ARCHIVE DVD

FREE DVD $39.90 VALUE!

RASPBERRY PI GEEK
THE COMPLETE ARCHIVE
2,000 pages of maker projects and more!

STREAM TO YOUR TV

PRO LINUX MAGAZINE

LINUX PRO MAGAZINE

APRIL 2020

STREAM TO YOUR TV

Cool tools for sound and video

Analyze File Metadata

Apache Cassandra
Database for the Internet age

Web Scraping
Find and format the information you need

Smart Display
Relive your memories with a Rasp Pi powered picture frame

Kindd
Create bootable images in a GUI

Kubuntu Focus
A laptop built for KDE

LINUXVOICE

- Chat with Jami
- maddog on Yggdrasil
- Krita vs. MyPaint

FOSSPicks
- enve Animation
- ShellCheck Script Analyzer
- Retro Puzzles with VVVVVV

Tutorials
- Customize GRUB and KDE Boot
- Create a Notification App

Issue 233
Apr 2020
US$ 15.99

LINUX NEW MEDIA
The Pulse of Open Source

HERE'S TO KNOWING THAT AGAIN

Dear Reader,

This column is about IT, not about politics. The names of politicians sometimes come up in this space – mostly because of something they did that is related to IT, but it is never my goal to descend into the political fray. In fact, I honestly believe the whole reason the political fray exists is because it is much easier to reduce everything to politics than it is to deal directly with the perplexing and often unsolvable issues that politicians face: social responsibility, economics, national security, personal liberty. A collection of opinions on these perplexing topics is often encapsulated into a convenient bundle plan that is associated with a particular politician, and when you say you like that politician, you imply some level of comfort with that politician's opinion bundle. Seems like it rarely ever gets much deeper than that.

On February 13, a federal judge issued an injunction to stop work on the massive Joint Enterprise Defense Infrastructure (JEDI) cloud project for the US Department of Defense (DoD). Amazon had brought the suit in an attempt to prevent the $10 billion project from going to the contract winner, Microsoft. In the USA, a flurry of lawsuit threats often follows the awarding of a gigantic government contract. A common legal theory behind many of these post-award legal maneuvers is that the contract was awarded in "bad faith," meaning that the government agency was acting with animus toward one of the losing bidders. These lawsuits hardly ever work, because bad faith is very difficult to prove, and the legal standard is that it must be proved convincingly – mere inference or innuendo are not enough, and even incompetence or negligence isn't a compelling argument. You really have to show that the government acted with malice. So usually it doesn't work. However, legal experts say the judge wouldn't have issued the injunction unless the case had at least some likelihood of succeeding in court. Amazon Web Services had to put a $42 million deposit, which they will forfeit if it is proven that the injunction to stop the project was wrongly issued.

Given the difficulty in proving bad faith, why do the judge and plaintiff seem so confident that there might be a case this time? Because former Secretary of Defense James Mattis has written that the president told him directly to "screw Amazon" on the JEDI contract [1]. Mattis, who is well regarded by most members of Congress and was affirmed in the Senate on a vote of 98-1 (the one no vote being a Democrat), is thought to be a highly credible source.

President Trump has a long-standing feud with Amazon president Jeff Bezos that apparently stems from Bezos's ownership of the *Washington Post*. Trump stated in 2015 that he hoped Amazon "…would crumble like a paper bag" [2], and, according to legal documents filed with the injunction, "…it was reported in April 2018 that President Trump discussed with his advisors ways to 'escalate his Twitter attacks on Amazon to further damage the company'" [3]. Amazon is seeking the opportunity to depose witnesses and gather information to determine what the president meant when he told the Secretary of Defense to "screw Amazon."

I can't remember any previous president (of either party) spending this much time worrying about how to "screw" or "damage" a legal US corporation. Conservatives, in particular, who have spent years rallying around the philosophy that "government should not be picking the winners and losers" in the business world, should be very concerned about the propriety of such remarks and should be leading the charge for more information.

Of course, once the evidence comes out, it might not bear out Amazon's side of the story, in which case, Amazon would not be the first company to overstate its case. But the extreme nature of the president's tweets, and what appears to be a direct order to "screw" a company that is competing for a government contract, raise serious questions about the integrity of the process.

Note that I'm not writing this to be pro-Amazon or anti-Trump or anti-DoD. What I'm really talking about is the IT procurement process. The leader of the government should not talk about "screwing" one of the bidders on a government contract for the same reason that a basketball referee should not talk about "screwing" one of the teams in a basketball game. We used to know that. Here's to knowing that again sometime soon….

Joe Casad,
Editor in Chief

Info

[1] "Trump Ordered Mattis to 'Screw Amazon' on Pentagon Contract According to New Book": *https://www.cnn.com/2019/10/26/politics/amazon-donald-trump-jim-mattis-pentagon-contract/index.html*

[2] "Trump Chides Amazon Head Bezos and His Washington Post": *https://www.newsweek.com/donald-trump-bezos-washington-post-amazon-401876*

[3] Federal Claims Protest (Redacted Version): *https://cdn.pacermonitor.com/pdfserver/37BX4IY/119628238/AMAZON_WEB_SERVICES_INC_v_USA__cofce-19-01796__0111.0.pdf*

LINUX MAGAZINE

WHAT'S INSIDE

The line between computers and television blurred long ago, but the new tools and new ideas keep coming. This month we highlight some innovative apps for multimedia in Linux, including Gnome Cast for TV, and the easy-to-use Serviio media server. Also in this issue:

- **Apache Cassandra** – Find out why this free NoSQL database fills an important niche for companies with lots of data (page 28).
- **Metadata in the Shell** – Use Bash scripts and built-in commands to track down useful metadata (page 40).

Check out MakerSpace for a look at how to create a smart picture frame, and turn to LinuxVoice for a tour of the privacy-focused Jami messenger app.

SERVICE

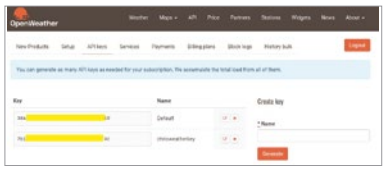
NEWS

REVIEWS

COVER STORIES

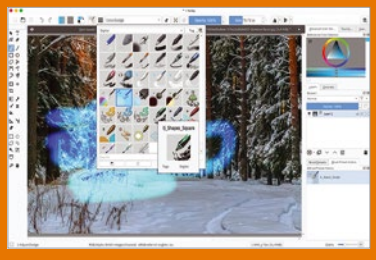
IN-DEPTH

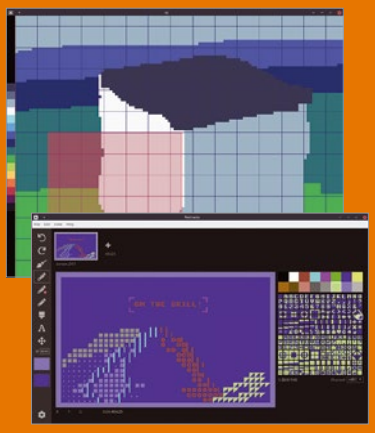
MAKERSPACE



SEE PAGE 6 FOR DETAILS

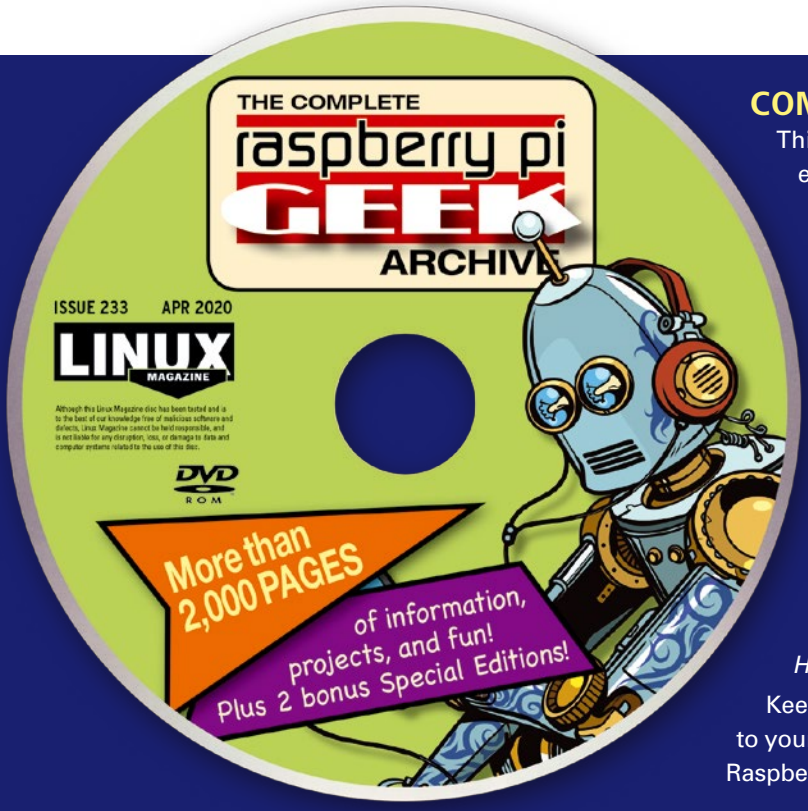
LINUXVOICE





On the DVD

THE COMPLETE raspberry pi GEEK ARCHIVE



COMPLETE RASPBERRY PI GEEK ARCHIVE

This month we bring you something special: an encyclopedic DVD with every issue of *Raspberry Pi Geek* magazine. You get more than 2,000 pages of practical Raspberry Pi Geek knowledge.

This comprehensive collection includes awesome projects, in-depth programming tutorials, and product reviews – all in a convenient searchable format.

On the disc, you'll find more than 350 articles on Raspberry Pi and Arduino, from games you can play, to servers you can run, through detailed instructions on how to build your own hardware projects with components you scavenge from around the house.

PLUS you get 2 bonus special editions: *Raspberry Pi Handbook* and *Raspberry Pi Adventures*!

Keep the Complete Raspberry Pi Geek Archive DVD close to your workbench as a permanent reference of the world of Raspberry Pi.

Your DVD includes...

- More than 2,000 pages of RPi goodness
- Searchable database
- Complete code repository
- Articles in PDF and HTML format
- No DRM – read anywhere

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.

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

LibreOffice 6.4 Released

LibreOffice 6.4 (*https://wiki.documentfoundation.org/ReleaseNotes/6.4*) has arrived for Linux (and other platforms) and is one all users will want on their desktops. Why? Two words: Productivity and compatibility. The developers have gone out of their way to make a number of tasks (such as saving spreadsheets and presentations) faster.

Version 6.4 adds new features and tweaks, making the office suite considerably more efficient. For example, the LibreOffice Start Center now includes thumbnails of previously edited documents, making it much faster to locate the file you want to work on.

One previous feature that has been drastically improved is the redaction tool (which was only recently introduced to LibreOffice). Prior to version 6.4, using the redaction tool was a bit of a challenge. Now, LibreOffice has a new automatic redaction mode that allows users to quickly mask any and all data that matches a specific word, phrase, or expression. The auto-redaction tool allows you to add and save words, phrases, and expressions as targets and then apply them to any document. This way you can save specific phrases you have to frequently redact so that they can be easily applied to documents with which you work.

Finally, the developers have spent considerable time and effort perfecting the compatibility with MS Office documents. In fact, the Document Foundation (*https://www.documentfoundation.org/*) now claims LibreOffice enjoys "almost perfect support for `.docx`, `.xlsx`, and `.pptx` files."

Other new features and improvements found in LibreOffice 6.4 include:

- A built-in QR code generator, so you can quickly add QR codes to documents.
- A Table panel included in the Writer sidebar, for faster table creation.
- Comments can now be marked as resolved.
- A much improved help system.
- The ability to enable/disable the sending of crash reports.

Official Evernote Client Coming to Linux

Evernote has been one of the more popular note taking apps for quite some time. Since the beginning, it was labeled as a cross-platform application. However, the one platform missing from the list was Linux. Evernote has always

been available for Android, iOS, Windows, and macOS. But that will soon be changing, as Evernote recently announced an official Linux client is on the way.

In a blog post (*https://evernote.com/blog/2020-update-progress-road-ahead/*), Ian Small, CEO of Evernote, said, "The re-engineered web client (in limited release), the new mobile clients (in first preview), and the (as yet unreleased) new clients for Windows, Mac, and (yes!) Linux, along with the ongoing re-architecture and data migration we've been doing in the cloud, will set up Evernote to be able to innovate and ship with quality at a pace we haven't seen in a long time."

The Evernote note-taking client offers features like:

- Handwriting search: Find your text in any note.
- Templates: Makes for faster and better note taking.
- Notes sync: Keep your notes available on all devices associated with your account.
- Offline notes (premium account required): Makes all of your notes available anywhere, anytime (even without an Internet connection).
- Uploads (premium account required): Up to 10GB monthly note uploads.
- Large notes (premium account required): A 200MB maximum note size.

Although there are other third-party Evernote clients for Linux (such as NixNote (*http://nixnote.org/*), ForeverNote (*https://github.com/milan102/ForeverNote*), and Tusk (*https://github.com/klaussinani/tusk*)), this will be the first official client for the platform. As of now, there has been no word on if the Linux desktop client will be released as an Electron or a native application. Nor is there a timeline for the release.

Thanks to Linux, Google and Valve are Bringing Steam to Chromebooks

On many supported Chromebooks, it is already possible to run Linux applications on the Chromebook. For certain user types, this has been a real boon. However, for gamers, not so much. That is about to change, thanks to a joint effort by Google and Valve.

Image © Konstantin Savusia, 123RF.com

According to Kan Liu, director of product management for Google Chrome OS, Steam is coming to Chromebooks. Steam is a digital video game distribution service, offered by Valve, originally released in 2003 as a means for Valve to provide automatic updates for their own line of games. Eventually the service was expanded to include third-party publishers, and it is now one of the largest digital distribution systems for games.

This new evolution for the Chromebook wouldn't be possible without the addition of Linux compatibility for Chromebooks. So not only will Chromebook users be able to install from the massive catalogue of Linux applications, they will (in the near future) be able to run the same Steam games available to the Linux platform.

There is, of course, one caveat: Many of the Chromebooks on the market today run low-end specs. Those devices will most likely only be able to enjoy the very basic 2D games. In order to run more modern, graphics-intensive games, the Chromebook will require significantly beefier hardware.

At the moment, it is possible to install the Steam Linux client on Chrome OS using the Crostini Linux compatibility layer. However, that installation offers zero support and very poor performance. The official rollout will take some time … maybe even years. Until then, you can satisfy your Linux fix on Chromebooks with the software available via Crostini and the `apt-get install` command.

