub client

GitHub CLI

Many of us use GitHub for our day-to-day work. It is closed source, and it's now owned by Microsoft, but its CEO is also Nat Friedman of Gnome, Ximian SUSE Studio fame, and it's still used by thousands of open source projects. The way most of us interact with GitHub is via its excellent web interface. The web UI is where you typically create a pull request, review and merge other pull requests, and manage issues. But we also still use the `git` command to interact with the typical git processes of cloning projects, checking out branches, and updating the code. This can create an interruption in you workflow as you need to shift from the command line to a web browser, also adding to the temptation of opening Reddit or YouTube. Which is perhaps why GitHub has launched its own official, and open source, command-line client.

After installation, `gh` is the default command to launch the client. When you first use it, a browser session will be spawned to authenticate your account with GitHub and consequently allow the command access to your public and private repositories, as well as general access to other GitHub repositories. Hopefully, this will be the last time you'll need to use the browser, because from the command you can create a pull request (`gh pr create`), check out pull requests (`gh pr checkout <number>`), list issues (`gh issue list`) and view the details for a specific issue (`gh issue view <number>`). You can also check the status of both issues and pull requests. These commands are a huge help if you're already working on the command line, and the checkout functionality in particular lets you skip a whole chunk of copy and pasting from the web interface.



The GitHub CLI makes you more productive by removing a web browser – and all its time-wasting temptations – from your workflow.

Project Website

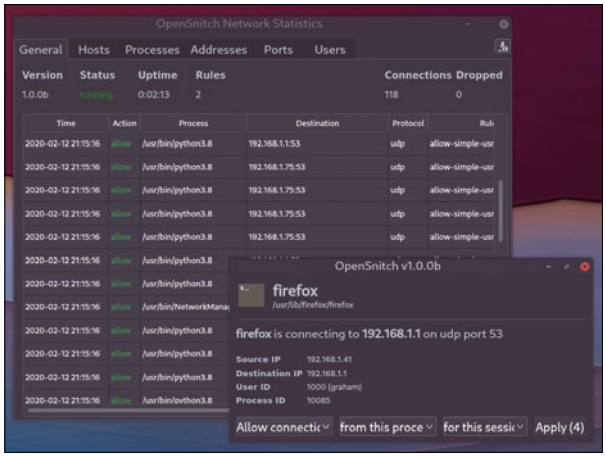
https://github.com/cli/cli

Application firewall

OpenSnitch

Now that almost every process you run requires some kind of network connectivity, whether it asks for it or not, it's become more important than ever to try to monitor what comes in and out of your network. Unfortunately, this user demand hasn't translated into a glut of options. Unless you're prepared to learn the intricacies of a monitoring tool like Wireshark, there has been very little ordinary users can do without resorting to a crude firewall to block entire devices. On macOS, there is a popular option called Little Snitch. This brilliant, proprietary, and paid-for tool will pop-up whenever some wayward application is trying to access some destination without prior approval, letting the user enable or disable network connections on a per-use, per-application basis. It's a utility that's become known as an application firewall.

OpenSnitch is an application firewall that covers much of the same functionality. We even looked at it a couple of years ago, but it has since iterated quickly and become a much more capable, and easier to install, application. Like Little Snitch, whenever a new application requests a new connection in the background, OpenSnitch will show you a small panel that tells you exactly which process is asking for what network resource. You can then choose to allow it through temporarily or permanently. If you don't do anything, a timer will eventually allow access, unless disabled, so that unattended essential processes can send their diagnostics back to base. Alongside these simple Python-drawn panels is an effective UI that shows you every terrifying connection made by your applications. You'll quickly see hundreds, but it's better to know what's happening with your own data than remain in ignorance. When first installed, it does take a while to go through your essential services and applications, but when saved, you can soon forget about them and focus on new and unpredictable connections.



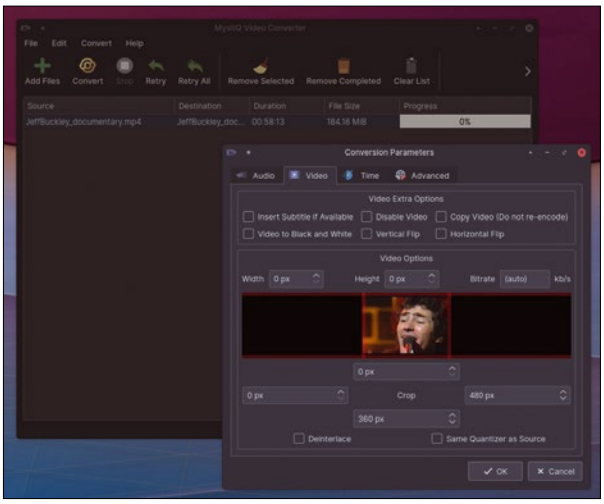
Enable or disable every connection for every application or process on your machine with the wonderful OpenSnitch.

Project Website

https://github.com/evilsocket/opensnitch

Video converter

MystiQ



Even in the age of binge streaming, a little video file conversion is usually a necessity.

There are so many different media converter applications. And yet, none have ever quite perfected the process of transforming a video file from one format to another without having to make so many compromises you usually end up going back to `ffmpeg` on the command line. It's complicated, but it's better than trying to work out what a developer thought was important in their GUI. MystiQ doesn't have this problem because, while its still using `ffmpeg` in the background, it presents very few options to the user, preferring instead to use a preset to cover the broad conversion options. One of the best things about MystiQ is that you can easily queue up files you want to convert, tweak their settings, configure where you want the output to go, and then start the conversion process. The entire batch will be converted according to your wishes.

There are some settings you can change, and these are accessed by right-clicking on a file in your queue and opening *Conversion Parameters*. Audio conversion is actually handled by the *sox* utility, and you can change the sample rate, bit rate, number of channels, and overall volume in the output. The extra video options contain an excellent cropping tool, complete with real-time preview. With this, you can see your video playing and see where you want to cut the edges. You can also add subtitles, convert to black and white, and flip horizontally or vertically. A time tab even lets you adjust the start and end points, as well as the overall playback speed, which is ideal for learning Jimmy Page guitar solos. And if all else fails, an *Advanced* tab enables you to enter the raw `ffmpeg` arguments, for when you really can't dial in the best conversion.

Project Website

https://github.com/llamaret/MystiQ

Unofficial Spotify client

Spotifyd

Turn any embedded Linux system into a high-quality streaming server with unlimited access to music.

For those of us cynical about proprietary IoT devices, there was some schadenfreude to enjoy recently. The expensive music streaming speaker manufacturer, Sonos, bribed its users with a 30 percent discount if they went through the process of enabling *recycling mode* on their old equipment. Enabling this effectively disabled perfectly usable devices, prohibiting their resale and secondhand use and effectively condemning them to the landfill. Disabling perfectly usable hardware is audacious, but yet another example of us having no control over the hardware we're effectively licensing the use of (see Logitech Harmony, Philips Hue, and Google Nest for other prime contenders). Which is why it's so important to find alternatives that don't compromise on quality or convenience.

Fortunately, when it comes to streaming music, there are plenty of options. There are simple Bluetooth devices you can wire into your preexisting system, and something like a Raspberry Pi with a decent USB audio interface or DAC is another good option. But when it comes to software, there's nothing that can really compete with Spotify, which is why it's one of the best reasons for using a proprietary piece of hardware.

There is an official Linux client, and in the past you could use a plugin with Logitech's SqueezeBox platform, but spotifyd is an even better option. Spotifyd is a background daemon that can run on any Linux box, effectively turning it into a sink for your Spotify client. After it's installed, it requires no further configuration if you're on the same network. It appears as a playback device from your official Spotify app, just like a Sonos speaker or other official destination. You can then remotely control the music streaming to your Linux box, or turn any Raspberry Pi into a superior smart speaker.

Project Website

https://github.com/Spotifyd

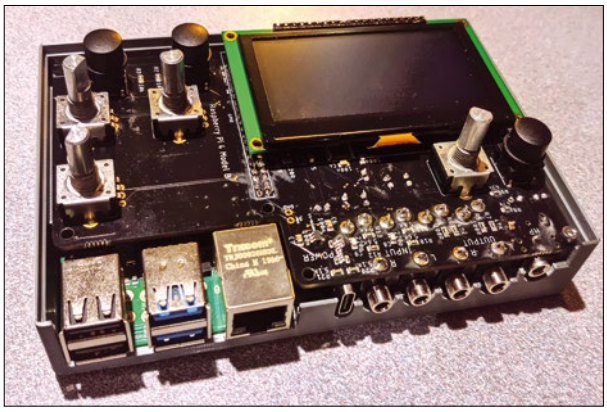
Music platform

norns

In Norse mythology, the Norns are female entities who control the destinies of gods and men, and their association with fate and destiny is perhaps why this project shares the same name. Norns is a completely open platform for music performance and abstract audio composition that also teaches you how to code. It's open source and built around Raspberry Pi hardware. While there's a commercial version that you can spend a lot of money on, there are affordable shields for your own Raspberry Pi and circuits you can easily build yourself. One of them is even called Fates. This hardware takes a specific form factor – an OLED display, three switches, and three rotary controllers, all running on a specific stack of software that provides audio input and output, MIDI and OSC communication, an audio mixer, and built-in effects, as well as a web-based coding, library, and development platform.

The norns platform is difficult to summarize, because it isn't a single thing. If you run the software, you can effortlessly install and run a huge variety of open source synths, sound effects, loopers, sequencers, and controllers, all built around a beautifully minimal input aesthetic. Many replicate the functionality of specific Eurorack modules, or previously bespoke software, such as MLR, which live samples and sequences loops of audio. If you own, or have built, accompanying grids and rotary controllers, these can often be used to expand on the small screen and limited input, and the project is expanding to include generic Linux platforms and Raspbian-based installations. You can do all this without touching the code, and many people do.