Original announcement: *https://www.androidpolice.com/2020/01/17/exclusive-google-is-working-to-bring-steam-to-chrome-os/*

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

Building a Virtual NVMe Drive
• Petros Koutoupis
An economical and high-performing hybrid NVMe SSD is exported to host servers that use it as a locally attached NVMe device.

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

Manipulation Detection with AFICK
• Tim Schürmann
AFICK is a small, free tool that helps administrators detect attempts to manipulate documents and system files.

Cron Alternatives fcron and hcron
• Anzela Minosi
The fcron and hcron scheduler alternatives to cron are suitable for computers that do not run around the clock, and each comes with specific benefits.

Container IDE with Cloud Connection
• Bernhard Bablok
Gitpod relies on technologies such as Docker and Eclipse Theia to serve up individual development environments for GitHub projects.

Nine-Year-Old Bug Found and Fixed in Sudo

Sudo is found in most Linux distributions and is responsible for elevating privileges for users, so that they can perform admin tasks. Recently it was discovered that a buffer-overflow bug (*https://www.sudo.ws/alerts/pwfeedback.html*) had been in hiding for nine years. This bug (CVE-2019-18634; *https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-18634*), which has existed in sudo versions 1.7.1 through 1.8.25p1) can be triggered when an administrator or a downstream distribution (such as any based on Debian/Ubuntu) enables the `pwfeedback` option in the `/etc/sudoers` file. Once `pwfeedback` is enabled, the vulnerability can be exploited by any user on the system (even those not listed in the `sudoers` file).

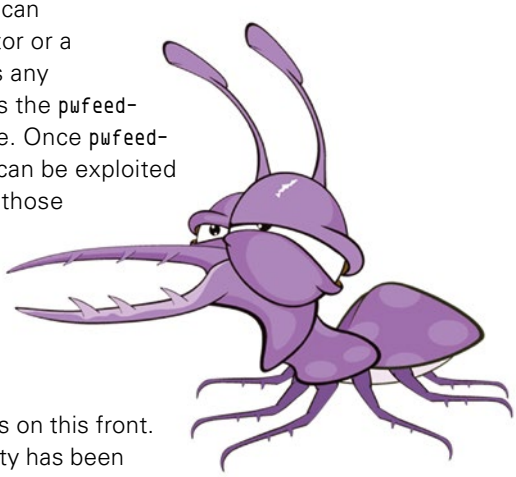


Image © Liudmila Pantelejenkova, 123RF.com

The `pwfeedback` option is used to hash passwords when you type them (so the irony of this feature being a security vulnerability cannot be missed).

There are two bits of good news on this front. First and foremost, the vulnerability has been patched. So long as you've updated sudo to any version beyond 1.8.25p1, you're safe. The second bit of news is that, even if you've not updated, `pwfeedback` isn't enabled by default in most distributions. Issue the command `sudo -l` to see if `pwfeedback` is listed among the enabled options. If not, you're good to go. If you do see `pwfeedback` in the output of the command, upgrade sudo immediately and consider disabling the option.

Systemd-homed Is Coming to a Linux Distribution Near You

For decades, the Linux home directories and user accounts have been managed in the same fashion. The `/etc/passwd` file included usernames, user IDs, and home directory locations, while the `/etc/shadow` file contained user password hashes. Those two files worked in conjunction to make user logins and home directories possible.

That might be changing soon, thanks to systemd-homed.

Lennart Poettering is the main developer behind the widely-adopted systemd Linux initialization system. Although much maligned initially, systemd eventually became the de facto standard for the majority of Linux distributions.

Poettering has been at work on something special for user home management.

Ladies and gentlemen, introducing systemd-homed.

Instead of using the traditional means of user/home management, systemd-homed will collect all configuration data for each component and store the information (username, group membership, password hashes, and any other relevant information) in a JSON file. On top of that, the home directories will be linked as a LUKS encrypted container, with encryption coupled with the user login. The biggest plus of this system is that as soon as a user logs in, the home directory is decrypted. Once a user logs out, the home directory is automatically encrypted.

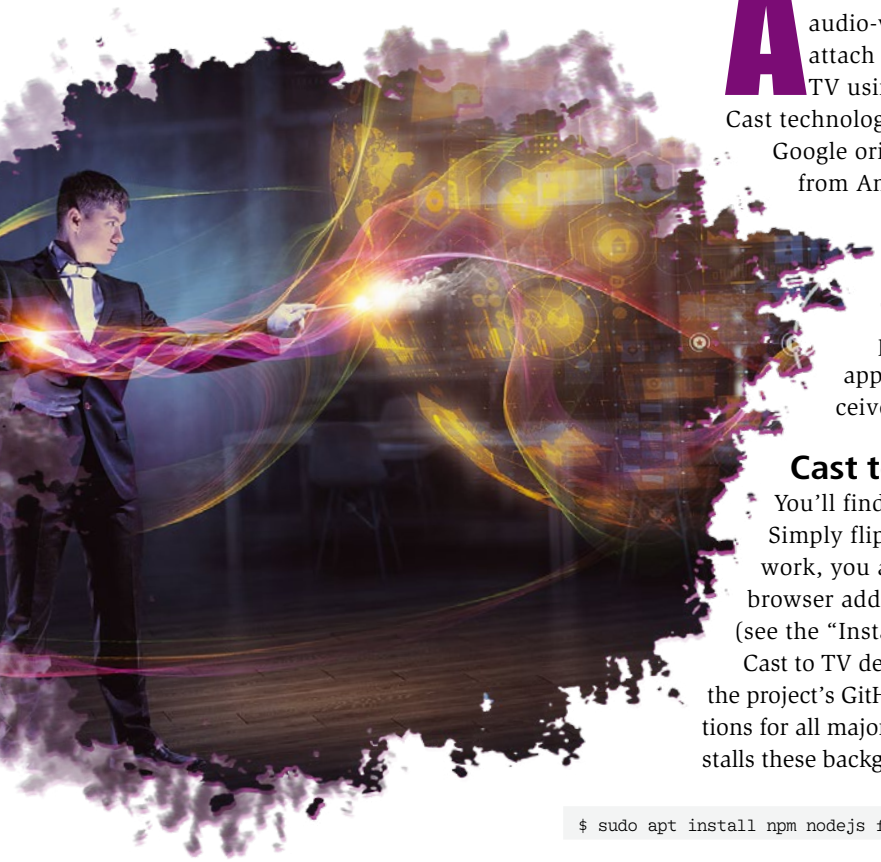
Although this has been in development for some time, it looks like systemd-homed will officially become a reality with the release of systemd 245. That release doesn't mean all systemd-based distributions will automatically adopt systemd-homed. However the idea of on-demand home directory encryption should appeal to most distributions and users.

Original announcement: *https://linuxreviews.org/Systemd-Homed_Is_Merged_And_It_Will_Fundamentally_Change_Linux_Home_Directories*

Cast to TV Gnome Shell Extension

Casting Pearls

Cast to TV is a Gnome extension that streams media files from a PC to a Chromecast-enabled device. *By Christoph Langner*



A Chromecast [1] is a small USB dongle that lets you stream audio-visual content from a mobile device to your TV. Just attach Chromecast to your TV, and you can stream to the TV using an Android or other device that supports Google Cast technology.

Google originally developed Chromecast to support streaming from Android phones, but the open source community has never been slow to grasp an opportunity. Cast to TV is a Gnome Shell extension that lets you send pictures, music, and video to a Chromecast-enabled TV from the Gnome desktop. And if you don't happen to have a Chromecast device, use the Playercast app on a Raspberry Pi or other Linux computer to receive the stream and act as an interface to your TV.

Cast to TV for Gnome

You'll find Cast to TV at the Gnome Extensions [2] website. Simply flip the switch from *Off* to *On*. For the installation to work, you also must install the native host connector and a browser add-on available for Chrome/Chromium and Firefox (see the "Installing Gnome Extensions" box).

Cast to TV depends on the packages *npm*, *nodejs*, and *ffmpeg*. On the project's GitHub site, the developers provide installation instructions for all major Linux distributions [6]. The following command installs these background packages on a recent Ubuntu:

```
$ sudo apt install npm nodejs ffmpeg
```

Keep in mind that Node.js is a heavyweight with numerous sub-packages; the installation will take a while, especially on older machines.

When these packages are installed, a new entry appears in the Gnome menu in the upper right corner called *Cast Media*. Unfold it and open *Cast Settings*, from which you install the required Node Package Manager (npm) modules as a final step: Open *Modules* and click *Install npm modules* (Figure 1). In just a few seconds, the system installs all the required files on the computer.

All requirements are now met and the configuration is complete. Now open the Gnome menu again and click *Cast Media* and then *Turn On*, which closes the menu. When you open the Gnome menu a third time, you will find a *Cast Media* entry with the sub-items *Video*, *Music*, and *Picture*, as well as *Turn Off* and the ubiquitous *Cast Settings* (Figure 2).

Installing Gnome Extensions

To install from the Gnome Extensions web portal, the system must meet two requirements: Both the native host connector and the appropriate web browser add-on must be installed. On Debian, Ubuntu, and Linux Mint, first set up the host connector with the *chrome-gnome-shell* package, and then install the appropriate add-on for Chrome/Chromium [3] or Firefox [4]. The Gnome wiki describes the procedure for all other common distributions [5].

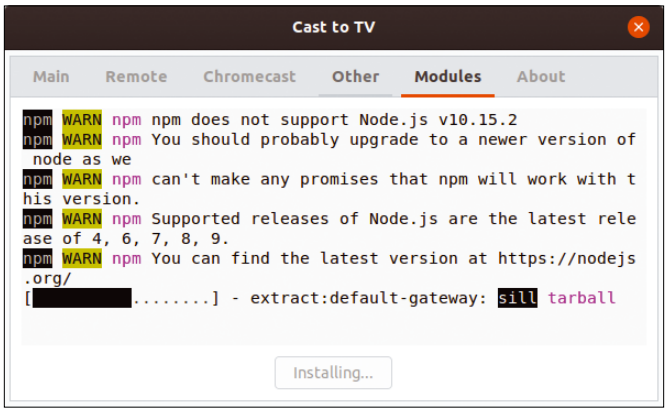


Figure 1: For Cast to TV to work, you need to install the npm modules required by the program. A single click does the trick.

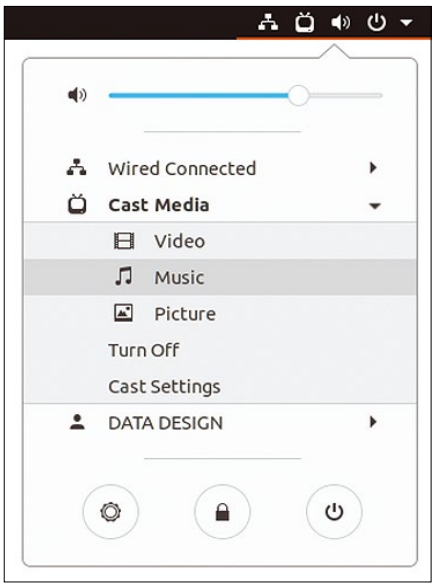


Figure 2: The Gnome menu in the panel lets you use Cast to TV to stream pictures, music, or videos to Chromecast-compatible devices, which include not only the Chromecast dongle itself, but also TV sets equipped with Android TV.

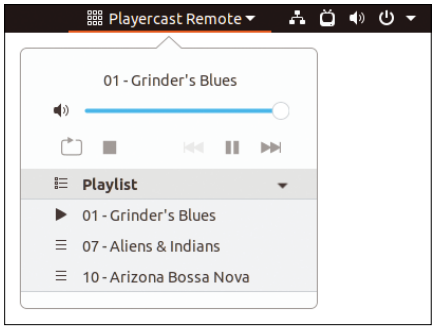


Figure 3: You can control the playback of the streamed media from another menu. Use the slider to jump to another position in the track. Drag and drop to control the order of items in the playlist.

Two or More Chromecast Devices

If you use multiple Chromecast dongles or Chromecast-enabled devices in your household, you need to tell Cast to TV the device to which you want to send the stream. For this purpose, go to the *Cast Settings* | *Chromecast* tab, and beside the *Device selection* item, Cast to TV lists all the Chromecast devices it finds. If necessary, devices that are not found can be configured manually by their IP addresses from the gear icon.

From Your Computer to the TV Set

Casting is now as easy as clicking on one of the media formats and selecting the files to play. Cast to TV lets you select several files in one step, as long as they are in the same directory. Then, you need to check the TV set, which should be playing the selected media files. Additionally, a new menu appears in the Gnome panel at the top of the screen labeled *Chromecast*. (See also the box "Two or More Chromecast Devices.")

Clicking on the entry brings up a dialog in which you can find a progress bar, playback control buttons, and – after clicking *Playback* – a playlist (Figure 3). If you tap the Stop icon, Cast to TV immediately stops streaming and the menu disappears. From the Gnome menu, you can open *Cast Settings* during playback to pop up a lightweight configuration dialog where you can adjust the Chromecast menu.

If you enable *Nautilus* | *Nemo Integration* under the *Cast Settings* | *Other* tab, it is even easier to select a video or track to stream. Restart the file manager completely (e.g., `nautilus -q`) to stop all background services, as well; a new *Playercast* entry appears in the context menu for multimedia files. With the *Play Files* and *Add to Playlist* options (Figure 4), you can then send the selected files to the Chromecast receiver.

On the other hand, Cast to TV cannot yet transfer the complete desktop to the TV. At press time, this function only worked with Chrome or Chromium by opening the web browser and selecting *Cast* from the menu. You will then find a new icon to the right of the address line. From this menu, you can choose between the Chromecast devices active on the network and the options to stream either a tab, the whole desktop, or a file (Figure 5). Note that streaming the desktop only

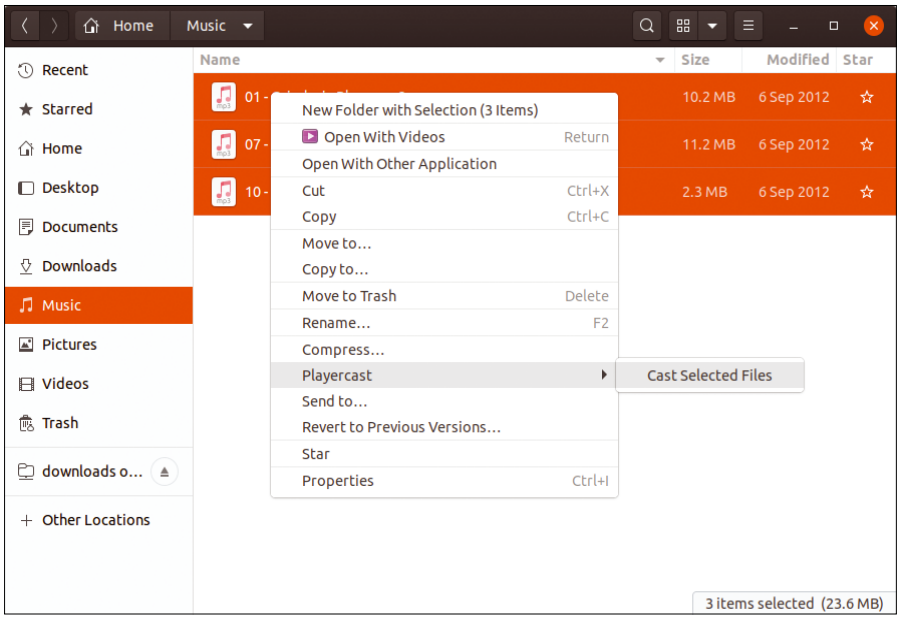


Figure 4: Cast to TV integrates into the file manager. From the context menu for multimedia files, you can cast pictures, music, or videos directly from Nautilus to the Chromecast receiver.

works with the classic X server, not with Wayland. To switch desktops, you need to log out of the desktop and, after selecting the user account from the gear menu, select *GNOME on Xorg*.

Installing Playercast

Even if you do not have a Chromecast dongle or a device with an integrated Chromecast receiver, you do not have to give up on the idea of streaming media content with Cast to TV. Equipped with Playercast [7], a Raspberry Pi connected to a TV over HDMI is the ideal receiver. (See also the "HDMI-CEC" box.) Alternatively, any other Linux PC will work, as well.

Playercast can be downloaded by npm to the computer you will be using to receive the streams, but first, you have to install npm and the MPV media player [8] as dependencies with the conventional package manager:

```
$ sudo apt install npm mpv
$ sudo npm install -g playercast
$ playercast <IP address>:<Port> ### Default port is 4000
```

Executing the `playercast` command on the third line tests the installation. The parameters are the IP address of the computer on which Cast to TV is installed and the port number (port 4000 in the standard configuration). Next, go to *Cast Settings* on the sending computer and change the *Receiver type* under the *Main* tab from *Chromecast* to *Playercast app*. As the *Listening port*, accept the default *4000*.

Streaming with Cast to TV is now no longer any different from streaming with a Chromecast dongle. To select the desired media files, use *Cast Media* from the Gnome panel. The Gnome add-on then immediately transfers the data to the system equipped with Playercast – whether pictures, music, or videos. Again, an additional menu in the panel provides controls, but it is now labeled *Playercast*. The elements in the dialog are no different from those of the Chromecast variant.

Calling Playercast manually at every start-up is not very practical, especially if you install the system on a Raspberry Pi (e.g., in an attempt to make a "dumb" TV "smart"). Therefore, you should set up Playercast as a system service after completing the initial tests (Listing 1). Now, the software will start automatically when you power on the computer, which you can hide behind the TV set without a keyboard or mouse.

HDMI-CEC

Playercast cannot be controlled by a smartphone, but HDMI-CEC (Consumer Electronics Control) lets you forward control commands from your TV's remote control to devices connected over HDMI, so you can control playback without walking over to the streaming computer. The Raspberry Pi's Raspbian operating system supports this technology out of the box after you install the *cec-utils* package.

Conclusions

Thanks to Cast to TV, Chromecast integrates almost seamlessly into the Gnome desktop. At the push of a button, you can stream local content to your TV set or a stereo system equipped with Chromecast Audio. Some audio/video receivers even support Chromecast out of the box, and TV sets with Android TV definitely do. Thanks to the Cast to TV Links add-on (see the boxout) you don't even have to reach for your mobile phone if you want to watch YouTube and other online streaming video on your TV.

Even without a Chromecast device in the house, Cast to TV has huge potential. In cooperation with Cast to TV, Playercast adds the functions of a Chromecast dongle to a Raspberry Pi or any other Linux-compatible computer. The software is easy to install and fulfills its purpose without too much work. ■■■

Listing 1: Playercast as a System Service

```
### Enabling the system service:
$ playercast <IP address>:<Port> --name '<Room>'
  --create-service
$ systemctl --user enable playercast
$ systemctl --user start playercast
### Removing the system service:
$ systemctl --user disable playercast
$ systemctl --user stop playercast
$ playercast --remove-service
```

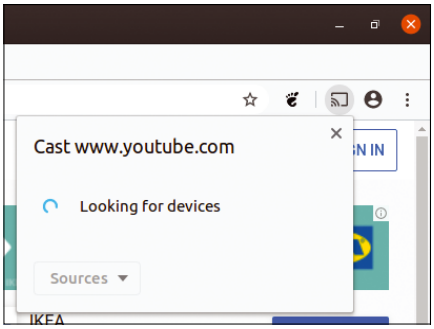


Figure 5: The desktop cannot yet be streamed with Cast to TV; you still have to use the function integrated in Chrome or Chromium, although the solution did not work with Wayland at press time.

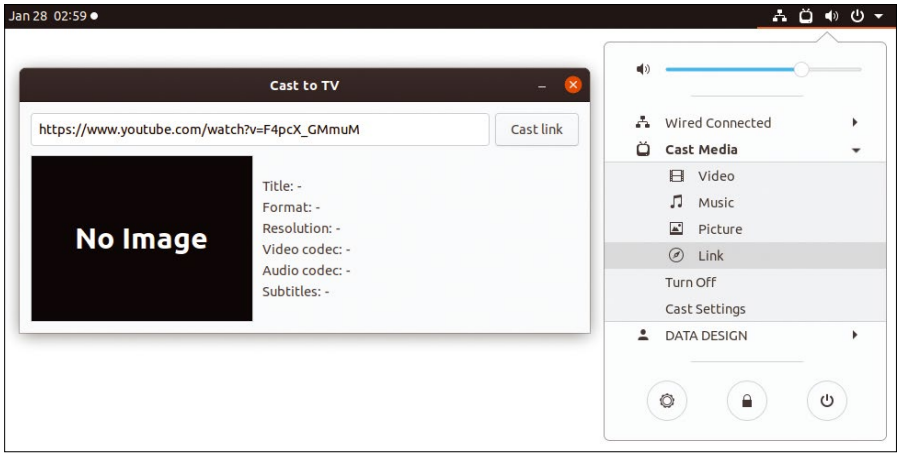


Figure 6: The optional Cast to TV Links add-on lets you stream web videos to your TV from YouTube or other video portals.

Cast to TV Links Add-on

Sending web content directly to the Chromecast receiver with Cast to TV is still somewhat experimental. To do this, you have to install an additional Gnome extension, the Cast to TV Links add-on [9]:

```
$ cd /tmp
$ git clone https://github.com/Rafostar/
cast-to-tv-links-addon.git
$ cd cast-to-tv-links-addon
$ make install
```

Log out of the desktop and back in again and use *Extensions* / *Optimizations* to activate the new Links add-on.

To load a web stream from the Internet, the add-on uses the `youtube-dl` [10] command-line tool. Make sure you install it up front with

```
sudo apt install youtube-dl
```

(on Ubuntu). As a final step, open *Cast Settings* from the Gnome panel and install the missing npm modules in the *Modules* tab by selecting *Install npm modules*.

Now all web videos from all sources supported by `youtube-dl` [11] can be sent to the Cast to TV receiver, including YouTube, Vimeo, TikTok, and others. A new *Link* entry shows up under *Cast Media*. On opening, a dialog appears where you can copy and paste a link (Figure 6). A click on the *Cast link* button transfers the video.

Info

[1] Chromecast: *https://www.google.com/chromecast/*

[2] Gnome Extension Cast to TV: *https://extensions.gnome.org/extension/1544/cast-to-tv/*

[3] Gnome Shell integration for Chrome: *https://chrome.google.com/webstore/detail/gnome-shell-integration/gphhapmejobijbbhgpjhcjognlahblep*

[4] Gnome shell integration for Firefox: *https://addons.mozilla.org/firefox/addon/gnome-shell-integration/*

[5] Integrating Gnome Shell extensions in the browser: *https://wiki.gnome.org/Projects/GnomeShellIntegrationForChrome/Installation*

[6] Cast to TV on GitHub: *https://github.com/Rafostar/gnome-shell-extension-cast-to-tv*

[7] Playercast: *https://rafostar.github.io/playercast*

[8] MPV media player: *https://mpv.io*

[9] Cast to TV Links add-on: *https://github.com/Rafostar/cast-to-tv-links-addon*

[10] youtube-dl: *https://ytdl-org.github.io/youtube-dl/index.html*

[11] Sites supported by youtube-dl: *https://ytdl-org.github.io/youtube-dl/supportedsites.html*

An easy-to-use media server for your home network

Media Distributor

Home media servers like Kodi or LibreELEC are feature-rich but difficult to set up. Serviio promises to make things simpler. *By Erik Bärwaldt*

A media center is a computer or computer-like system that plays multimedia content to a playback device. Many media centers are configured to stream media files (music, video, or images) to a home television set, but you can also use a media center to stream to a game console, mobile device, or computer. The FOSS universe includes several Linux distributions that are designed to serve in the media center role – including Kodi and LibreELEC – but some users consider these tools too complicated for a casual home environment. The complex functions of the common media center can overwhelm many users and often require advanced knowledge.

If you are looking for a media server for your home network that sets up quickly, is immediately ready for operation, and streams content to any computer or DLNA-compatible TV, game console, or MP3 player, then Serviio [1] might be the right choice for you. The Java-based Serviio media server comes from a fairly young company, Six Lines Ltd., based in the county of Northamptonshire in England.

The Serviio software is subject to a proprietary license. You can choose between a free version and a commercial Pro version, which costs a one-off fee of $25. Serviio Pro has some additional functions not available in the free version, such as a web-based media browser and an app for Android devices. Serviio uses free libraries, such as FFmpeg, x264, and the LAME MP3 encoder.

The Serviio server also supports less common video and audio formats and understands popular image, playlist, and subtitle formats. You can pick up the server as a tar archive of only about 30MB from the project's website. The download version is the Pro edition, but it will automatically

switch to the free version after 15 days if you do not purchase a license.

Installation

First unpack the downloaded Serviio tarball in a subdirectory of your choice. Then change to the `serviio-2.0/bin/` directory, where you need to call the `./serviio.sh` script. As a prerequisite, you need to have installed the FFmpeg libraries and a Java Runtime Environment. If these components are not in place, the script terminates and displays an error message; otherwise the server starts – but without reporting successful completion.

Now, switch to your web browser, and then enter the following in the address bar to launch the configuration interface:

```
http://localhost:23423/console
```

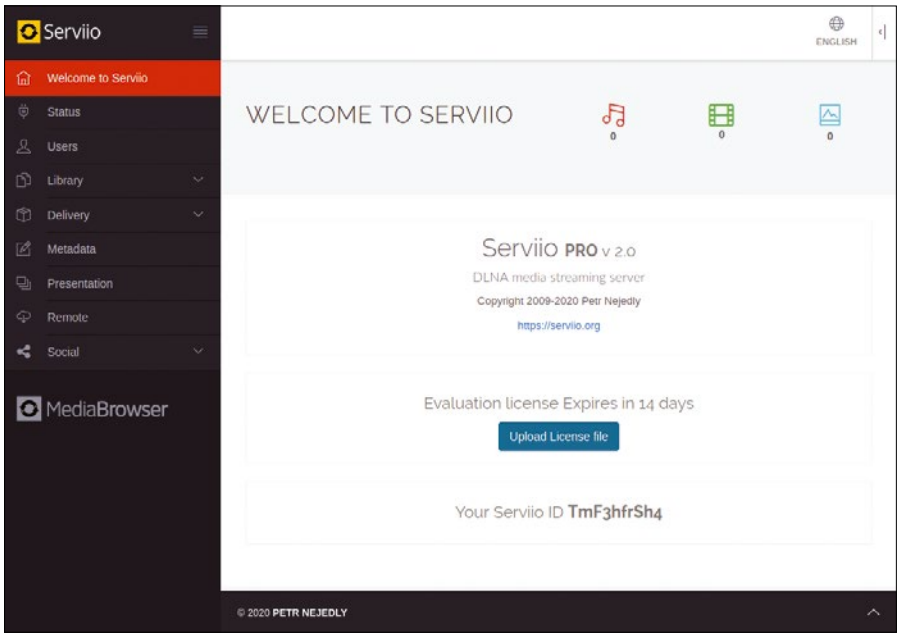


Figure 1: Using the dashboard to configure the Serviio media server.

If you replace `localhost` with the IP address of the server, you can access the dashboard from any other computer on the network (Figure 1).

The command center is tidy and has a modern, split interface. Choose a menu option on the left, and the various settings associated with the menu item appear on the right.

Content

To make your content available via the server, select the *Library* option in the menu on the left of your screen. Choose *Shared folders* to add the local subdirectories where your multimedia content will reside.

To add the subdirectories, click on *Add* and, under *Media Type*, first define whether you mean pictures, movies, or music. If the directory contains mixed media content, you can enable multiple media types. Then click on *Browse…* to the right of the folder display below to open a file manager in which you can select the folder on your system you wish to share. A click on *OK* enters it in the folder display. Please note that Serviio also recursively adds the subfolders. To make sure that more meaningful names appear in the list instead of the directory names, assign a suitable name in *Display name*.

After entering all the folders, click on *Save* at bottom center in the window; Serviio now refreshes the display. If you want to make modifications to a folder setting or delete individual folders, click the small triangular icon to the right of the list entry next to *Edit*. From the list, select whether you want to edit or delete the folder; deletion happens without a prior confirmation prompt. If you delete accidentally, you can undo the action by clicking on *Reset* at the bottom of the window.

By default, the software permanently checks the contents of the defined directories for up-to-dateness and refreshes the display if necessary. If you don't want this, uncheck the boxes for *Update shared files* and *Update library automatically*. The server is then ready for streaming (Figure 2).

To integrate online services, use the *Library* option in the *Library* menu. After calling the *Add* option, you can integrate sources from RSS feeds or HTTP streams in Serviio. To integrate online sources, you just need to save the URL, including the media type. A meaningful name and a preview for the display round off the options.