While the project itself was initiated and designed primarily by Brian Crabtree of Monome fame (also known as tehn), it's the community that has shaped and expanded upon the original idea to create the current flourishing ecosystem. This community, known as "lines," can be found at *https://llllllll.co*, and it's here that many of the scripts, engines, and experiments that run on norns originate. These are installed and run on your hardware using a built-in package manager called Maiden that also acts as an IDE for modifying these scripts or creating your own. There's a brilliant onboarding guide that steps you through coding your own projects, and the hardware itself hosts the interactive shell and editing environment where you can create projects, edit those already installed, and see and hear the effects instantly. The default scripting engine is Lua, which is both easy to get started with and ideal for processing audio and music-related data. The online editor has syntax highlighting and an interactive



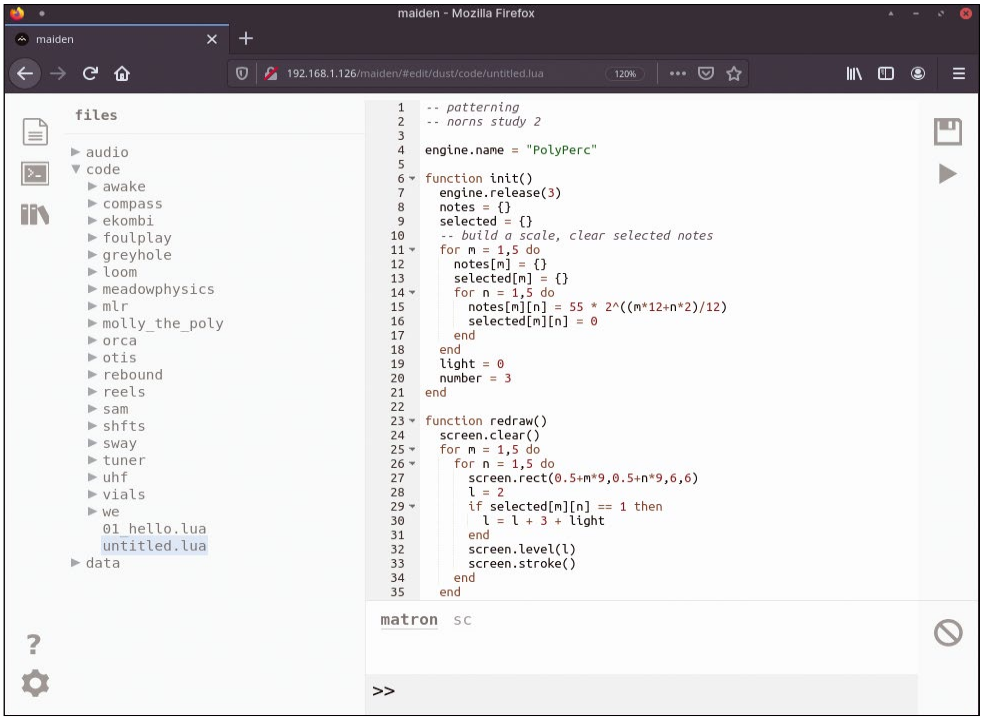
While norns is a Raspberry Pi-based hardware platform, it also runs on Linux.

interpreter so you can experiment with the API, which is both well designed and well documented.

The background DSP engine is written using SuperCollider. Because the entire stack is modular, you can swap-out one module for another, or swap a module with your own libraries and DSP code. There are polyphonic and mono synth engines, samplers, and compressors, all of which can be imported into your own project with just a line or two of code. You can then easily write your own routines for sequencing notes, or for setting your own scales and intervals or hardware modifier. It's an amazing system that offers unlimited possibilities and almost singlehandedly proves that Linux is finally good for music production.

Project Website

https://github.com/monome/norns



The best thing about norns is that you can use the interactive Maiden package manager to install apps and edit their source code.

Console space shooter

Terminal Phase

If you're a beleaguered sys admin, you can play terminal games like Terminal Phase within tmux to quickly switch panes if you need to look like you're working.

Sometimes, the simplest games are still the best. A great example of this is Scramble, which is an early arcade classic developed by Konami way back in 1981. It was a shoot-em-up, where the player flew across a horizontal landscape that morphed from a city, to mountains, and back through a cave system. You needed to shoot the baddies whilst simultaneously dropping bombs on the land-based threats and collecting fuel to keep your ship flying. The best part was when you were in the cave system and you needed to move up and down, left and right, at the same time. The arcade version wasn't simple, for the time, but its gameplay successfully made it to 8-bit home computers almost untouched. Most impressive was the LCD handheld version of the game. At a time when Nintendo was creating LCD games with fixed shapes and configurations for each level, Scramble's scrolling landscape was built from a never-ending grid that effectively gave you an infinite amount of variety, and an infinite challenge.

If it's the right game, simple playability can also survive a move to the command line. Ascii Patrol is a brilliant example because it successfully recreates the arcade classic Moon Patrol in your terminal with simple character-based graphics. Terminal Phase has just become another great example, with a gameplay mechanic that's best described as a blend between Galaxians and Scramble. In true command-line style, your spacecraft is nothing more than the greater-than symbol, and you fly across a landscape made from hashes – much like the squares in that early LCD interpretation of Scramble. There are even the same tunnels, although the enemies seem to move considerably faster than the original. Either way, it's addictive, and a great way of taking a break from your normal terminal processes without having to switch context to the desktop.

Project Website
https://gitlab.com/dustyweb/terminal-phase

MMO

Isleward

Isleward can be played online, through a browser on hosted on your local machine, and even with an Android app. And it's open source!

There aren't too many open source, online, massive multiplayer games for Linux. But Isleward is one. With some wonderfully enticing 16-bit style graphics, and a top-down retro Rogue-like exploration gameplay style, it's a difficult game to resist. And you can instantly play it, for free, because the developer hosts a server (*https://play.isleward.com/*) where you can simply create an account and start playing. However, unlike almost every other game we can think of with an enticingly free account creation and play portal, you're also completely free to download the code to Isleward and run it yourself from your own Linux machine. It might be a little harder to get friends to join, but you'll stay in control of your own data, and of course, you're free to make and contribute whatever modifications you wish to the game engine.

Like nearly every RPG, you start the game by creating a character – first by giving it a name, and then a "spirit." There are currently three to choose from: an owl for spellcasters, a bear for physical types, and a lynx for the nimble and dextrous. After this, you enter the game world, and there's a small introductory tutorial to ease you into the controls and principle mechanics. You move with the *WASD* keys, and you can map out a path quickly while your character is in motion. Certain locations, such as the inside of an Inn, are occluded until you enter them. A pane in the lower part of the game window gives you a "nethack" like description of your surroundings, which is helpful as you often can't work out what the pixels represent. If you need to attack anything, you select it with the cursor and press Space to auto-attack when the target is in range, which usually means when it's in the adjacent square. All of this is played out with other people also in the game, which adds to that old-school feeling, familiar to anyone who played early Ultima games. There's still a lot of development needed, but it's already a lot of fun, and thanks to being open source, anyone can contribute.

Project Website
https://gitlab.com/Isleward/isleward

From disk to paper

Scripted Printing

A few commands and some simple shell scripts make it easier to manage your printer so that you can access print functions quickly and automate recurring tasks.

BY HARALD ZISLER

If you work with LibreOffice or an image processing program like Gimp, you don't have to look too hard for the print function. The print icon is usually located in the upper left corner of the buttonbar; alternatively, you can press Ctrl+P. In many situations, however, it would be more practical to print without the help of an application – for example, if you want to print from a script.

Complex printing commands can also be transferred as shell commands. There are instructions to fit several pages on one sheet, for duplex printing, for cover sheets to make sorting easier, or for options to change the page orientation.

Linux basically comes with two commands for controlling printers at the command line, `lp` and `lpr`. Table 1 shows some important options, while Table 2 lists some helpful variants for everyday use. For additional options and settings, check out the extensive man pages for `lp` [1] and `lpr` [2].

Displaying the Queue

The status of the currently pending print tasks can be displayed using the `lpq` or `lpstat` commands. If executed without any further options, both commands display the queue for the default printer. Optionally, use the `-P <printer>` option to specify the desired printer.

Table 1: Print Commands

Action	lp	lpr
Output to default printer	`lp <file>`	`lpr <file>`
Output with printer definition	`lp -d <printer> <file>`	`lpr -P <printer> <file>`
Number of copies (max. 100)	`lp -n <count> [...]`	`lpr -# <count> [...]`
Print without filter	`lp -o raw [...]`	`lpr -o raw [...]`
Pages to print	`lp -P <Pages> [...]`	

Table 2: Print Options

Action	Input	Note
Paper size	`-o media=<format>`	For example, `a3`, `a4`, `a5` ...
Landscape	`-o landscape`	
Single-sided printing	`-o sides=one-sided`	
Duplex printing	`-o sides=two-sided-long-edge`	Flip on long edge
Duplex printing	`-o sides=two-sided-short-edge`	Flip on short edge
Fit to page	`-o fit-to-page`	
Group multiple pages on one sheet	`-o number-up=<number>`	Supported values: `2`, `4`, `6`, `9`, `16`

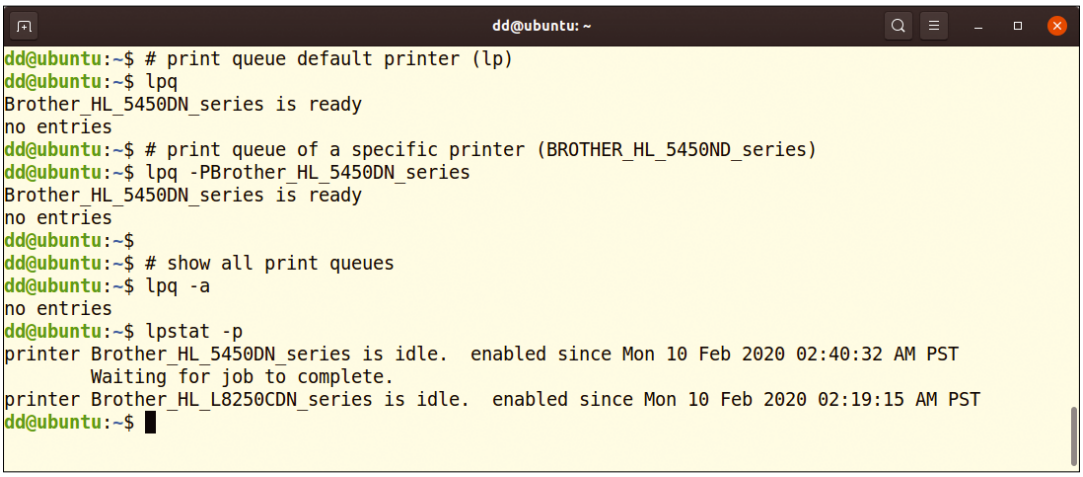


Figure 1: There are many approaches for displaying the queued jobs.

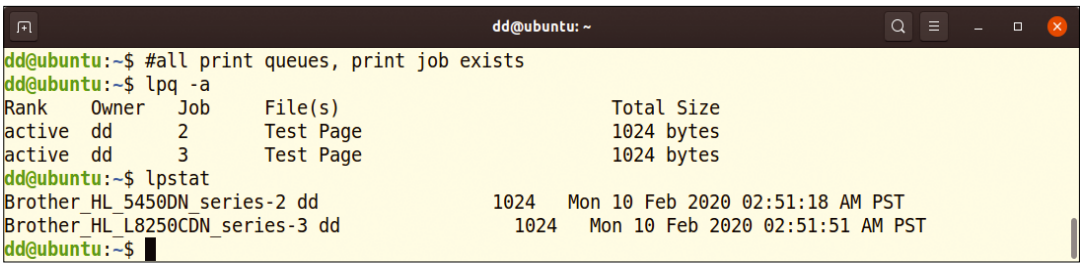


Figure 2: The number appended to the printer name with a hyphen corresponds to the number of the print job.

All printers and their current status can be obtained by typing `lpstat -a`, `lpstat -o`, or `lpq -a` (Figure 1). The `lpstat -t` command provides a comprehensive overview of printers, queues, and jobs.

If print jobs are available, you will see output like that shown in Figure 2. Among other things, you will find the job number with which you can manage a print job in the queue, if needed. For various actions you have to extract the print job number from the output.

You can use the command from Listing 1 to filter the number of the current print job from the status report and then use the number to obtain additional information about the print request, delete the associated request, or move it to another printer.

Managing Print Jobs

To cancel a print job, type `lprm <job number>` or `cancel <job number>`. You can extract the job number from the queue display by typing `lpq -a`. To cancel a print job, you either have to be the owner of the print job or have appropriate administrative rights. Regardless of which printer, `cancel -a` deletes all the current print jobs. This is why it makes more sense to specify the printer with the option. In Figure 3, `lprm` is used to delete one of the existing print jobs.

If a printer fails during operation, but you do not want to interrupt the print job, you can move it to another device using `lpmove` – provided you have the appropriate rights. You can use this command in the form `lpmove <Job number> <New printer>` or for all print jobs of a printer with `lpmove <Old`

Listing 1: Print Job Number

```
$ lpq -al | grep job | cut -d\[ -f2 | cut -d ' ' -f2
```

Figure 3: If you send your 100-page master's thesis to the wrong printer, you can cancel the job at the command line.

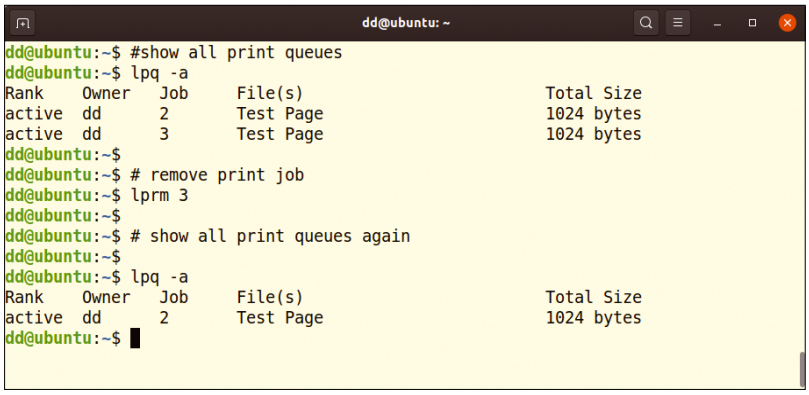
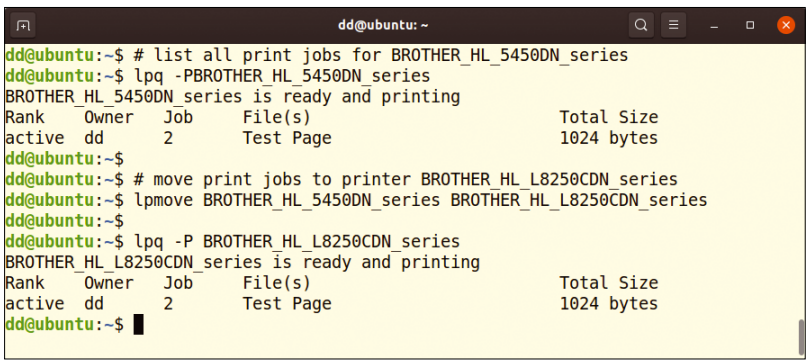


Figure 4: If a printer unexpectedly fails, you can reroute the print jobs to a working printer.



Listing 2: Print Job Details

```
01 #!/bin/bash
02
03 djobs () {
04   clear
05   echo "Current print jobs:"
06   echo "-------------------------------------------------------------------"
07   lpq -a
08   echo "-------------------------------------------------------------------"
09   echo "Select print job: "
10   dj=$(lpq -al | grep job | cut -d\[ -f2 | cut -d ' ' -f2 | smenu -n10 -t1 )
11
12   echo "Selected print job: $dj"
13   lpq -a $dj
14   echo "-------------------------------------------------------------------"
15   action=$(echo "Nothing cancel" | smenu -m "Select action" )
16
17   if [ "$action" = "Cancel" ]; then
18     lprm $dj
19     if [ $? -eq 0 ]; then
20       echo "Print job $dj deleted"
21     fi
22     sleep 2
23   fi
24   exit 0
25 }
26 djobs
```

Listing 3: Select Printer

```
#!/bin/bash
prin () {
  clear
  target=$(/usr/sbin/lpc status all | grep \: | tr -d \: | smenu -n3 -c
    -m "Select a printer:")
  # Example: Queue output
  lpq -P$target
  exit 0
}
prin
```

Figure 5: The `djobs()` function from Listing 2 wraps what are often cryptic printing commands in simple dialogs.

printer> <New printer>. Figure 4 shows how to move print jobs from one printer to another.

Scripted Jobs

Together with some other shell functions, the commands presented here can be assembled to create a script that prints details about a print job and deletes it at the push of a button if necessary (Listing 2). If you want to use the routine as a function in your own shell scripts, cut the `djobs () { [...] }` function and paste it at the start of your own program.

With a little help from the smenu [3] and YAD tools, it is quite easy to implement selection dialogs. The smenu command-line tool is recommended for command-line wizards and users who need to work on a remote computer via SSH. YAD on the other hand is an option for users who prefer the comfort of the desktop environment. The program displays windows on the screen with simple instructions.

Smenu and YAD are missing from the default software selection in most distributions. But, thanks to the *smenu* and *yad* packages, the two programs can be installed quickly from the package sources.

In smenu, the `-m <Title>` option shows the user what they are selecting and, if necessary, why. Without further options, you would select the desired entry word by word within a line. `-n <number>` limits the selection lines; `t1` tells smenu to display the selection line by line in a single column. Figure 5 shows the flow of the script.

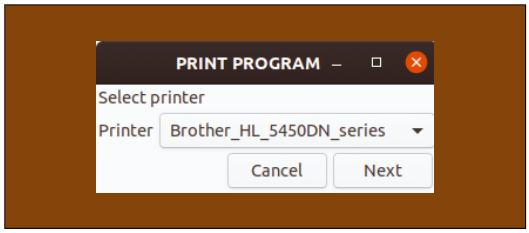
The script from Listing 3 is recommended for a terminal session. It selects the printer for the print output, the queue display, or a queue you want to move. The example in Figure 6 lists the print jobs for a specific printer.

YAD is a good choice if you are looking for a graphic alternative. Without too much programming work, you can use this tool to create simple applications based on a shell script. The example in Listing 4 shows how to select a printer and the file to be printed.

The `yad` calls in lines 12, 22, and 23 first ask for the desired printer (Figure 7), then open a dialog for file selection (Figure 8), and finally show a confirmation.

Figure 6: The `prin()` function from Listing 3 lists all the available printers and outputs the print queue for the selected device.

Figure 7: A Bash script can also be used to implement programs with graphical dialogs as in Listing 4.



Scripted Printing

Constantly recurring tasks, such as a database query, can be easily automated using a shell script. However, the command-line tools usually output the results without printable formatting.

With a little help from the `enscript` command, the output can be processed quite easily. `enscript` writes the result directly to a PostScript (PS) file and also supports printing the output content. This is used in the sample script from Listing 5.

However, it is important to note that Enscript cannot handle UTF-8 encoded files and pipes. You need to convert the data to the desired character set in advance using `recode`. Both programs work both in a pipe and with files. If necessary, the PS files can also be converted to PDF format using `Ps2pdf14`.

The sample script in Listing 5 still offers plenty of scope for improvements and your own ideas. The

Figure 8: The YAD command-line tool supports selection lists, file dialogs, and simple buttons.