Serviio also offers you the option of restricting access to the content. Choose the *Users* group on the left side of the main

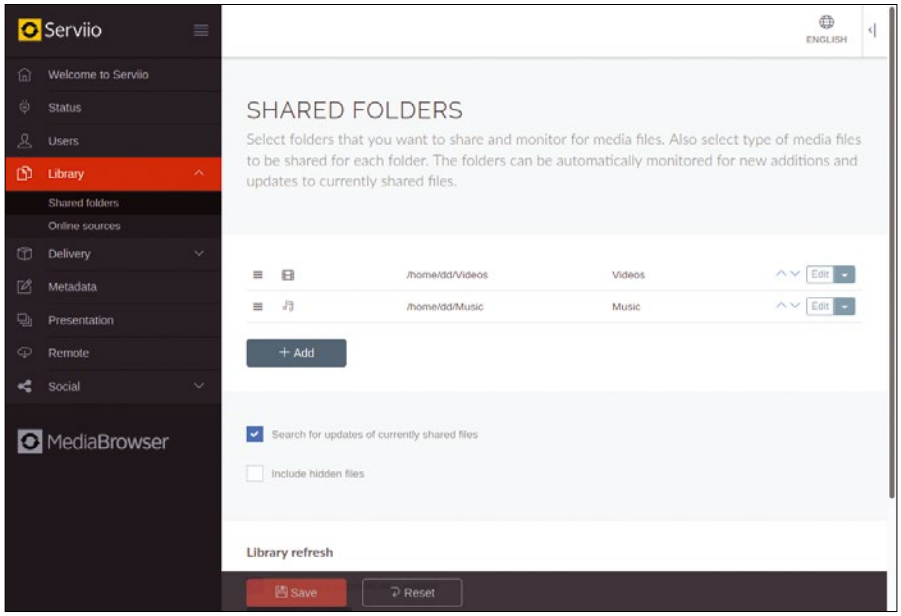


Figure 2: Serviio manages the contents of local directories in *Library* and makes them available for streaming.

Profiled

Serviio comes with profiles for various end devices, which means that it cooperates without any problem with many TV sets, as well as with popular gaming consoles. If the end device does not play a file format supported by Serviio, the server detects this from the profile and transcodes the file during the streaming process so that the device receives a playable data stream.

If no profile exists for the playback device, you can create a separate profile for it in the `config/profiles.xml` of the server installation directory. The manufacturer provides detailed instructions for creating a profile [2]; however, the tutorial does presuppose advanced knowledge of codecs and playback devices.

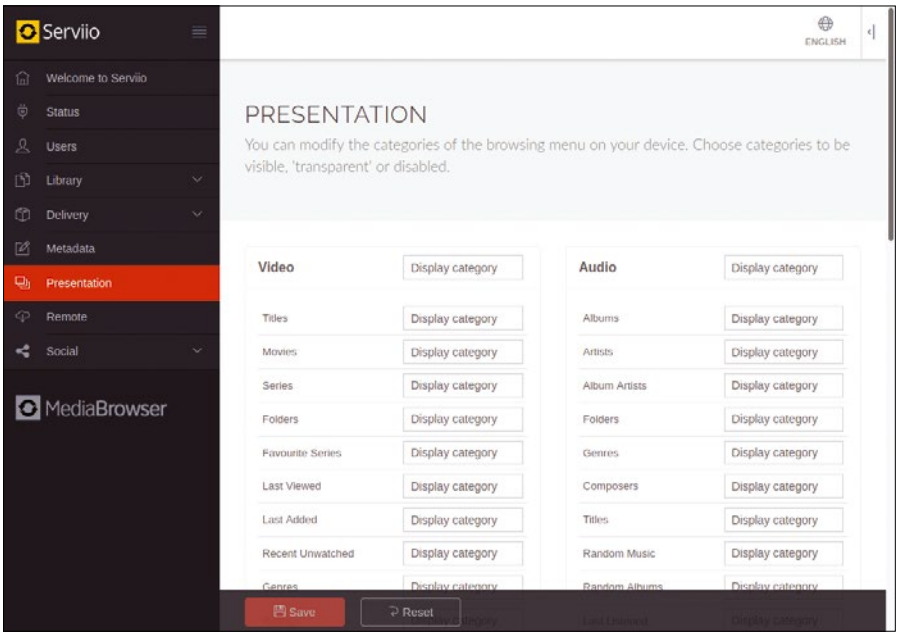


Figure 3: The display of the *Presentation* categories in the client can be customized to suit your needs.

window. The corresponding settings menu on the right side of the window lets you define users and passwords. In the directory settings, check the boxes to determine which users are granted access to what content. The users then need to log on to the server in a login dialog if they want to access the shared data directories.

Transcoding

For those devices on the network that do not support the file format of the source streamed by Serviio, the server offers the option of transcoding the content on the fly. However, this option is only possible for devices for which Serviio has its own profile, usually TVs and game consoles (see the box entitled "Profiled"). PCs that work with DLNA-compatible software do not benefit from transcoding.

By default, content transcoding is enabled. You can set individual options for this in the *Distribution* | *Transcoding* dialog. Specify the storage path for the temporary files and the number of CPU cores you want to dedicate to transcoding. Some settings can also be made for audio tracks. Note that, for high-definition videos, sufficient storage capacity must be available for the temporary files. Otherwise, the stream might be interrupted due to capacity bottlenecks, especially if you are streaming high-res movies.

The *Subtitles* and *Languages* menu items let you set different options for handling multilingual movies and movies with subtitles. Serviio reads both fixed and variable subtitle tracks. In the language settings, you can define the preferred language selection for multilingual video films, both for audio and subtitle tracks. Note that for fixed subtitle tracks, Serviio will always retranscode when you switch tracks.

The *Metadata* group lets you use preview images to make the list of your content clearer and more attractive. The server loads the preview images, which might include DVD or Blu-ray covers, from various online databases as required.

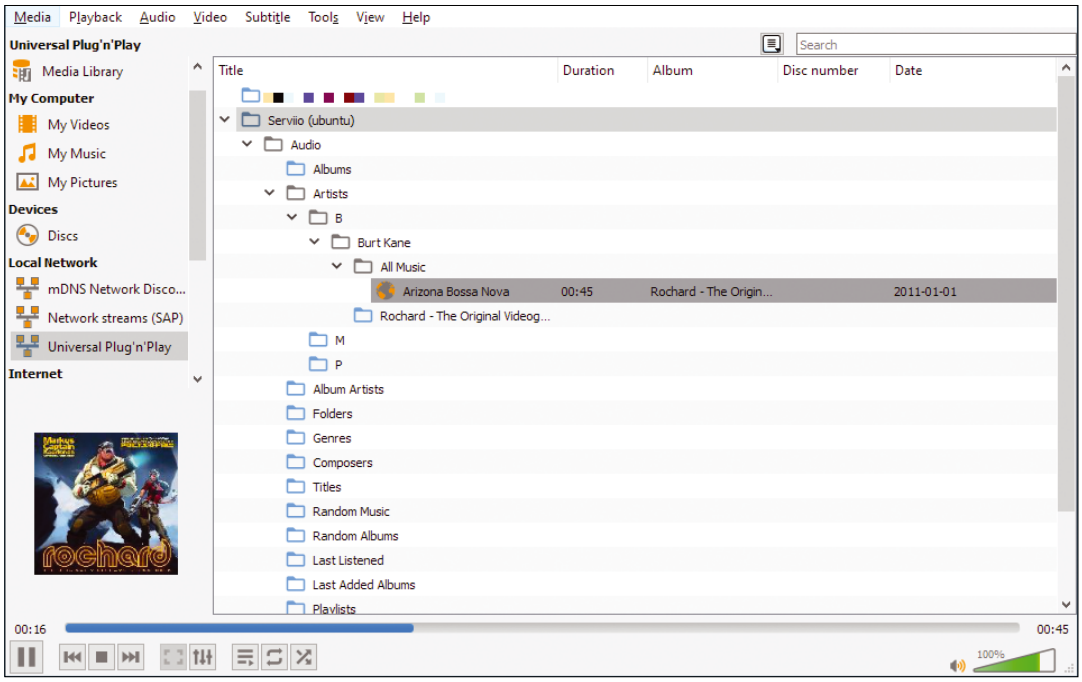


Figure 4: DLNA helps VLC handle content from the Serviio server.

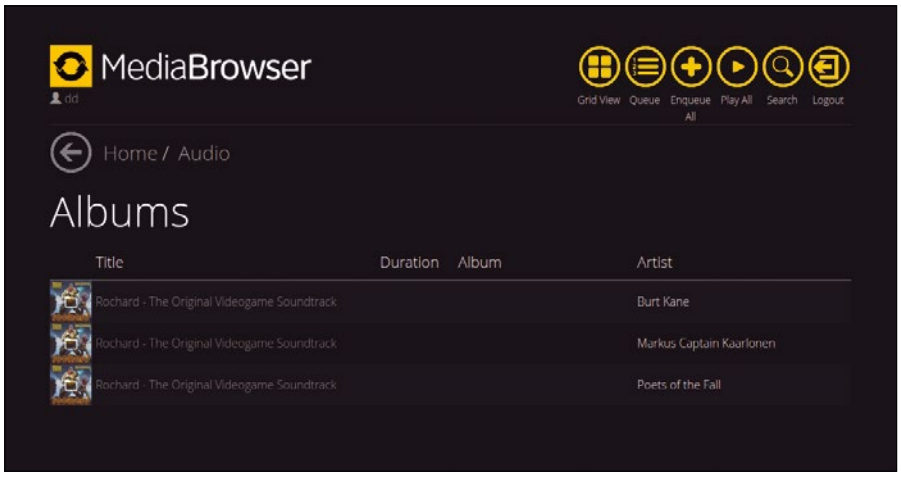


Figure 5: The media browser integrated in Serviio lets you play back multimedia content in a conventional web browser.

Views

The *View* menu item lets you modify how the server contents are displayed on the clients. The software provides categories that appear on the clients. The categories can be configured separately for *Video*, *Audio*, and *Pictures*. A selection field is provided to the right of each category, in which you will find the three options *Show Category*, *Show Content Only*, and *Disable*. Any changes you make will affect both external media players and the software's built-in media browser (Figure 3).

Playback

Once the server is configured, you can access it from any playback-capable device on the LAN. Several DLNA/UPnP-compatible programs are available for Linux. The VLC player, for example, is suitable for retrieving videos from the Serviio server on Linux. Other Linux applications with DLNA support are the Totem Player, which is available on the Gnome desktop, the Rhythmbox audio player, and the eezUPnP multimedia player.

After opening the software, launch the source manager by pressing the keyboard shortcut Ctrl + L. You will now find a list of all available sources in a column on the left. A click on *Universal Plug'n'Play* opens the search for the Serviio server on the network and shows its shared directories in a tree structure on the right. You can open the tree hierarchy by clicking on the right arrow; select the medium for playback from the desired folder (Figure 4).

The *Stack* category on the left side of the configuration menu lets you play longer movies divided into several files as if they were a single file. This feature is especially useful for high-resolution and therefore storage-intensive Blu-ray media.

Media Browser

If your system does not include a media player for DLNA, the Serviio Pro version also lets you playback the stream in a web browser. You can access the media browser via the URL `http://<Server IP>>:23424/mediabrowser`. Log on to the browser with a user account created on the server.

Afterwards, you'll see a simple interface with three tiles. A click on one of the tiles opens either *Pictures*, *Movies*, or *Music*. Using the tile overview that then appears, you can select the desired content based on various criteria. However, the media browser does require you to share the desired folders with the logged-in user; otherwise, only empty categories appear.

You can open the actual playback window by clicking on the desired file. The server then starts the stream, which can be stopped or paused at any time. Add files to a playlist by clicking on the plus symbol that appears in the file previews; display the files by clicking on *Queue* at the top right of the main window.

Click on *Play All* again to play the playlist (Figure 5).

Conclusions

Serviio is a user-friendly solution for a home media player. Serviio doesn't just act as a DLNA/UPnP server – it harmonizes with various playback devices, such as TVs and game consoles. In order to play streams on computers without a DLNA-capable player, the Pro version also integrates a web-based player that you can open in any modern browser. ■■■

Info

[1] Serviio: *https://serviio.org*

[2] Creating a profile: *https://serviio.org/index.php?option=com_content&view=article&id=16*

Spotifyd delivers beautiful music without the bloat

Retro Charm

So you like to travel light and work at the command line? Why not access Spotify in the terminal and do without the official client? *By Christoph Langner*

Spotify has offered an official client for the Linux desktop for years [1]. Although no major functional differences exist between the versions for Windows, Mac OS X, and Linux, some in the FOSS community have criticized the Spotify desktop client – not just because it occupies more than 280MB on the hard disk, but also because of its license terms, which prohibit free distribution. The non-free license means that Spotify cannot be directly integrated into the package sources of the popular Linux distributions [2].

Some users who wish to avoid the Spotify client tune into the service via a web browser. The web browser must support the digital rights management (DRM) implemented in Spotify. However, the Linux environment does include some music players, such as Volumio, that support streaming of Spotify [3]. Especially in the Linux world, however, many users like their programs lean and don't consider a web interface much of an improvement over a desktop GUI. If you're a Linux user who would rather operate from the command line, you can still play your favorite tunes on Spotify. This article introduces you to some useful tools for playing Spotify in the terminal window.

Spotifyd

Spotifyd [4] is a lean Spotify client written in Rust [5]. The Spotifyd client requires access to the Spotify Connect service, which means you'll need a commercial premium Spotify account. If you only use the streaming service via an ad-funded account, you cannot use Spotifyd. In addition to running on Intel-based Linux systems, Spotifyd also runs on the ARM-based Raspberry Pi, which makes it easy to deploy an unobtrusive RaspPi as part of the streaming environment. Spotifyd uses only 3MB RAM. Because Spotifyd supports the Spotify Connect protocol, you can control it from other Spotify clients, which means you could connect the Spotifyd client system to your TV or stereo system, and then control it from another computer or Spotify device.