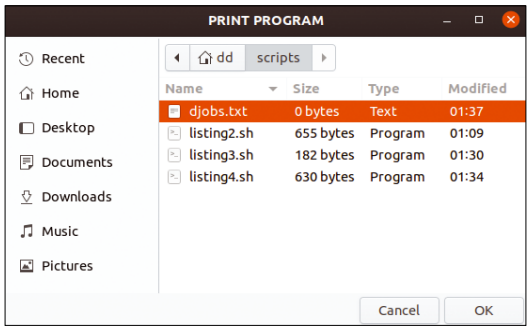


Figure 9: Listing 5 automates querying a PostgreSQL database. The script first prints the results directly and then writes a PDF file.

```
dd@ubuntu:~/scripts$ # query with printout
dd@ubuntu:~/scripts$ ./listing5.sh
Print PDF
Select printer:

Brother_HL_5450DN_series
Brother_HL_L8250CDN_series
CUPS-BRF-Printer
[ 0 pages * 1 copy ] sent to Brother_HL_5450DN_series
dd@ubuntu:~/scripts$
dd@ubuntu:~/scripts$ # query with PDF file
dd@ubuntu:~/scripts$ ./listing5.sh
Print PDF
[ 0 pages * 1 copy ] left in partlist.ps
removed 'partlist.ps'
Query stored in partlist.pdf
dd@ubuntu:~/scripts$
```

Listing 4: Select Printer and File

```
01 #! /bin/sh
02 prin {
03   clear
04   # Read printers
05   z=0
06   pline=""
07   for i in $(/usr/sbin/lpc status all | grep : | tr -d \: ); do
08     pline=$(echo $pline$i!)
09   done
10
11   # Menu item with Yad
12   printer=$(yad --title="PRINT PROGRAM" --text="Select printer" --form \
13   --field="Printer":CB $pline \
14   --button="Cancel":1 \
15   --button="Next":2)
16   if [ $? -eq 1 ]; then
17     exit
18   fi
19   printer=$(echo $printer | tr -d \| )
20
21   # Print sample file
22   file=$(yad --title="PRINT PROGRAM" --file)
23   yad --title="PRINT PROGRAM" --text="Print selected file $file?" --yesno
24   if [ $? -eq 0 ]; then
25     lpr -P$printer $file
26   fi
27   exit 0
28 }
29 prin
```

Listing 5: Process Output

```
01 #!/bin/bash
02 # Database query (PostgreSQL) with print preparation
03
04 # Select PDF file or print
05 approach=$(echo "Print PDF" | smenu )
06 if [ "$approach" = "Print" ]; then
07   # Select printer
08   target=$(/usr/sbin/lpc status all | grep \: | tr -d \: |
     smenu -n3 -c -m "Choose a printer:")
09   # Database query, character set conversion, print preparation and
       printing
10   psql -P border=3 -c "select * from parts;" |
     recode UTF8..ISO-8859-15 | enscript -H1 --highlight-bar-gray=0.8
     -fCourierBold10 -P$target
11 elif [ "$approach" = "PDF" ]; then
12   # Database query, character set conversion, generate PS
13   psql -P border=3 -c "select * from parts;" |
     recode UTF8..ISO-8859-15 | enscript -H1 --highlight-bar-gray=0.8
     -fCourierBold10 -o partlist.ps
14   # Convert to PDF file
15   ps2pdf14 partlist.ps
16   # Delete PS file
17   rm -v partlist.ps
18   echo "Query saved in partlist.pdf"
19 fi
```

Source Code in Color

Text editors intended for programming usually color highlight commands, variables, or instructions; this makes it far easier to keep track of the source code. `enscript` also offers this kind of function with the `-E<Language>` option. A list of all supported schemas can be obtained with the `enscript --help-highlight` or `enscript --help-pretty-print` commands, depending on the version. For example, the command

```
enscript -H1 --highlight-bar-gray=08 -fCourierBold10 --color -Ebash -o Source.sh.ps Source.sh
```

creates a colored image of the `Source.sh` shell script (Figure 11). Before doing this, the correct character set again had to be set with `recode`.

process flow is shown in Figure 9, and the database query's output is shown in Figure 10. The script is primarily intended to demonstrate how little effort it takes to solve even very complex tasks. Compared to the clicks required with a database client from an office package, the terminal script saves a huge amount of work.

Preparing the Output

The call to `enscript` (lines 10 and 13 of Listing 5) can be adapted to suit your own needs, if required. For example, you can opt to print in landscape format, specify the type and size of the font, and even output source code with syntax highlighting (see the "Source Code in Color" box). Some of the corresponding options are listed in Table 3.

For example, it is often necessary to avoid line breaks. If the lines are too long, it helps to use a smaller font or print in landscape format. This problem can also be solved by scripting. The `wc -L` command lets you determine the length of the longest occurring line. You can then use the value obtained in this way as a

Figure 10: This is how the database query leaves the printer. The `enscript` command-line tool prepares the results.

```
                Wed Oct 30 19:02:15 2019        1
+---------+--------------------+--------+
|   des   |        type        | number |
+---------+--------------------+--------+
| 2N3055  | Transistor         |   5.00 |
| BC107A  | Transistor         |  20.00 |
| IE500   | Diode ring mixer   |   2.00 |
| 1N4001  | Diode              |  30.00 |
| TCA440  | IC AM receiver     |   3.00 |
+---------+--------------------+--------+
(5 Rows)
```

Figure 11: Optionally, the Enscript output can also use syntax highlighting, to improve the source code printout.

```
Quelltext.sh        Wed Oct 30 22:45:16 2019        1
#! /bin/bash
name=$USER
clear

# Data entry
read -ei $name -p "Name: " name

# Evaluation
if [ ! "$name" = "$USER" ];
then
echo "DifferentName"
fi
```

Listing 6: Choose Page Orientation

```
01 #!/bin/bash
02
03 # File selection
04 file=$(ls -1 | smenu -n 10 -t 4)
05
06 # maximum line length
07 mz=$(cat $file | wc -L)
08
09 # Portrait up to 80 characters, landscape above this
10 if [ $mz -lt 80 ]; then
11         cat $file | recode UTF8..ISO-8859-15 | enscript -H1 --highlight-bar-gray=0.8
             -fCourierBold10 -o $file.ps
12 elif [ $mz -gt 80 ]; then
13         cat $file | recode UTF8..ISO-8859-15 | enscript -r -H1 --highlight-bar-gray=0.8
             -fCourierBold10 -o $file.ps
14 fi
15
16 [... Print commands, PDF conversion ...]
```

criterion for determining the font size and page orientation:

- Up to 80 characters: 10/12pt font size
- 80-132 characters: 8pt font size or 10/12pt and landscape format.
- 132 characters or more: 8/10pt font size and landscape orientation.

Listing 6 shows a sub-script that uses `wc` to determine the longest line (Line 7) and then tells `enscript` to print in landscape mode or leave it in portrait mode with the conventional orientation (`if` loop starting at line 10).

The Author

Harald Zisler has been working with FreeBSD and Linux since the early 1990s. He writes magazine articles and books on technical and IT topics.

Conclusions

Even in the shell and in scripts, you do not have to do without the convenience that graphical interfaces offer when printing. Printer selection, queue management, and the creation of attractive print output can be easily integrated into your shell scripts. And you don't have to reinvent the wheel or learn programming. Simple shell scripts and practical command line tools take much of the work off your hands. ■■■

Info

[1] lp manpage: *http://manpages.org/lp*

[2] lpr manpage: *http://manpages.org/lpr*

[3] "Create a select menu with smenu" by Harald Zisler, *Linux Magazine*, Issue 205, December 2017, p. 32, *http://www.linux-magazine.com/Issues/2017/205/smenu*

Table 3: enscript Options

Action	Option	Note
Column specification	`-<Number>`	
Specification of the pages to be printed	`-a<FirstPage>-<LastPage>`	
Print odd pages	`-a odd`	
Print even pages	`-a even`	
Suppress page header	`-B`	
Suppress job header	`-h`	
Curtail over-length lines	`-c`	
Specify printer	`-d <printer>`	Also possible, `-P <printer>`
Duplex printing	`-DDuplex`	
Syntax highlighting	`-E<language>`	Output overview with `enscript --help-highlight` (in newer versions also `enscript --help-pretty-print`)
Text font	`-f<font size>`	
Header font	`-F<font size>`	
Reading lines	`-H<number>`	`<number>` outputs the frequency of reading lines; `-H1` results in a "zebra crossing" with alternating light and dark lines
Specify grayscale for reading lines	`--highlight-bar-gray=<value>`	
Title	`-t`	
Multiple copy	`-n <number>`	
Output file	`-o <file>`	`-p <file>` also possible
Landscape format	`-r`	
Footer	`-u<Text>`	
Multiple logical pages per page	`-U <number>`	`<number>` must be 2 or a multiple of it

■■■

A LÖVE animation primer
I <3 Animation

LÖVE is an extension of the Lua language, designed to make developing games easy. In this tutorial, we'll explore this framework by creating some animated sprites. BY PAUL BROWN

The topic of game design can (and does) fill entire books, so we aren't going to tackle all of that today. Instead, we'll focus on a specific project, creating animated sprites in LÖVE [1], a Lua-based [2] framework that provides you with hundreds of tools and a coherent template for creating 2D games. By creating a simple animation cycle and having LÖVE show it in a window, we will explore the fundamentals of LÖVE and help you start thinking of the games you can create with the platform.

Running LÖVE

You can download LÖVE from the project's main page or use your software manager, as it is included with most mainstream distros. The current version is LÖVE 11.3.

When you run the LÖVE interpreter for the first time from the command line with `love`, you will initially see a screen that says "no game."

To actually see a game, you can feed the `love` command a directory like:

```
love path/to/love/project/
```

The directory must contain a file called `main.lua`, which is what the interpreter will try to run.

Listing 1: Basic LÖVE

```
01 function love.load ()
02   love.graphics.setBackgroundColor (0.5, 0.8, 1, 1)
03 end
04
05 function love.update ()
06 end
07
08 function love.draw ()
09 end
```

Anatomy of LÖVE

As the wiki [3] explains in the documentation, a LÖVE program typically consists of three parts: the `load`, `update`, and `draw` functions, as seen in Listing 1.

The `love.load` function (lines 1 through 3) is where you set up things. You load images, set the background, calculate the frames in each animation, set the initial values of variables, create objects, and so on. In this case, all we're doing is changing the background color of the playing field from the default black to a lighter blue to make it easier to see our animation.

The `setBackgroundColor ()` method is a LÖVE graphics function that takes four parameters between `0` and `1` indicating the degree of red, green, blue, and opacity of the background. So `0, 0, 0, 0` would be completely transparent black, `0.5, 0, 0.5, 0.5` would be a semi-transparent purple, and `1, 1, 1, 1` would be a pure opaque white. As mentioned above, `0.5, 0.8, 1, 1` is an opaque light blue.

`love.update` (lines 5 through 6) is the main loop of the game, and where things change as the game progresses. Here you calculate the new coordinates for sprites; read in keystrokes, mouse

Listing 2: All Frames

```
01 function love.load ()
02   love.graphics.setBackgroundColor(0.5, 0.8, 1, 1)
03
04   reel = love.graphics.newImage ("images/cmcf.png")
05 end
06
07 function love.update ()
08 end
09
10 function love.draw ()
11   love.graphics.draw (reel, 100, 100)
12 end
```

movements, or other player-generated input; modify the playing field, and so on. When animating a character as we are doing today, this is where you would calculate which animation frame to show at each moment.

`love.draw` (lines 8 through 9) is where you draw what will be seen on the screen after each iteration in `love.update`. In today's project, that will be the actual frame of the animation we calculated.

Loaded LÖVE

For today's project, we will first load the image with the animation frames. I made a simple character called Cubey McCubeFace and gave him a simple (and probably anatomically incorrect) walk cycle for my animation (Figure 1), which is saved as `cmcf.png`.

As you can see, there are six frames in the animation and each frame is a 64x64 pixel square. Listing 2 shows how you would display the whole image in a LÖVE game window.

On line 4, you use LÖVE's `newImage ()` function to load an image from the filesystem. I'm calling the object *reel*, because it is like a reel of film from the olden days, containing multiple frames. On line 11, we use LÖVE's `draw ()` function to draw the image at the `100, 100` position in the game window. Easy.

Run the program and it will show what you see in Figure 2.

Fortunately, grabbing cropped parts from an image and displaying them is not much harder in LÖVE thanks to something called *quads*.

A quad stores the coordinates of a chunk of a larger image, so it can be displayed. Listing 3 is an example of how to use a quad.

As you can see on line 5, you make a quad by passing LÖVE's `newQuad ()` function the coordinates of the upper left hand corner of the quad you want and then the width and height of the chunk you want to cut out. In this case, we are taking the first 64x64 square from our reel (Figure 3). The `newQuad ()` function also needs the dimensions of the whole image as a reference.

A quad is only the coordinates and dimensions of part of a drawable, that is, something that can be drawn in the `draw` section of your program.

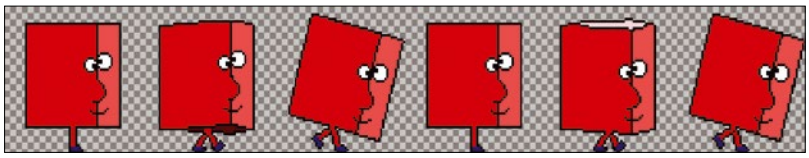


Figure 1: Cubey McCubeFace's walk cycle.

When you go to draw the cropped image (line 12), you need to tell the interpreter what drawable (in this case `reel`) the quad is referring to.

Also in Listing 3, notice the colon on line 5. A colon tells the function to use the object before the colon as the argument, much like the keyword `self` is used in object-oriented languages. In other words, it tells `getDimensions ()` that it must return `reel`'s width and height.

You could have hard-coded the image's dimensions in here like so:

```
frame = love.graphics.newQuad ➥
  (0, 0, 64, 64, 384, 64)
```

But then the code would only work for images of that exact size.

Splitting Up

What you want to do is not only show one frame, but split the `reel` into its individual frames and store each frame (or, to be precise the quads that

Listing 3: One Quad

```
01 function love.load ()
02   love.graphics.setBackgroundColor(0.5, 0.8, 1, 1)
03
04   reel = love.graphics.newImage ("images/cmcf.png")
05   frame = love.graphics.newQuad (0, 0, 64, 64, reel:getDimensions())
06 end
07
08 function love.update ()
09 end
10
11 function love.draw ()
12   love.graphics.draw (reel, frame, 100, 100)
13 end
```

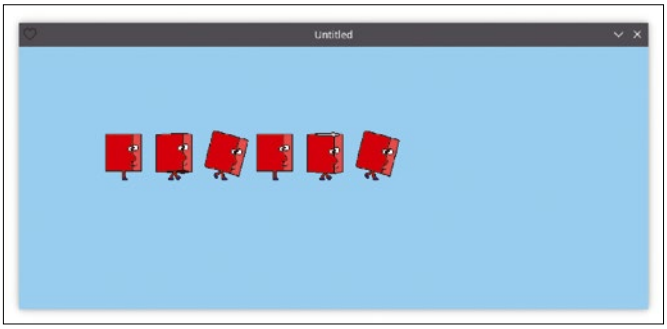


Figure 2: Drawing an image in the game's window is simple.

Figure 3: Use quads to cut out chunks from larger images.

Listing 4: Separate Frames

```
01 function love.load ()
02   love.graphics.setBackgroundColor(0.5, 0.8, 1, 1)
03
04   reel = love.graphics.newImage ("images/cmcf.png")
05
06   frames = {}
07   for i = 0, reel:getWidth() - 64, 64 do
08     table.insert (frames, love.graphics.newQuad
                  (i, 0, 64, 64, reel:getDimensions ()))
09   end
10 end
11
12 function love.update ()
13 end
14
15 function love.draw ()
16   love.graphics.draw (reel, frames[5], 100, 100)
17 end
```

describe each frame) in some sort of array. This will let you pick the one you want at each moment. Fortunately, arrays, or, more specifically, tables, are a cornerstone of the Lua programming language.

Look at Listing 4 and notice how you read each frame/quad into the `frames` table on lines 6 to 9.

On line 6 you make `frames` a table by assigning it an empty array. Then on line 7, you start a `for` loop to fill up the table. In a similar way to C/C++ and JavaScript, `for` loops in Lua take three arguments: the initial value for the iterator (`0`), the end value for the iterator (`reel:getWidth () - 64`) and the increment (`64`) you add to the iterator after each loop.

Here's how this goes: `i` starts out as `0`, and on line 8 you use the Lua table method `insert` to dump the quad that goes from `(0, 0)` to `(63, 63)` into the `frames` table. Next time around, `i`'s value is `64` and the quad you dump is from `(64, 0)` to `(127, 63)`. This goes on until you reach the left side of the last frame at `(320, 0)` and you dump from there to `(383, 63)` into the quad table, covering all six frames.

Listing 5: Cubey McCubeFace Walks

```
01 local framecounter = 1
02
03 function love.load ()
04   love.graphics.setBackgroundColor(0.5, 0.8, 1, 1)
05
06   reel = love.graphics.newImage ("images/cmcf.png")
07
08   frames = {}
09   for i = 0, reel:getWidth() - 64, 64 do
10     table.insert (frames, love.graphics.newQuad (i, 0, 64,
                  64, reel:getDimensions()))
11   end
12 end
13
14 function love.update ()
15   framecounter = framecounter + 1
16   if framecounter > 6 then
17     framecounter =  1
18   end
19 end
20
21 function love.draw ()
22   love.graphics.draw (reel, frames[framecounter], 100, 100)
23 end
```

Listing 6: Cubey McCubeFace Walks Slow

```
01 local framecounter = 1
02 local time = 0
03
04 function love.load ()
05   love.graphics.setBackgroundColor(0.5, 0.8, 1, 1)
06
07   reel = love.graphics.newImage ("images/cmcf.png")
08
09   frames = {}
10   for i = 0, reel:getWidth() - 64, 64 do
11     table.insert (frames, love.graphics.newQuad
                  (i, 0, 64, 64, reel:getDimensions()))
12   end
13 end
14
15 function love.update (dt)
16   time = time + dt
17   if time >= 0.5 then
18     time = 0
19   end
20
21   framecounter = math.floor ((time/0.5) * #frames) + 1
22 end
23
24 function love.draw ()
25   love.graphics.draw (reel, frames[framecounter], 100,
                  100)
26 end
```

You can now use an index to refer to any of the `frames` and show it in the game window, which is exactly what you do on line 16.

To run through the frames and animate your sprite, you now use the `update` part of your program – see Listing 5.

The new thing here is that we have a variable, `framecounter`, that does what it says on the tin: It starts out as `1` (line 1) and increments to `6` on each iteration of `update ()` (lines 15 to 18). You can then use it to show each of the frames in the `draw ()` section (line 22).

Run the program and Cubey McCubeFace walks … but badly.

LÖVE in Time

When you run Listing 5, Cubey "walks," but the movement is jittery and way too fast to appreciate as a smooth animation.

That is because you are using the speed of your processors to run the animation. Depending on the load of your CPU, the execution of your program will be faster or slower and, in consequence, so will your animation. The speed of your animation will also vary depending on the speed of the device you use. This is not good. You cannot have your game run so fast that it's unplayable or so slow that it's sluggish and boring.

What you need is to run the animation in game time. Game time means that, if you want an action to take half a second to complete, it will take half a second regardless of the load of the processors or the power of the machine it is running on.