The developers are currently working very intensively on Spotifyd. Packages are only available for Arch Linux (*spotifyd-full*; see the "Installation on Arch Linux" box) and FreeBSD. Users of other distributions need to compile the program. Listing 1 shows the process for Ubuntu 19.04. Compiling takes a long time, especially on older

Listing 1: Build in Ubuntu

```
$ sudo apt install git rustc cargo libasound2-dev libssl-dev pkg-config
$ git clone https://github.com/Spotifyd/spotifyd /tmp/spotifyd
$ cd /tmp/spotifyd
### Option 1: Build and Run Locally
$ cargo build --release
$ ./target/release/spotifyd
### Option 2: System-Wide Installation
$ cargo install --path .
```

Listing 2: Add to Path

```
# set PATH so it includes user's
# cargo bin if it exists
if [ -d "$HOME/.cargo/bin" ] ; then
    PATH="$HOME/.cargo/bin:$PATH"
fi
```

Listing 3: spotify.conf

```
[global]
username = username
password = password
bitrate = 320
backend = alsa
cache_path = /var/tmp/spotifyd
volume-normalisation = true
normalisation_pregain = -10
```

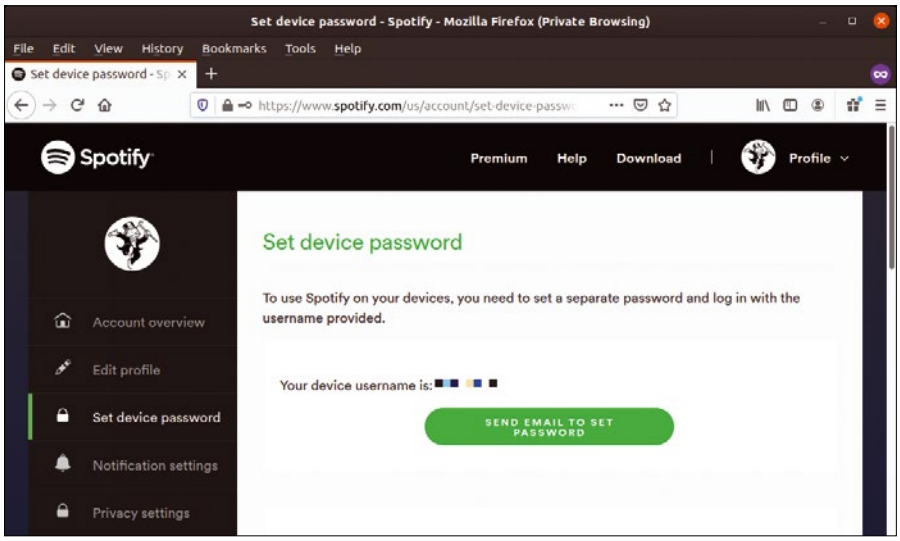


Figure 1: In order for Spotifyd to access the streaming service, you need commercial premium access. A device username and password can then be created from the Spotify settings.

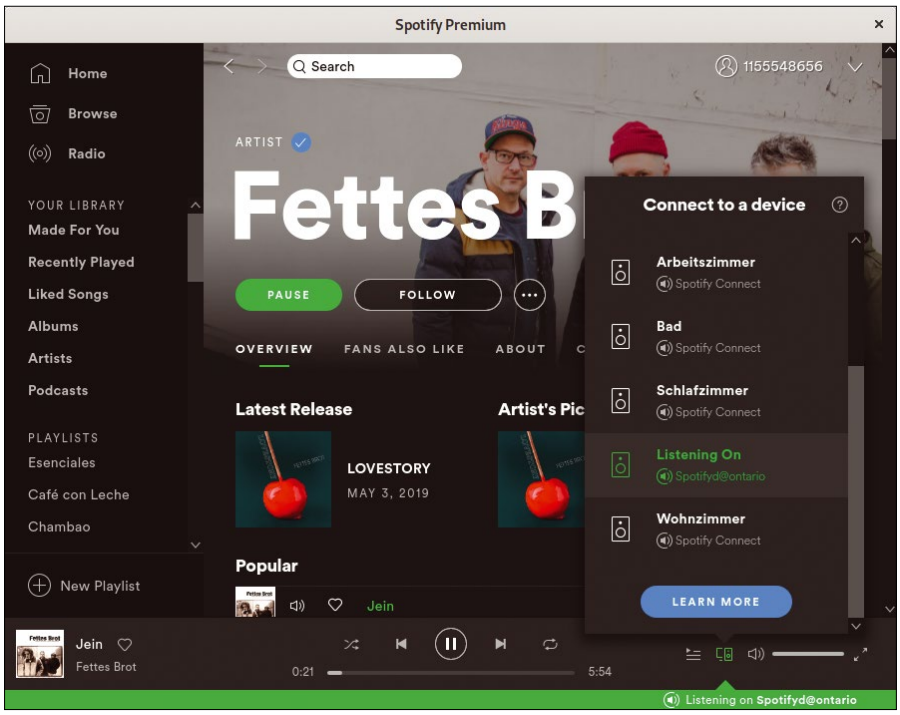


Figure 2: Spotifyd can also be controlled from the official Spotify application. In the list of devices connected with Spotify Connect, it appears as *Spotifyd@<computer_name>*.

computers. I recommend you choose option 2, which installs the compiled program to `~/.cargo/bin`. To run Spotifyd, add the lines from Listing 2 to `~/.profile`, restart the terminal program, and then run the `spotifyd --version` command to test.

The service is configured via the `spotifyd.conf` file in the `~/.config/spotifyd/` folder. Create the folder, and then create a file in it with the contents from Listing 3. The user name and password for Spotify Connect are available at the Spotify website [6] (Figure 1).

To test the configuration, call the `spotifyd --no-daemon` command. The command starts the service and should only print `INFO` lines. Afterwards you can stop Spotifyd by pressing Ctrl + C. After a successful test, it is best to start the service via systemd. Later on, you can use systemd to configure Spotifyd to automatically start and shutdown when you boot and power down your computer (Listing 4).

If you have installed Spotifyd on a small headless computer like the Raspberry Pi that is connected to a stereo system, you can use the official Spotify app from a smartphone or PC to redirect playback to the Spotifyd-equipped computer using the icon to the left of the volume control and the *Spotifyd@<computer_name>* menu item (Figure 2). You can also use this approach with Spotify-compatible network speakers, such as the Sonos system.

Listing 4: With systemd

```
### Start and stop service
$ systemctl --user start spotifyd.service
$ systemctl --user stop spotifyd.service
### Activate/Deactivate Service at System Start
$ systemctl --user enable spotifyd.service
$ systemctl --user disable spotifyd.service
```

The `spotify-tui` command-line tool [7] can act as an interface for the Spotifyd client. `spotify-tui` serves as the music center, with Spotifyd handling the playback. Like Spotifyd, `spotify-tui` was developed in Rust. Listing 5 shows to install `spotify-tui` in Ubuntu using the Rust package manager `cargo` [8].

Listing 5: Installing with cargo

```
$ sudo apt install cargo libssl-dev
$ cargo install spotify-tui
```

Before the program can talk to the streaming service via the Spotify Web API [9],

you have to go through the configuration, which involves more than just entering an account. To access the API, you need to register as a developer [10] and create an "app" – don't worry, you don't have to do any programming. All you have to do is click on *CREATE A CLIENT ID* and enter some basic information (Figure 3). In the next step, click *NO* to promise that you have no commercial intentions and finally accept the terms of use.

Then select the newly created Spotify app from the *Dashboard* with a left click and open the settings via *EDIT SETTINGS*. Now enter the address `http://localhost:8888/callback` as the last administrative task in *Redirect URIs*. Write down the *Client ID*, and – after a click on *SHOW CLIENT SECRET* – the corresponding secret password for the client (Figure 4).

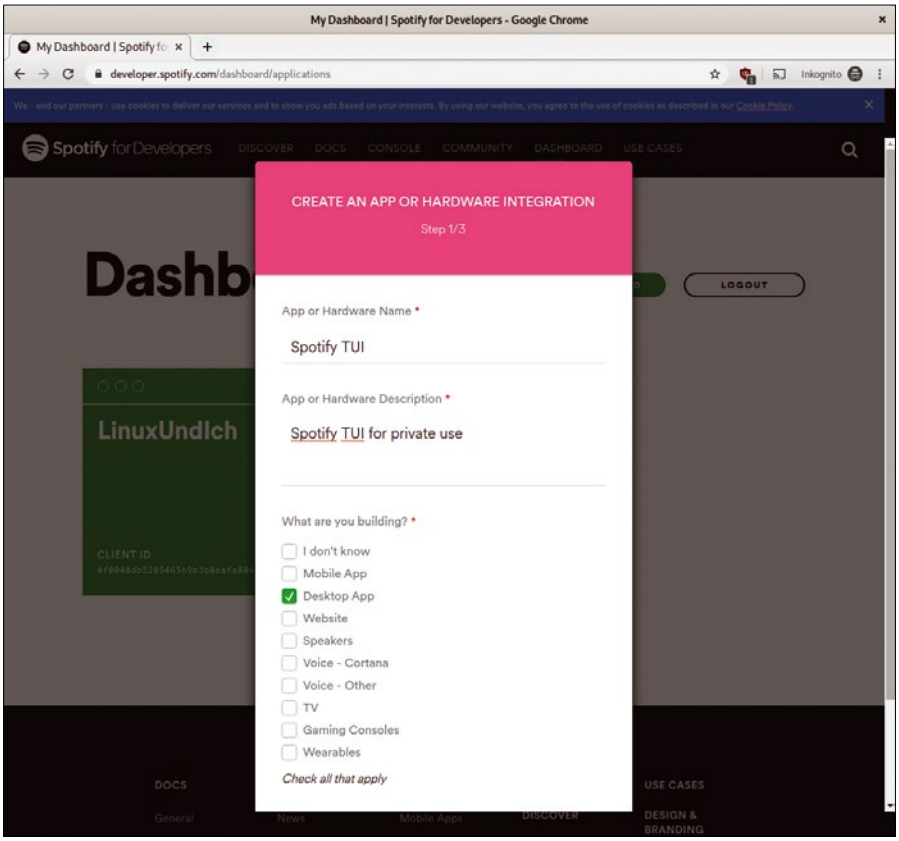


Figure 3: To access the Spotify Web API, you need a developer account and your own Spotify app for `spotify-tui`. Don't worry: You don't have to program anything; a few mouse clicks are all it takes.

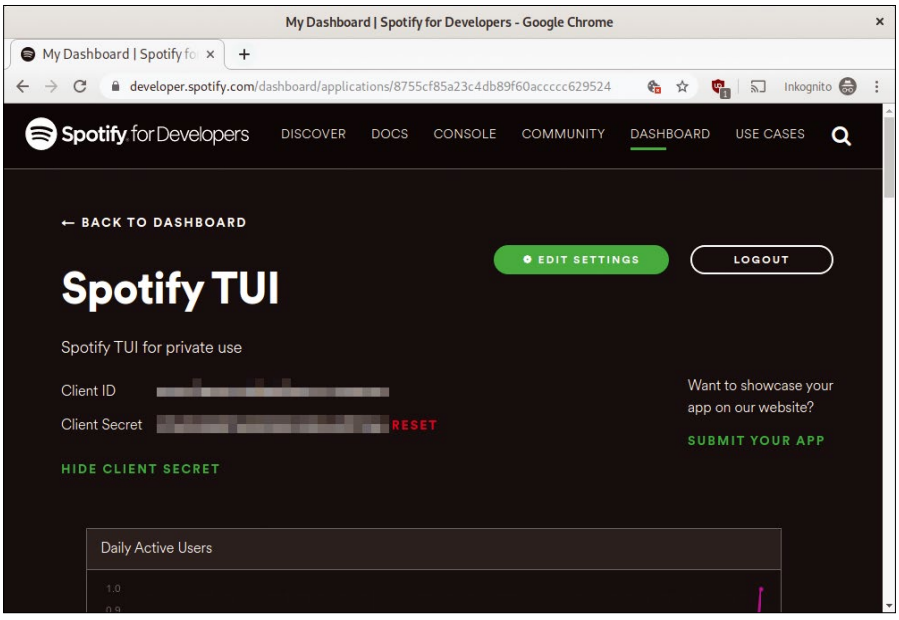


Figure 4: From the Spotify dashboard you need to transfer the *Client ID* and the *Client Secret* to set up `spotify-tui`. Do not forget to adjust the *Redirect URIs* under *EDIT SETTINGS*.

Armed with this information, you can now start `spotify-tui` with the `spt` command. The program wants to know the *Client ID* you just created, along with the appropriate *Client Secret* (Figure 5). After that, the procedure gets a bit confusing: The program tries to open an extremely cryptic and long URL of the type *http://localhost:8888/callback?code=AQt[...]pdjW*, which throws an error, because you do not usually have a web server running on your own system. Ignore the error message and simply copy and paste the address of the page from the browser into the setup of `spotify-tui`.

In the first dialog after the setup, the program searches for Spotify Connect devices on the local network, including all smartphones and PCs running the Spotify app. Spotifyd should appear in the list of *Devices* as *Spotifyd@computer_name*. Select this entry using the arrow keys and press Enter. If necessary, enlarge the terminal window, otherwise texts and elements often do not appear on the screen (see the box entitled "Tips"). `spotify-tui` then opens the actual application window, in which you will find a search bar at the top and sidebars with your Spotify library and playlists on the left (Figure 6).

In `spotify-tui`, you can use the arrow keys to navigate through the individual elements of the interface. The program always color-highlights the

Installation on Arch Linux

In the Arch User Repository (aka the AUR), the Arch Linux community maintains a collection of recipes from which a package for the Pacman package manager on an Arch system can be easily compiled using a helper program. Both Spotifyd and `spotify-tui` are already included in the AUR. If you work with the currently recommended AUR helper, Yay, you can install both programs at once with the command `yay -S spotifyd-full spotify-tui`.

Tips

`spotify-tui` tries to transfer the Responsive Design concept used with websites into the terminal. Depending on the size of the terminal window, the program shows and hides elements of the text interface and removes explanatory text. In practice, however, this approach proves to be tricky. For example, in a standard terminal with an 80 characters width and 24 lines height, the explanatory text is already completely missing in the first screen in which you need to select the Spotify device. Ideally, you should therefore work with a terminal maximized to the full size of the screen, especially at the beginning. Also when choosing the colors, you have to adapt a little to the preferences of the `spotify-tui` developers. The application uses hard-coded colors that only work well on a terminal with light text on a dark background.

current selection. To open the element currently highlighted, press Enter. Press Esc to go back one level. Much like text tools such as Less, you start a search in the Spotify library by entering `/search term`. `spotify-tui` then immediately displays the matches in the tracks, artists, albums, and playlists in the central section of the application. You can get help by entering a question mark.

As soon as you play a song, another box appears at the bottom with information about the song currently playing and a progress bar. The space bar can be used to pause and resume playback, but it is not possible to jump to a specific point in the song. Playback so far only offers a random mode, which you can activate with Ctrl + S. If necessary, press *D* to return to the Spotify device selection – this way you can use `spotify-tui` to control multiple devices at once. Press *Q* to exit the application. Since the playback relies on Spotifyd, the currently active playlist continues to play.

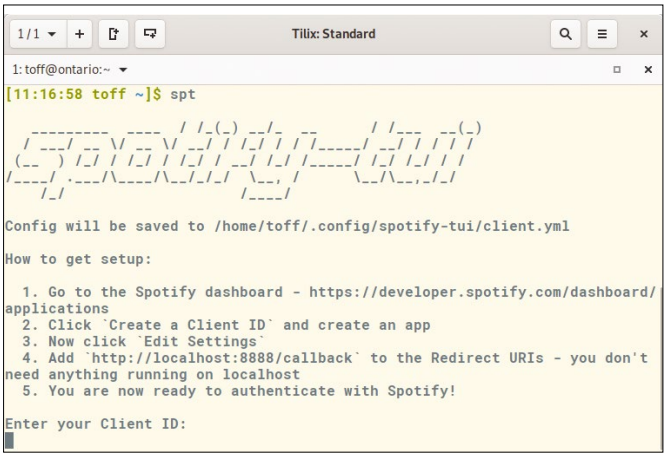


Figure 5: When first launched, `spotify-tui` wants to know the *Client ID* and the *Client Secret*. The resulting URL tries to open the setup in a browser. In the last step, you transfer this address to the terminal window.