Game time is implemented in LÖVE via the `dt` variable. The idea behind `dt` is simple: It contains the time that has passed since the last time `update` was called. Just with that informa-

Listing 7: main.lua

```
01 require 'animation'
02 local time = 0
03
04 function love.load ()
05   love.graphics.setBackgroundColor(0.5, 0.8, 1, 1)
06
07   cmcf = {}
08   cmcf.walk = animation ("images/cmcf.png", {64, 64}, 0.5, {100, 100})
09 end
10
11 function love.update (dt)
12   cmcf.walk.frame, cmcf.walk.time = framecounter (cmcf.walk.time + dt, cmcf.walk.duration)
13 end
14
15 function love.draw ()
16   love.graphics.draw (cmcf.walk.reel, cmcf.walk.frames[cmcf.walk.frame], unpack (cmcf.walk.pos))
17 end
```

Listing 8: animation.lua

```
01 function animation (reel, framesize, duration, pos)
02   local animation = {}
03
04   animation.reel = love.graphics.newImage (reel)
05
06   animation.frames = {}
07   for i = 0, animation.reel:getWidth() - framesize[1], framesize [1] do
08     table.insert (animation.frames, love.graphics.newQuad (i, 0, framesize[1], framesize[2],
                    animation.reel:getDimensions()))
09   end
10
11   animation.duration = duration
12   animation.frame = 1
13   animation.time = 0
14
15   animation.pos = pos
16
17   return animation
18 end
19
20  function framecounter (time, duration)
21   if time >= duration then
22     time = 0
23   end
24   return math.floor ((time / cmcf.walk.duration) * #cmcf.walk.frames) + 1, time
25 end
```

tion you can make actions last exactly as long as intended.

Say you want a full walk cycle of Cubey McCubeFace to last exactly half a second. You could modify your program to look like Listing 6.

You set a new variable `time` to `0` (line 2), and each time you enter `update ()` you add `dt` to `time` (notice you need to make `dt` a parameter of `love.update ()` for this to work). Then, if you divide `time` by the duration you want for your animation (`0.5` seconds, in this case) as shown on line 21, you will have the stage in the animation counting from when `time` was `0`. For example, if `time` is `0.25`, a quarter of a second has passed since the animation started:

```
0.25 / 0.5 = 0.5
```

This tells your program you are halfway through the animation.

Now multiply that by the number of frames (`#frames` – you use `#` to get the number of items in any table), and you will get a number between zero and five. Get rid of the decimals using Lua's `math.floor ()` function, add one and you've got a number between one and six, the number of the frame you need to show.

As soon as the time reaches the duration of the animation, you reset `time` to `0` (line 17 through 19) and start all over again.

Run the program, and you will see that Cubey McCubeFace runs through one walk cycle every half a second.

Clean LÖVE

To keep things nice and tidy, you can take all the initiation and calculations out to separate functions. To make your `main.lua` file even cleaner, you can then separate those functions into a different file.

In this example, Listing 7 is your main file, and Listing 8 is where all the gory stuff goes.

To include the contents of Listing 8 in your `main.lua` file, all you need is the `require` command (line 1).

Most of what is going on in Listing 8 should be self-explanatory, but it is worth pointing out how Lua tables can act like pseudo-classes that will allow you to create a `cmcf` (Cubey McCubeFace) walk cycle type-object, a jump cycle type-object, a run cycle type-object, and so on.

User Libraries

That is how you can create an animation from scratch using LÖVE's built-in tools. But you could also use *anim8* [4], a LÖVE library that takes the drudgery out of animating sprites and lets you have the different frames on a grid (as opposed to just on a line) and pick what frames to use in an animation.

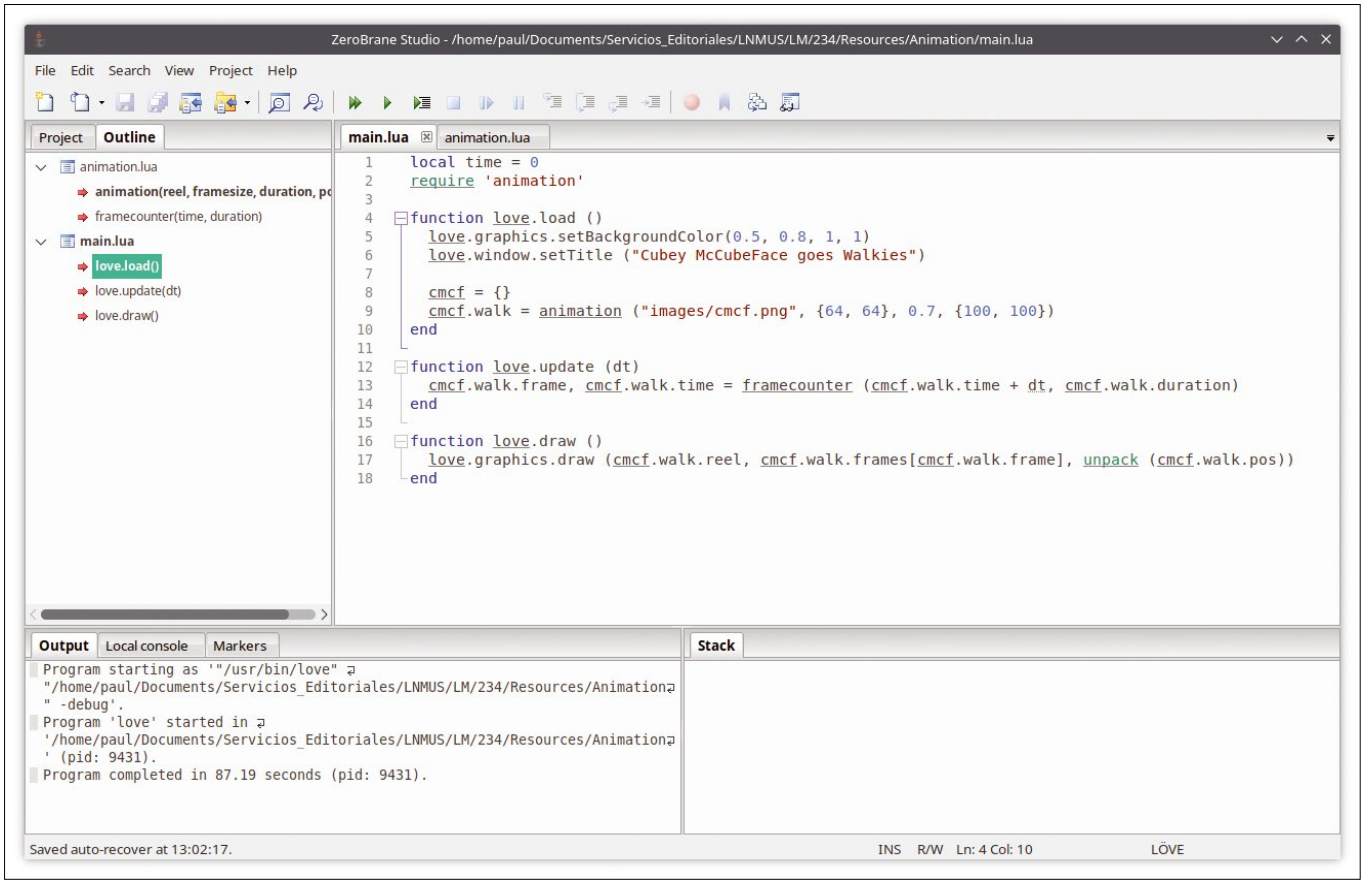


Figure 4: ZeroBrane Studio is probably the best IDE for programming in Lua.

Like *anim8*, there are many other libraries for LÖVE, that apart from making animation easy, cover physics, isometric 3D tiles, collision detection, and much more. Before putting yourself through the hassle of programming a whole super-framework from scratch, check that someone hasn't already done it for you [5] – unless you are doing so to learn the basics. (Also, for a brief word about the editor you use for programming, see the box "Ideal IDE.")

LÖVE Platforms

Your LÖVE games can be ported to other platforms, including Windows, iOS, and Android (Figure 5), and the LÖVE wiki explains in detail how to do that [7]. However, you can test-run your program on Android before you go to all the trouble of compiling or pushing it through a toolchain to get a native APK.

On your computer, you have to create a file called `conf.lua`, like the one in Listing 9.

`"XX.X"` is the version of the LÖVE interpreter on your computer. You can find this out by running the LÖVE interpreter from the command line with `love`. The version of LÖVE is shown in the titlebar.

Put `conf.lua` in the same directory as your `main.lua` file, enter the directory and zip everything up with

```
zip -9 -r YourGame.love .
```

Listing 9: conf.lua

```
01 function love.conf(t)
02     t.version = "XX.X"
03 end
```

Ideal IDE

Although your regular, favorite text editor will do just fine, there is some merit in going with ZeroBrane Studio [6] (Figure 4). Not only is it an efficient little editor with all the bells and whistles you need for Lua programming (syntax highlighting, text completion, file management, etc.), it also comes with hooks to a bunch of the most popular Lua interpreters, including LÖVE and Moai, another framework for gaming development.

Go to *Project* | *Lua Interpreter* in the menu, pick LÖVE from the list, and you'll be able to run and debug your game directly from the IDE.

You can choose a different name from `YourGame`, of course, but your zipped file must have the extension `.love`.

On your phone, download the LÖVE interpreter for Android from Google Play [8] and then copy the `.love` file from your computer over to your Android device. You'll be able to run it with your newly installed LÖVE interpreter.