Conclusions

You don't necessarily have to use the duo of Spotifyd and `spotify-tui` together. For example, you can install Spotifyd on a Raspberry Pi that you have connected to a stereo system, and then control the music through the official Spotify application or, if necessary, through `spotify-tui` on another computer. Alternatively, control the music playback from the terminal using the official Spotify app.

On systems that are not completely under your control, it is a good idea to enter the access data for Spotify in plain text in the configuration. The GitHub page for Spotifyd describes how to store the account information in a secure keyring. Otherwise Spotifyd and `spotify-tui` leave plenty of scope for your own experiments. Both programs are backed by active developer communities that react quickly to bug reports and suggestions for improvement. ■■■

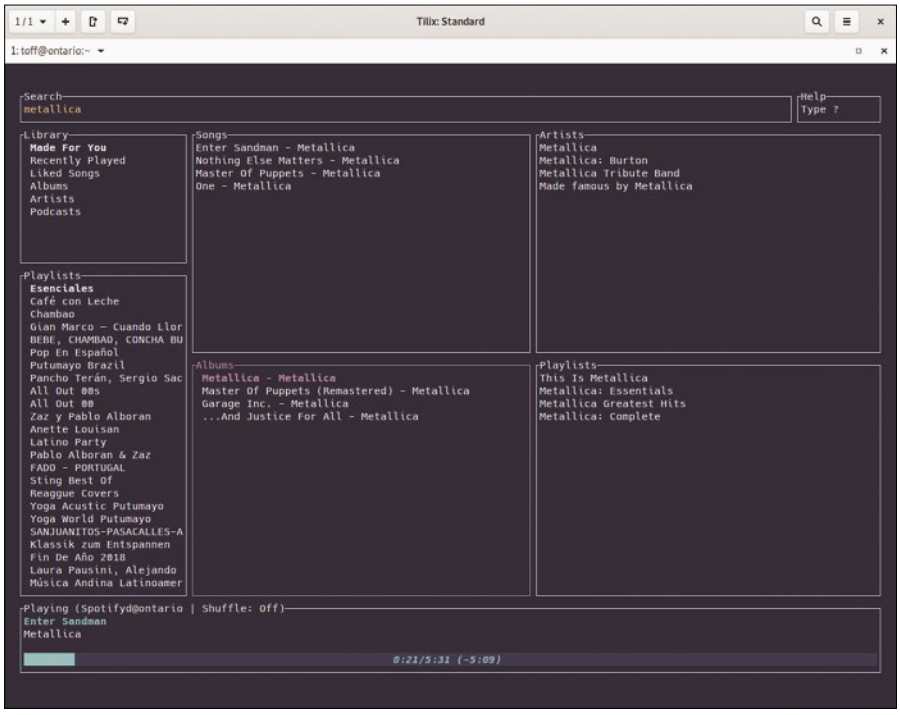


Figure 6: `spotify-tui` serves as the music center, but Spotifyd is responsible for the playback. You can exit the program without the music playback stopping.

Info

[1] Spotify for Linux: *https://www.spotify.com/de/download/linux*

[2] "Redistribute Spotify on Linux Distributions": *https://community.spotify.com/t5/Live-Ideas/Redistribute-Spotify-on-Linux-Distributions/idi-p/1695334#M188735*

[3] Volumio: *https://volumio.org*

[4] Spotifyd: *https://github.com/Spotifyd/spotifyd*

[5] Rust: *https://www.rust-lang.org*

[6] Spotify Connect: *https://www.spotify.com/connect*

[7] spotify-tui: *https://github.com/Rigellute/spotify-tui*

[8] Rust Package Registry: *https://crates.io*

[9] Spotify Web API: *https://developer.spotify.com/documentation/web-api*

[10] Spotify for Developers Dashboard: *https://developer.spotify.com/dashboard/applications*

Exploring the Kubuntu Focus laptop

In Focus

If you're a gamer, a developer, or a Linux power user who appreciates powerful hardware, you might be ready for the Focus – a high-end Linux laptop built and tested for Kubuntu. *By Rick Timmis*

The Kubuntu Focus [1] is a high-performance laptop outfitted and optimized for maximum compatibility with Kubuntu and its KDE desktop. The Focus, which is produced by the Kubuntu Council, Tuxedo Computers, and MindShare-Management Inc., is the brainchild of MindShareManagement CEO Mike Mikowski. I first heard about this initiative through my role as Councilor for the Kubuntu project. Mike had approached the council via former council member Ovidiu-Florin Bogdan, with whom I co-host the Kubuntu Kafe live stream meetup and podcast.

The idea of the Focus is to provide a highly polished, high-performance laptop computer targeting workflows for web designers, developers, and professional creators. The goal is to offer a "Power out of the Box" experience beyond that of the PC and MacBook.

Unboxing

The unit arrived just two days before Christmas Eve (Figure 1), and I was able to record an unboxing video complete with Christmas tree and festive hat [2].

The Focus comes in a full-color printed box, wrapped in protective bubble wrap and surrounded by an exterior shipping box. Opening the box revealed further attention to detail, with a full-color printed Getting Started guide, and a welcome pack that included a Kubuntu-branded USB recovery key.

Shipping from the US meant that my device came with an American power lead for the power supply. However, I discovered that the laptop had 70% charge so I was able to experience first boot using the onboard battery. US and European orders will be fulfilled via separate distribution centers, thus production models will all come with a power lead appropriate for the destination country.

Figure 1: My Kubuntu Focus laptop arrived just in time for the holidays.

Lead Image Photo by Kevin Ku on Unsplash

Two features impressed me at power on:

1. The beautiful backlit keyboard, which lights up in a shimmer of rainbow colors.
2. The Kubuntu logo on the BIOS splash screen – a nice touch that demonstrates a commitment to the Kubuntu brand and community.

From power on to the decrypt password field is around 6 seconds. Full disk encryption is enabled out of the box, and unlocking the disk realizes the login screen in around 3 seconds, with delivery to the desktop post password entry in a further 6 seconds. This delivers the user to a productive state in less than 20 seconds on power up, and takes less than 6 seconds to resume from RAM. Suspend and resume is triggered through closing and opening of the clamshell case, which is predominantly made of metal.

The Kubuntu Focus is intended for professionals, and the hardware specifications are very robust. The Focus builds around the Core i7-9750H, operating with 6 cores at 4.5Ghz and providing 12 threads of symmetric processing. The order form at the KFocus website offers several hardware options [3]. The base system, which sells for US$1795, comes with 16GB 2666MHz dual channel memory and Samsung Evo Plus 250GB NVMe storage. My Focus includes the upgrade to 32GB memory and 1TB Samsung EVO Plus NVMe 3,500MB/s.

For audio-visual creators and gamers, there is an NVIDIA GeForce RTX 2060 6GB GDDR6 with PhysX and CUDA. However, the Focus goes beyond the gamers with workflows to support data scientists and machine-learning researchers. The Kubuntu Focus comes with Nvidia CUDA, Python 3, Tensorflow libraries, and Jupyter notebooks pre-installed.

Recently, it has become fashionable for ODM manufacturers to remove external connectivity ports, which might be cool but is impractical in the real world. It is very pleasing to see that the Focus hardware team rejects this modern folly. Display output provides HDMI, display port, and USB-C display output, along with a higher-amperage always-on USB-C connector for charging your mobile devices. USB 3 ports occupy the left and righthand side, with Ethernet, Microphone, Headphones SD/MMC, and MicroSD card slots available too.

A pressure-sensitive Glass Synaptics touchpad offers multi-point detection. Two-finger scrolling is enabled by default, along with tap to click. The keyboard has backlighting, with function controls for changing the color scheme and switching the backlights on/off. Brightness controls are available via the number pad when used in conjunction with the function control button.

The laptop's close association with the Kubuntu brand is an indication of the deep level of integration. For instance, the keyboard Meta (or "Windows") key has been laser-etched with the Kubuntu logo (Figure 2), and the outer lid of the clamshell is emblazoned with a laser cut, polished aluminum Kubuntu logo veneered with transparent acrylic gloss (Figure 3).

The desktop defaults to the Kfocus dark theme (Figure 4). As a Kubuntu

Figure 2: The Focus keyboard has the Kubuntu logo on the meta key instead of the Windows logo.

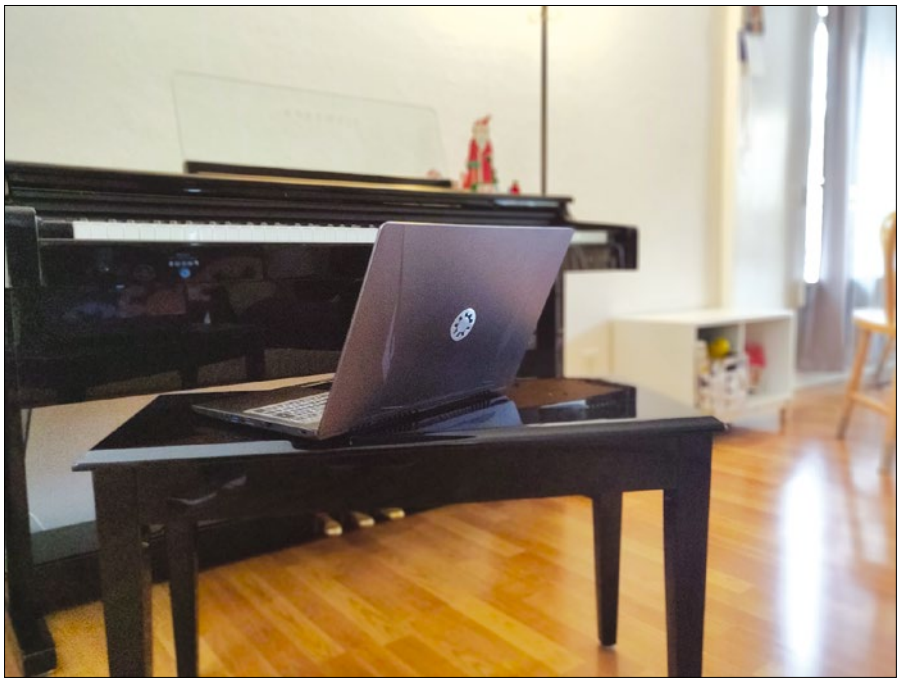


Figure 3: The lid also sports a polished, laser-cut Kubuntu emblem.

Figure 4: The Kubuntu Focus desktop uses the dark theme by default.

user, I am used to the blended dark/light theme that comes with Kubuntu by default. I found the all-dark theme has taken some time to adjust to. However, I made the decision not to change the theme and to leave the address bar on the right – with the kickoff menu at the top right, which all felt really alien to me. However, I have to say that having the menu bar on the right has quickly grown on me. Western cultures tend to read left to right, and I had never given thought to how my eye moves over the screen until now. I have adjusted quickly to having the menu on the right, and furthermore, this has increased the top to bottom usable screen space. I am a convert to this new design, and I genuinely think others will really like this layout too, provided that they stick with it for a few days.

The Kubuntu Focus comes with a wide array of software installed, and the producers are keen to point out via their guidance notes that much effort has been invested by the product team in providing software stacks and tuning the experience for multiple workflows.

As part of my unboxing experience, I decided to try out this workflow theory by producing the actual unboxing video on the unboxed laptop. (The very fact that it sounds paradoxical makes it exciting.) The Focus comes with Kdenlive installed and configured for the hardware, which meant I could simply grab the video cuts from the cloud; drop them into the video editor; then cut, slice, and render. Kdenlive is configured to render with just one core out of the box; I would recommend changing those settings to take advantage of the multiple processors. Rendering the Video to H264 ready for YouTube took about 11 minutes, just a little longer than the unboxing video itself.

Conclusion

As I put the finishing touches to this magazine article, which I have written using the Kubuntu Focus, I have been using this laptop as my daily driver for one month. This includes utilizing it in my professional capacity as Senior Engineering Manager for Boomi – a Dell Technologies Ltd company, as well as my voluntary open source community efforts and my software development projects.

The workflows are simply fantastic – transferring the GoBandit project to the Focus is so simple because the Focus comes with the JetBrains toolbox already installed. Git is installed out of the box, but the laptop producers have been smart enough not to install Go, choosing instead to leave the user to choose between a snap-confined GoLang or the system-wide version, depending on personal preference. My personal preference is a system-wide install.

With all the graphics and compute performance on tap, users should not expect extensive battery life. The manufacturer claims battery life of between 3 and 4 hours. In my experience, I have not been able to get much beyond 2 hours before having to find a power source. However, this is a high-performance workstation, and I wouldn't anticipate the always-connected web surfer or Internet nomad to be considering the Kubuntu Focus as their laptop choice.

Steam and Nvidia drivers installed and configured out of the box, coupled with the high-end performance of this machine, make gaming a lot of fun, and the graphics performance is simply outstanding.

The Kubuntu Focus is built with the care and attention to detail that provides a polish that I have previously only seen from Apple products. My reviewer's edition cost $2,245, which is significantly less than a 16-inch MacBook Pro with 8-Cores and the AMD Radeon pro 550 GPU. Of course, because I am a member of the Kubuntu community, it could be argued that I have a natural bias. However, I have tried to be as objective as possible in my evaluation, and I am giving this laptop a resounding 10 out of 10. ■■■

Info

[1] Kubuntu Focus Project Website: *https://kfocus.org*

[2] Unboxing: *https://kubuntu.org/news/kubuntu-focus-laptop-christmas-unboxing/*

[3] Ordering a Focus: *https://kfocus.org/order/*

Author

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

SUSECON '20

Be the Difference

March 23-27, 2020 | Dublin, Ireland

#SUSECON #BeTheDifference

Elevate your business through increased agility and faster innovation.

Learn how at SUSECON with:

- Access to SUSE leadership and technical experts
- 100+ hours of hands-on training
- Technology showcase
- Certification exams
- 150+ sessions
- Tutorials

Register at susecon.com!

Disaster tolerance with Apache Cassandra

Highly Available

The size and scope of today's Internet companies require more than your average SQL. Apache Cassandra is one of the NoSQL systems filling the need for high availability at scale. *By Aleksandr Volochnev*

Apache Cassandra is an open source NoSQL distributed database that stores and manages large volumes of data on standard servers. Cloud providers use Cassandra for configurations with many data centers spread across global networks.