Conclusions

We have only touched on one of the many superficial tasks you will have to carry out when creating your own game. However, my hope is to tempt you to get your toes wet and to help you get your head round the most basic principles of programming with LÖVE. As a fan of old-school, casual gaming, I can't wait to see what you do with it. ■■■

Info

[1] LÖVE game framework: *https://love2d.org*

[2] Lua programming language: *https://www.lua.org/*

[3] LÖVE wiki: *https://love2d.org/wiki/Tutorial:Callback_Functions*

[4] The anim8 library for LÖVE: *https://love2d.org/forums/viewtopic.php?f=5&t=8281*

[5] List of LÖVE libraries: *https://www.love2d.org/wiki/Category:Libraries*

[6] ZeroBrane Studio: *https://studio.zerobrane.com/*

[7] How to port your games to other platforms: *https://love2d.org/wiki/Game_Distribution*

[8] LÖVE interpreter for Android: *https://play.google.com/store/apps/details?id=org.love2d.android*

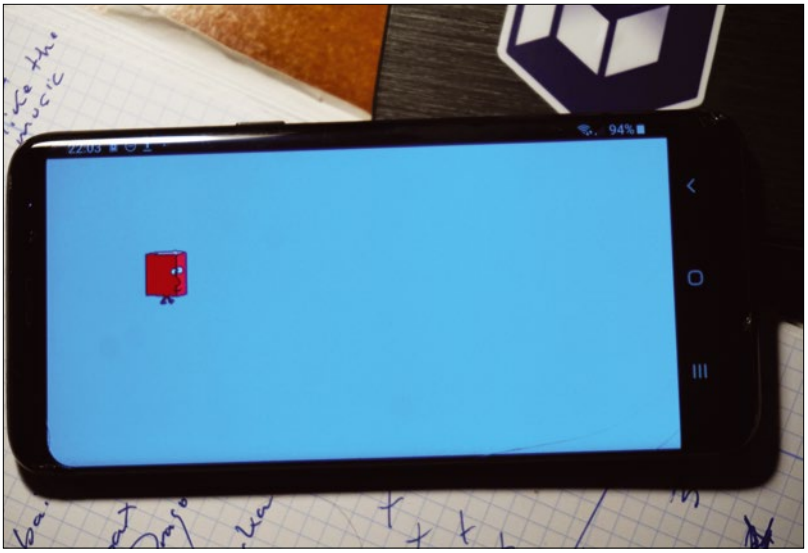


Figure 5: Cubey McCubeFace goes walking on an Android phone.

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

LINUX NEWSSTAND

Order online:
https://bit.ly/Linux-Newsstand

Linux Magazine is your guide to the world of Linux. Monthly issues are packed with advanced technical articles and tutorials you won't find anywhere else. Explore our full catalog of back issues for specific topics or to complete your collection.

#232/March 2020

Stop Ads

Browser-based ad-blockers are useful for controlling many types of pop-ups and banners, but they are less effective with ads built into applications. We look at a couple of alternative tools for blocking ads at the network level: Pi-hole and Privoxy.

On the DVD: GParted 1.0.0 and Kali Linux 2019.4

#231/February 2020

Tiling

Are tiling window managers still relevant for today's desktop? We explore the possibilities of the tiling paradigm with a modern Linux built for tiling.

On the DVD: MX Linux MX-19 and Zorin OS 15 Core

#230/January 2020

Automation Tricks

Home automation is no longer the stuff of science fiction. We explore some versatile tools for Linux users who are interested in IoT but don't want to surrender the freedom of open platforms and open source.

On the DVD: Fedora Workstation 31 and Ubuntu "Eoan Ermine" Desktop 19.10

#229/December 2019

The Future of Vector Graphics

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.

On the DVD: Arch Linux 2019.10.01 and CentOS 8.0.1905

#227/October 2019

Backup

The bad news is that disasters happen. The good news is that Linux has a variety of powerful backup tools that will protect you when disaster strikes. We cover some backup apps for the Linux environment, including Rclone, restic, and rsnapshot.

On the DVD: Linux Mint 19.2 "MATE Edition" and Debian 10 "Buster"

#226/September 2019

Spotlight on Small Distros

Hardware resources keep expanding, and mainstream OS systems also keep expanding, sucking up the newfound space with more and better bloatware. Many users would rather keep it simple. We review some of the top resource-conscious small distros.

On the DVD: Manjaro Gnome 18.0.4 and MX Linux 18.3

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*.

NOTICE

Be sure to check the event website before booking any travel, as many events are being canceled or converted to virtual events due to the effects of COVID-19.

Red Hat Summit 2020

Date: April 28–29, 2020

Location: Virtual Experience

Website: *https://www.redhat.com/en/summit*

Immerse yourself in our free virtual event and find your inspiration at the intersection of choice and potential. Our virtual event will feature the keynotes, breakout sessions, and collaboration opportunities. This programming will be shared as a blend of live and recorded content designed to inspire and engage a global audience.

ISC High Performance 2020

Date: June 21–25, 2020

Location: Frankfurt, Germany

Website: *https://www.isc-events.com/ct/click.php?id=129*

More than 3,700 practitioners and users will come together in Frankfurt June 21-25 to discuss and explore international developments and trends in some of the fastest growing segments of IT. Registration is open now!

Events

DevOpsCon	April 21-24	London, United Kingdom	https://devops.jaxlondon.com/
Red Hat Summit 2020 Virtual Experience	April 28-29	Worldwide	https://www.redhat.com/en/summit
Linux Presentation Day 2020	May 16	Cities across Europe	https://l-p-d.org//en/start
DrupalCon Minneapolis 2020	May 18-22	Minneapolis, Minnesota	https://events.drupal.org/minneapolis2020
OSCON (Open Source Software Conference)	June 13-16	Portland, Oregon	https://conferences.oreilly.com/oscon/oscon-or
Software Architecture Conference	June 15-18	Santa Clara, California	https://conferences.oreilly.com/software-architecture/sa-ca
stackconf 2020	June 17-18	Berlin, Germany	https://stackconf.eu/
ISC High Performance 2020	June 21-25	Frankfurt, Germany	http://ct.isc-events.com/click.php?id=138
Linux Security Summit North America	June 24-26	Austin, Texas	https://events.linuxfoundation.org/linux-security-summit-north-america/
Strata Data & AI Conference	September 14-17	New York, New York	https://conferences.oreilly.com/strata-data-ai/stai-ny
Open Source Summit Japan	September 15-16	Tokyo, Japan	https://events.linuxfoundation.org/open-source-summit-japan/
Storage Developer Conference	September 21-24	Santa Clara, California	https://www.snia.org/events/storage-developer
Open Networking & Edge Summit Europe	September 29-30	Antwerp, Belgium	https://events.linuxfoundation.org/open-networking-edge-summit-europe/

Images © Alex White, 123RF.com

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to *edit@linux-magazine.com*.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at:
http://www.linux-magazine.com/contact/write_for_us.

NOW PRINTED ON recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Copy Editors
Amy Pettle, Megan Phelps

News Editor
Jack Wallen

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
© Ikorch, 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

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the disc provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2020 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media USA, LLC, unless otherwise stated in writing.

Linux is a trademark of Linus Torvalds.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by 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.

Approximate

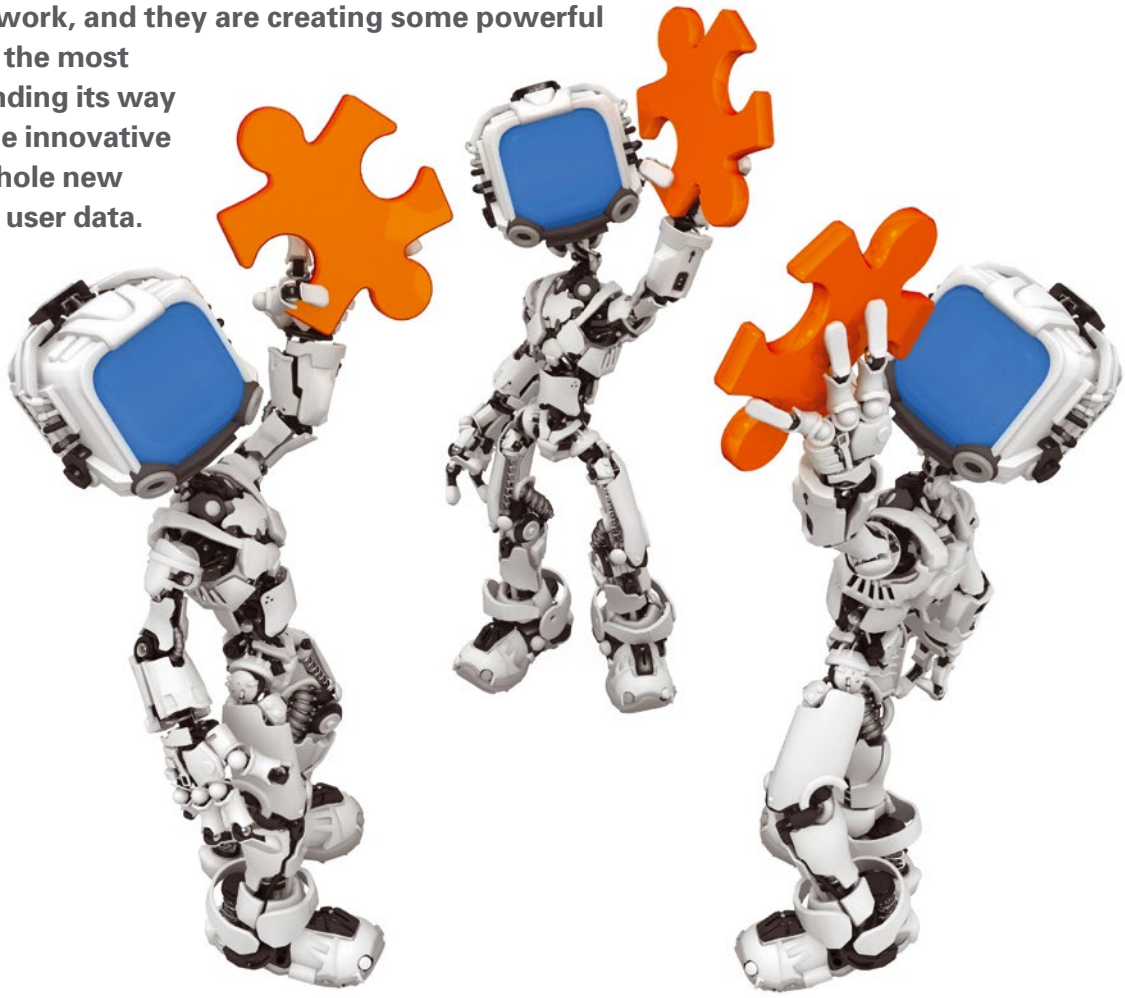
UK / Europe	May 09
USA / Canada	Jun 05
Australia	Jul 06

On Sale Date

Issue 235 / June 2020

Coming to Systemd

With the dust of the flame wars finally settling, it seems that the controversial systemd service manager truly is here to stay. If you're still getting used to systemd, you might not realize that the developers are still at work, and they are creating some powerful enhancements. One of the most important changes winding its way down the pipeline is the innovative Systemd-homed – a whole new approach to managing user data.