The story of Apache Cassandra began in 2007 when Facebook engineers Prashant Malik and Avinash Lakshman developed a very early version for Facebook's inbox search. The challenge was to store the data for huge datasets residing on hundreds of servers. A year later, Facebook released Cassandra on Google Code, making it an open source project. In 2009, it joined the Apache incubator, paving the way to it becoming a top-level Apache Foundation project. Since then, many well-known companies have implemented Cassandra or a commercial version (DataStax Enterprise), including Apple, Netflix, Twitter, Sony, eBay, Walmart, and FedEx. Cassandra and other NoSQL alternatives are part of a new generation of data tools designed to fulfill the massive storage needs of the Internet era. A conventional relational database, such as an SQL database, is difficult to cluster, subdivide, or scale horizontally. Companies can either keep their data at a single location and let their customers contend with long wait times to access it remotely, or they can operate two instances of the database. Neither of these scenarios is viable for a modern international company that needs both global data availability and the ability to grow without incurring additional costs. NoSQL systems are built to be extremely scalable. To increase performance, you can simply add additional nodes to the cluster on the fly. To double the performance of the database, you just need to add the same number of nodes as the cluster already has. Apache Cassandra is based on Java and has symmetrical nodes organized in clusters, rather than the master and named nodes used with SQL implementations. Cassandra is useful for real-time data storage for online applications with multiple transactions. You can also use Cassandra as a read-intensive database for business intelligence systems. If you're accustomed to SQL, you'll find that the Cassandra Query Language (CQL) is strongly reminiscent of SQL in terms of syntax and keywords. Cassandra is designed for a distributed environment. To fully implement Cassandra's disaster tolerance capabilities on a massive scale, companies need to distribute the data across different regions or even different cloud providers. If one instance fails, some latency may occur, but the data remains available.

Author

Aleksandr Volochnev is Developer Advocate at DataStax.

CAP Theorem

The CAP theorem is a principle of computer science that helps to explain why NoSQL systems like Cassandra differ from conventional data tools. The CAP theorem (or Brewer's theorem), which describes the relationship between consistency (C), availability (A), and partition tolerance (P), was first articulated by Eric Allen Brewer, Professor Emeritus of Computer Science at University of California, Berkeley and Vice President of Infrastructure at Google. CAP forms the basis for planning a distributed architecture. The basic parts of the CAP decision framework are:

- Consistency: "Each read operation accesses the last write operation or an

error." A consistent system returns the same value from each node that is requested.

- Availability: "Each request receives an error-free response." Whatever happens within the cluster does not affect the clients. A highly available system always sends an answer, even if half of the cluster is already dead.
- Partition tolerance: "The system continues to work despite network problems between nodes." A partition-tolerant system continues to run even if there are serious communication problems within the cluster.

The CAP theorem states that no distributed system can fully achieve all three of these objectives. Because a distributed database must continue to operate if the network stops or part of the system is down, the third objective (partition tolerance) is required. That means a distributed database can either be consistent and partition-tolerant (CP) but less available; or it can be highly available and partition-tolerant (AP) and less consistent (Figure 1). These two mutually exclusive options are best understood if you consider the basic trade-offs between consistency and availability. If you wish to maximize availability, the system must continue to receive data when the distributed nodes are not able to communicate with each other (e.g., it can't just stop working and send an error message). But a scenario that calls for a node to provide data to a user when it is unable to verify that the data is up to date does not fulfill the ideal for consistency. On the other hand, if you wish to maximize consistency, the system will need to ensure that the data provided to the user is the latest version or else return an error, which means that, if the nodes cannot communicate, the system would not be fully available.

Cassandra is known as an AP system, because it maximizes availability and partition tolerance at the expense

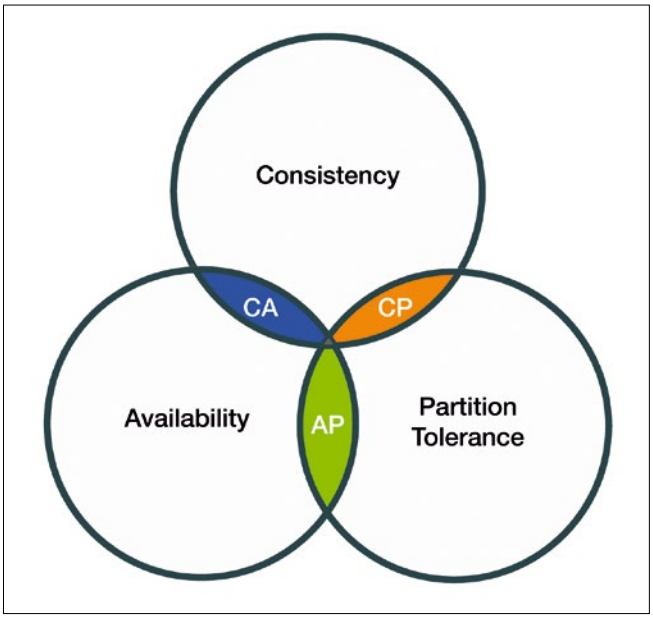


Figure 1: The CAP theorem states that distributed IT architectures cannot simultaneously prioritize consistency (C), availability (A), and partition tolerance (P).

of consistency. Cassandra's developers are willing to tolerate some inconsistency in order to ensure that the database remains available when operating in a partitioned state. This emphasis on availability over consistency is one reason why Cassandra is so highly scalable compared to many conventional database options. The steps necessary to maximize consistency do not scale for multiple nodes and large datasets. However, although Cassandra emphasizes availability in the partitioned state, it does include synchronization features that provide consistency among the distributed nodes in normal operating conditions.

No SPoF

Cassandra's AP design, with its emphasis on availability, requires that the system eliminate all Single Points of Failure (SPoF). (In a relational database, by contrast, each master node is a potential SPoF.) In order to eliminate SPoF, either all components must be designed redundantly or the design must reflect a masterless architecture, in which the nodes are peers. In the case of Cassandra, every node can process a request, no matter if it needs to read or write. If one of these nodes fails, its data must be available at another location – waiting to restore a backup is not an option for a system designed to achieve zero downtime. Instead, the data is provisionally replicated before anyone needs it. Cassandra lets you define a replication factor. If you set the replication to `3` or `5`, each data element is replicated in the corresponding number of nodes. Redundant replication causes additional costs; however, the cost of storage is small compared to the loss of reputation and long-term economic damage associated with lost data. It is also important to remember that replicas should not reside next to each other on the servers. Servers in the same rack tend to fail together. Any event can paralyze not only the rack, but the entire data center. It is therefore advisable to opt for georedundant replication.

Automatic Recovery

More servers means the higher the probability of a failure. A cluster must be capable of restoring itself independently. There are two mechanisms for this restoration:

- Announced redirects: When one node fails, other nodes start keeping updates for the failed node. If the node is regenerated within a reasonable amount of time (typically one to four hours, depending on the configuration), these handover packages are reinstalled on the node, restoring it completely and autonomously.
- Repair/NodeSync: If network delays or similar issues cause problems, a cluster performs a health check and recovery operation. In Apache Cassandra, this is known as a "Repair."

The nodes constantly communicate with one another in order to implement workaround options when needed. In Cassandra, nodes immediately report when a new node enters the cluster or an old node fails. When a client application connects to the database using the specified IP address, it loads the database metadata and prepares to send a request to each subsequent node if the actual target node is not reached. Depending on the setting, an application may repeatedly request other nodes. Each node in the cluster can receive a request for data. A node receiving the request acts as a "coordinator node" and sends the request to the nodes responsible for data. This system requires knowledge of the cluster metadata, which the nodes constantly exchange. Each node knows the cluster schema and the position of all usable nodes.

Adjusting Consistency

Although Cassandra prioritizes around availability at moments of network failure, it does include options for enhancing consistency in normal operations. For each query in an application, you can select a consistency level: `Any`, `One` (`Two`, `Three`), `Quorum`, or `All`. Then a special rule is applied for the query, which is only considered fulfilled if the specified number of nodes confirm the write operation or give the same answer to a read query. The more answers, the more consistency, but the price of pursuing a high level of consistency is a certain loss of performance, because several nodes need to react. High consistency also weakens availability. Configuring your Cassandra system for high consistency tends to push it toward the CP end of the CAP diagram (refer to Figure 1). For example, an update with the consistency level of `All` can be written to the cluster in which three of the replication nodes store the data (`Replication Factor: 3`). If a node does not respond, the procedure is stopped and an exception is thrown. The most commonly used consistency level for important operations is `Quorum`. Accordingly, the write is considered successful if 50 percent + 1 nodes confirm the write operation.

Conclusions

Cassandra is too complex for a typical small to medium-sized project. However, companies that don't want to keep data on a single server, need geographic distribution, and are looking for the highest possible availability would do well to consider the Cassandra option. Apache Cassandra is an excellent tool for companies that wish to quickly create distributed, disaster-tolerant systems. ■■■

DrupalCon
MINNEAPOLIS 2020
MAY 18-22

Minneapolis Convention Center
events.drupal.org/minneapolis2020

We've announced our stellar lineup of curated sessions featuring top minds in open source digital experience.

Solidify your plans. Register **today** to start your amazing DrupalCon journey!

Network, learn and be inspired.

Important Dates

Regular Registration Ends - March 31
Hotel Booking Deadline - April 7

Options for systemctl

Working the System

Every major aspect of a system that runs in userland can be controlled by systemctl, a command that acts on systemd's units. *By Bruce Byfield*

Systemd has replaced init in most distributions for several years. Yet for many users, it remains mostly unknown. One reason for this obscurity is that, to avoid confusion, distributions hide systemd with aliases to traditional commands. It is only when you look into the intricacies of `systemctl`, a systemd command that views and manages a system, that you begin to appreciate how far-reaching systemd actually is.

Like other init systems, systemd's main purpose is to start and control services that must be started after the kernel – those that run in userland, as it is sometimes expressed. The difference is that `systemctl`'s controls extend to other parts of a system as well and are far more extensive than init's or Upstart's controls. `systemctl` carries out these tasks by acting upon systemd's units. Units are often described – not very usefully – as objects that systemd knows how to manage. A more revealing explanation is by example. You can get a general overview from the different types of units, each identified by a suffix: `.service`, `.socket`, `.device`, `.mount`, and so on. For a more detailed example, enter:

```
systemctl list-unit-files
```

and you will see that not only system calls but everything from Bluetooth and

```
bb@nanday:~$ systemctl list-unit-files
UNIT FILE                                  STATE
proc-sys-fs-binfmt_misc.automount          static
-.mount                                    generated
dev-hugepages.mount                        static
dev-mqueue.mount                           static
home.mount                                 generated
media-cdrom0.mount                         generated
proc-fs-nfsd.mount                         static
proc-sys-fs-binfmt_misc.mount              static
run-rpc_pipefs.mount                       static
sys-fs-fuse-connections.mount              static
sys-kernel-config.mount                    static
sys-kernel-debug.mount                     static
tmp.mount                                  generated
usr.mount                                  generated
var.mount                                  generated
acpid.path                                 enabled
cups.path                                  enabled
systemd-ask-password-console.path          static
```

Figure 1: The start of a list of unit files on a typical workstation.

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/*.

```
bb@nanday:~$ systemctl cat atd.service
# /lib/systemd/system/atd.service
[Unit]
Description=Deferred execution scheduler
Documentation=man:atd(8)
After=remote-fs.target nss-user-lookup.target

[Service]
ExecStartPre=-find /var/spool/cron/atjobs -type f -name "=*" -not -newercc /run/system
ExecStart=/usr/sbin/atd -f
IgnoreSIGPIPE=false
KillMode=process
Restart=on-failure

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

Figure 2: The `atd` scheduler is a simple example of a unit file.

CUPS to Plymouth and Apache have systemd units, configured in a corresponding unit file (Figure 1). Notice, too, that units can define both software and hardware attached to the system. You can also write your own unit files for such tasks as mounting a filesystem or running a script at startup.

Unit files have multiple sections that can be in any order, since systemd reads them all before acting on them. However, for the convenience of humans, the tradition is to start each unit file with the `Unit` section, whose properties name the unit, specify the path to documentation, and list any dependencies. Where relevant, the next section is `Install`, which defines elements like aliases and reverse dependencies – system elements that depend on the unit. It is usually followed by a section relevant to the type of unit and any other relevant part of the system. Nothing is arcane about unit files – they are basically configuration files, with one property defined per line, most of which have self-explanatory names. If you have any experience with pre-systemd administration, you can soon understand them. The trouble is, there are dozens, and just to summarize all possible entries would take several thousand words. However, you can explore an individual unit file with the command structure:

```
systemctl cat UNIT
```

Figure 2 shows the unit file for `atd`, the job queue. As you can see, it consists of a general description, followed by the `Service` section, which sets how `atd` runs, and the `Install` section, which defines `atd`'s relationship to other applications. As unit files go, `atd`'s is a simple one, but it gives you some idea of what to expect in other unit files.

As you explore, you will soon find that every major aspect of a system that runs in userland can be controlled by `systemctl`, making it a powerful command indeed. Unit files for the entire system are stored in `/lib/systemd/system`, but are over-ridden by unit files in `/etc/systemd/system`.

```
root@nanday:~# systemctl show bluetooth
Type=dbus
Restart=no
NotifyAccess=main
RestartUSec=100ms
TimeoutStartUSec=1min 30s
TimeoutStopUSec=1min 30s
RuntimeMaxUSec=infinity
WatchdogUSec=0
WatchdogTimestampMonotonic=0
```

Figure 3: A `systemctl` command to display the unit file for Bluetooth.

```
root@nanday:~# systemctl status bluetooth
● bluetooth.service - Bluetooth service
   Loaded: loaded (/lib/systemd/system/bluetooth.service; enabled; vendor preset: enab
   Active: active (running) since Fri 2020-01-31 13:52:50 PST; 5min ago
     Docs: man:bluetoothd(8)
 Main PID: 5027 (bluetoothd)
   Status: "Running"
    Tasks: 1 (limit: 4915)
   Memory: 1.5M
   CGroup: /system.slice/bluetooth.service
           └─5027 /usr/lib/bluetooth/bluetoothd

Jan 31 13:52:50 nanday systemd[1]: Starting Bluetooth service...
Jan 31 13:52:50 nanday bluetoothd[5027]: Bluetooth daemon 5.50
Jan 31 13:52:50 nanday systemd[1]: Started Bluetooth service.
Jan 31 13:52:50 nanday bluetoothd[5027]: Starting SDP server
Jan 31 13:52:50 nanday bluetoothd[5027]: Bluetooth management interface 1.14 initializ
```

Figure 4: The current status for Bluetooth. Note the color coding.