Preview Newsletter

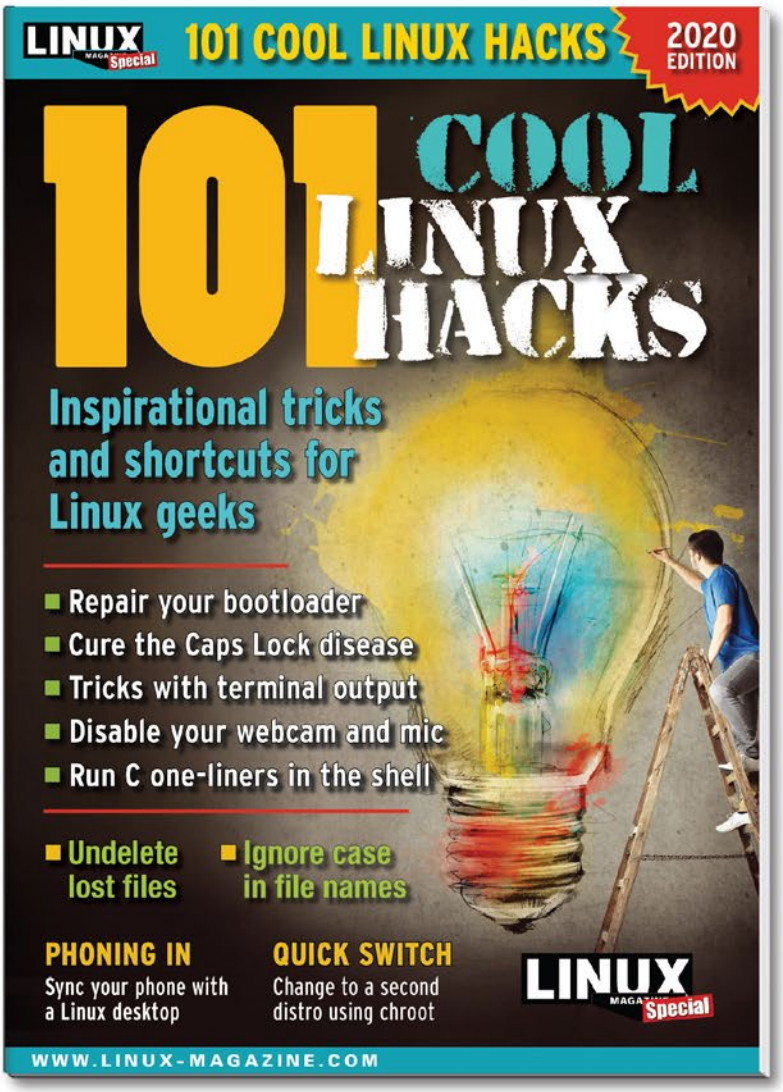
The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

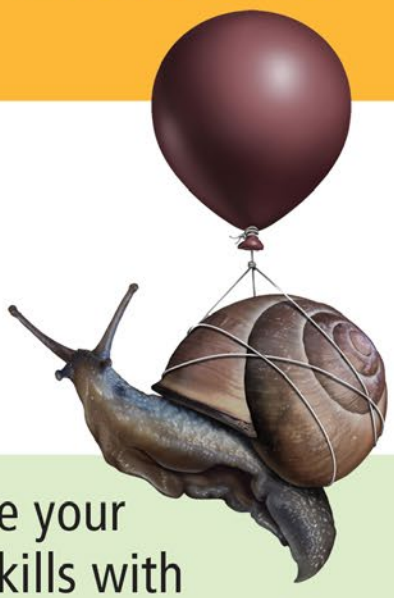
Sign up at: ***https://bit.ly/Linux-Update***

Lead Image © higyou, 123RF.com

SHOP THE SHOP
shop.linuxnewmedia.com

GET PRODUCTIVE WITH 101 LINUX HACKS





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

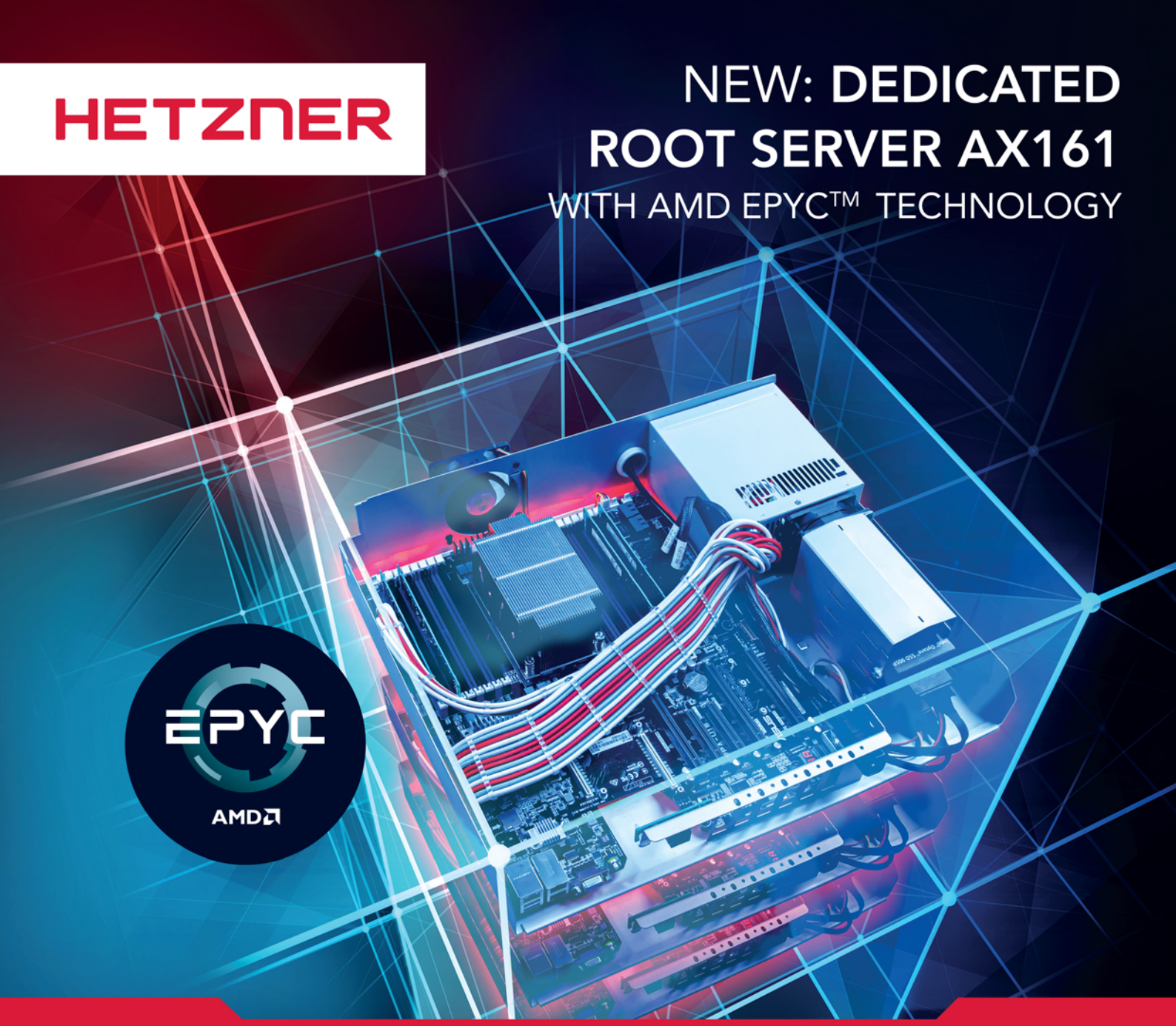
- Undelete lost files
- Cure the caps lock disease
- Run C one-liners in the shell
- Disable your webcam and mic
- And more!

ORDER ONLINE:
shop.linuxnewmedia.com/specials

HETZNER

NEW: DEDICATED ROOT SERVER AX161

WITH AMD EPYC™ TECHNOLOGY



EPYC'S ALL GROWN UP!

UNMATCHED PERFORMANCE WITH STRONG SECURITY SYSTEM

Configure it with upto 8 x NVMe/SATA SSD or 3 x HDD and 512 GB RAM

Dedicated Root Server AX161

- ✔ AMD EPYC 7502P 32 Core "Rome" (Zen2) Simultaneous Multithreading
- ✔ 128 GB DDR4 ECC RAM (configurable, at additional cost)
- ✔ individual storage capacity (at additional cost)
- ✔ 100 GB Backup Space
- ✔ Traffic unlimited
- ✔ Location Germany
- ✔ No minimum contract
- ✔ Setup Fee $135

monthly from **$135**

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. All rights reserved by the respective manufacturers.

www.hetzner.com