```
root@nanday:~# systemctl enable bluetooth
Synchronizing state of bluetooth.service with SysV service script with /lib/systemd/s
stemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable bluetooth
```

Figure 5: Use `enable` to automatically start a service during bootup.

`systemctl` operates by issuing commands to units, sometimes modified by command options. The basic structure is:

```
systemctl COMMAND OPTION
```

The unit may be specified with its suffix, or without, since the command issued will often be unique to a particular unit (Figure 3). But what commands might you want to issue? The majority fall into one of two categories: controlling services and gathering information about units.

Controlling Services

If you have encountered `systemctl` previously, probably it is as a means of managing services. You can check the status of a unit using the `status` command (Figure 4) and use the `start` and `stop` commands to toggle a unit. However, you may be better off using the `reload` or `restart` commands, or, if you are not sure which to use, `reload-or-restart`, since they will work regardless unit's state. All these commands apply in the current session. However, if you want to make a permanent change to the system and have the service start during bootup, use `enable` or `disable` (Figure 5). Both `enable` or `disable` create a symbolic link from the copy of the unit in `/lib/systemd/system` to the one in `/etc/systemd/system`, where systemd looks for units to start automatically. Remember, though, neither `enable` or `disable` take effect until the next system boot.

If you are troubleshooting, you might start a service using `isolate`. To use this function, a unit must have enabled `AllowIsolate=`. The function runs the unit and its dependencies and stops all others – including, possibly, the desktop environment or command prompt, so be warned. The man page likens `isolate` to changing the runlevel in init. More details are available in the `systemd.unit(5)` man page. `Rescue` offers a similar function, except that it includes the option of sending a message to logged in users.

In addition, `systemctl` has several functions that are aliased to standard Unix commands for root users, like `kill`, `halt`, `poweroff`, `reboot`, `suspend`, and `hibernate`. Mostly, these commands are like the standard Unix ones and include some obvious synonyms. One exception is that `halt`, `poweroff`, and `reboot` have the built in option `--message STRING`, which broadcasts a message to users before the function is activated. Prior to the adoption of systemd, of course, the separate `wall` command was needed for such messages.

Gathering Information

To learn any unit's current state, you can use the `status` command, either to see a tree view of all units or the status of a named unit. To get more specific results, you can use `is-active`, `is-enabled`, and `is-failed`.

For broader views of the system, the command is `list-units`. It returns results that include each unit that is loaded, active or running, as well as a short description. `list-units` can be filtered by a number of options. For example, followed by `-all` (`-a`), it lists all units, regardless of their state. Followed by `--type TYPE` (`-t TYPE`), it lists only the type of unit named, such as service or mount. Similarly, `--state` can be followed by a comma-separated list of `LOAD`, `SUB`, `ACTIVE`, `FAILED`, or `help`. Alternatively, you may prefer to use the `list-units`, `list-sockets`, or `list-timer` commands as a filter.

The `show` command zeroes in on the properties of one or more units or jobs, or, if neither is specified, systemd itself. For even more of a close-up view, you can add `--property` (`-p`) to search for unit files that contain a specific property or `--recursive` (`-r`) to include units in a container in search results. Once you have located a unit, you can learn more about it with `list-dependencies` or, in some cases, `list-jobs`.

With most of these commands and options, you have some control over how the output displays. When you are new to systemd, you may want to use `--full` (`-l`) to display property names and statuses spelled out in full. Later, when you have learned enough of systemd and how it displays information, you may be content with `--values` and not bother to display the property names.

Keeping systemctl on Your Side

This is only an overview of an extremely lengthy subject. You do not need to study `systemctl` for long before you can understand why critics of systemd complain that systemd violates the unofficial Unix philosophy of one simple command to accomplish one simple task. In contrast to that philosophy, `systemctl` can potentially affect everything on a system. However, the obverse of that philosophy is that systemd creates a consistent administrative layer, that, once learned, can be applied elsewhere. In that sense, systemd is simple as well, although its simplicity is of a different nature than a simple shell script or a command with limited philosophy.

However, it hardly needs to be said that with `systemctl`'s great power comes the potential for great harm. Especially as you learn `systemctl`, be careful to double-check your command structure and make sure that you understand exactly what various functions and options do. Otherwise, `systemctl`, which is a benefit to administration, could quickly make life complicated. ■■■

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

Creating bootable images in a GUI

A Kinder dd

Kindd offers a GUI alternative to the ubiquitous dd command-line tool, offering a risk-free option for transferring bootable images to USB sticks. *By Ferdinand Thommes*

When it comes to managing Linux computers, two frequently used commands, `rm` (remove) and `dd` (disk dump), require special attention. These two very powerful tools are potentially destructive. If they are not used carefully, data can be very quickly lost. While `rm` has an integrated emergency brake for the root partition, `dd` overwrites specified partitions without asking, causing some users to interpret `dd`'s name as an acronym for "destroy data."

To prevent specifying the wrong partition (and the resulting data loss), graphical user interface (GUI) tools can be helpful for creating bootable distribution images on external media, such as USB sticks or SD cards. In addition to some distribution-specific solutions, these tools include UNetbootin [1], which is not supported by all distributions; Rufus [2], which is only available for Windows; and Etcher [3], which has gained in popularity recently. Because these applications only offer external devices as options for where to write the data, users are prevented from accidentally overwriting partitions.

The basically excellent Etcher can be used on almost all distributions due to its packaging as an AppImage. However, since it was developed with the Electron framework [4], it needs many dependencies, which inflate the AppImage's size to about 100MB.

Another `dd` alternative, the relatively new Kindd [5], described as "A kindful DD GUI written in Qt Quick," makes do with considerably less disk space. If you remove the unnecessary folder with screenshots after compiling the application, the space requirement can be reduced to a manageable 30MB.

Kindd is ideally suited for desktop environments like Plasma or LXQt and requires at least Qt 5.13. Since Qt 5.13 is not yet available in all distributions, Arch Linux is currently the best distribution choice for running Kindd. In Arch, you will find Kindd in the Arch User Repository (AUR) both as a release version and as a Git excerpt. For all other distributions that already offer Qt 5.13.x or 5.14, you will need to build the software yourself from the source code.

Kindd still lacks important documentation on GitHub, mainly in terms of dependencies needed to compile Kindd. The tool was developed on Void Linux, which is not too widespread, and the developer only lists dependencies for Void. However, I found additional dependencies that needed resolving.

Build-It-Yourself

In addition to the packages listed on Kindd's GitHub page, I also needed to install the following on Void Linux at the time of my test: *git*, *qt5-qmake*, *qt5-declarative-devel*, and *qt5-quickcontrols2-devel*. I built Kindd on a KDE neon User Edition, which is based on the latest long-term support (LTS) version of Ubuntu and has Qt 5.13, result-

Dependencies on Debian Derivatives

You will need to resolve the following dependencies to build and run Kindd on Debian and its derivatives: *git*, *gcc*, *g++*, *make*, *qt5-default*, *libpolkit-qt5-dev*, *qml-module-qt-labs-platform*, *libqtquickcontrols2-5*, *qtquickcontrols2-5-dev*, *qml-module-qtgraphicaleffects*, and *qtdeclarative5-dev*.

Author

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

Photo by Matthew T Rader on Unsplash

ing in additional dependencies (see the "Dependencies on Debian Derivatives" box).

One general annoyance when it comes to building packages on different distributions is the different names used for the required libraries. One distribution uses `-dev` as part of the name, the next `-devel`, and, in some cases, `-qt` is found in different parts of the name.

As a helper for finding the correct package names, Debian uses `apt search` and `apt-file search`, where you first have to install the *apt-file* package and load the package list by typing `apt-file update`. Google can also help here, if you combine a name known from another distribution with the term "Debian." In any case, this is often a time-consuming task, even if you work with distributions, like KDE neon, with a fairly short package lists.

After installing all the dependencies, it's time to fetch the source code from GitHub. You can do this either via Kindd's GitHub page [5] by clicking on *Clone or download*, or you can use `git` in the terminal (Listing 1, line 1). Next, go to Kindd's source code folder and call `qmake` (lines 2 and 3). In my test, I got a prompt back immediately. The following

Listing 1: Downloading Kindd

```
01 $ git clone https://github.com/LinArcX/Kindd/
02 $ cd Kindd/
03 $ qmake
04 $ make
```

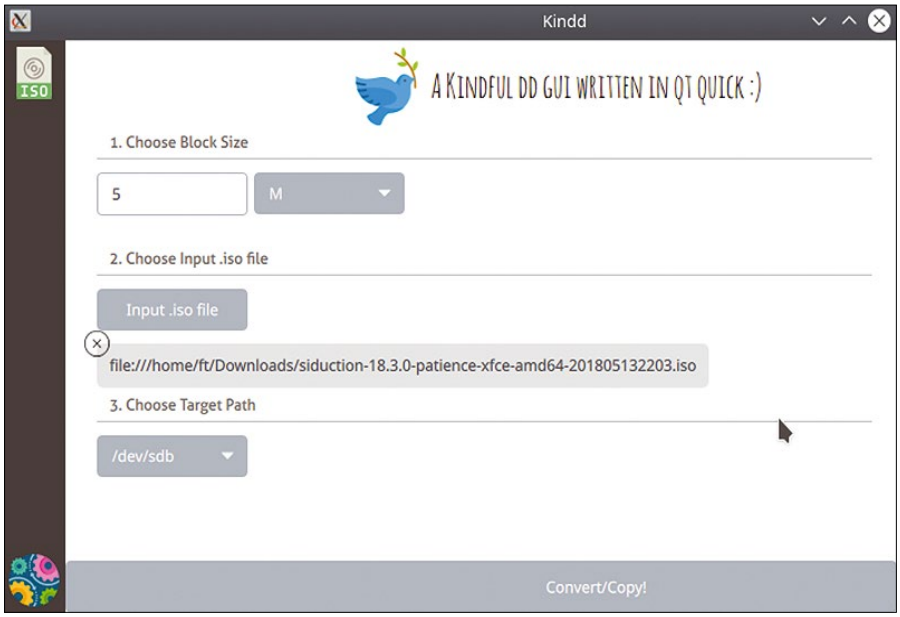


Figure 1: Kindd's straightforward user interface has just three selection fields, including defining block size.

Figure 2: While writing, the user interface shows a progress bar that indicates the amount of data already written and the write speed.

command `make` takes a little longer (line 4). If you see an error message here, you're probably missing a dependency.

If you use a window manager like i3 instead of a full desktop environment, see the *polkit* notes on the GitHub project page.

Simple Interface

After successfully navigating the installation, start the application by calling `./kindd`. The very simple interface (Figure 1) needs virtually no explanation. In contrast to other GUIs that also use `dd`, Kindd lets you adjust the block size; this corresponds to the `bs=` option in the command-line variant.

The *Block Size* option defines the size of the blocks that Kindd reads from the input file and writes to the output device. Instead of the default 5MB, 4MB is a better choice for an ISO image. This is an exact multiple of the 4KB block size of the ext4 filesystem and ensures efficient read and write speeds [6].

The middle selection field, *Choose Input .iso file*, lets you select the image to be copied. *Choose Target Path* lists any connected external devices, such as USB sticks, SD cards, or external hard disks. If there are several external data storage devices on the computer, the `sudo fdisk -l` command can help you decide on the right target.

Transfer

Clicking on the *Convert/Copy* button at the bottom of the application window starts copying the image after you have confirmed another prompt and entered the root password. Kindd displays a progress bar during the process (Figure 2). Upon completion, a small window saying *Congratulations* appears to confirm that the image has been successfully transferred.

If you want to select a different image to be copied, pressing the *ISO* button in the top left corner takes you back to an empty selection mask. The colored ball bottom left takes you to the configuration dialog (Figure 3) where you can change the GUI's look in terms of theme and font type and size. You will want to set the default block size to 4MB and save the settings.

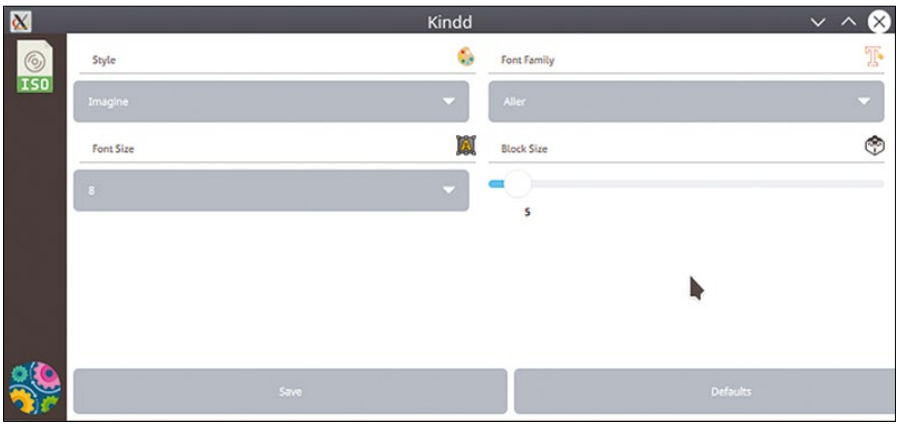


Figure 3: The settings dialog offers some options to customize the appearance of the application, as well as set a default block size.

Conclusions

Kindd is aimed at newcomers who prefer GUIs. However, unless you use the software as a binary package on Arch Linux and its derivatives, it is difficult to build it yourself. Even with Void Linux, I was unable to do this without some adjustments despite reading the manual. Since the software is still fairly new and unpolished, I asked the developer how to proceed with Kindd [7], but his answer was fairly vague.

Once installed, Kindd at least keeps its promises. In the worst case, if you have several USB sticks plugged in, you might overwrite the wrong one, but never a system partition, which can happen when using `dd`. I didn't find the tool's GUI really thrilling, and the themes on offer did little to change my impression. However, you will not be working with Kindd for hours on end, so this is not a deal breaker. ■■■

Info

[1] UNetbootin: *https://unetbootin.github.io/*
[2] Rufus: *https://rufus.ie*
[3] Etcher: *https://www.balena.io/etcher/*
[4] Electron: *https://en.wikipedia.org/wiki/Electron_(software_framework)*
[5] Kindd: *https://github.com/LinArcX/Kindd*
[6] Block size: *https://en.wikipedia.org/wiki/Block_(data_storage)*
[7] Questions to the Kindd developer: *https://github.com/LinArcX/Kindd/issues/9*

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

Analyzing file metadata in the shell

Taking Stock

Armed with the right shell commands, you can quickly identify and evaluate file and directory metadata. *By Harald Zisler*

Imagine you have a directory with hundreds or even thousands of files (without uniform extensions) that you want to organize. Or maybe you want to know the last access date of a file for backup, forensics, or version management purposes.

Instead of tediously clicking your way through the files in a graphical file manager, a shell script with the `test` command can help identify filesystem objects as well as provide additional information about the files.

Determining File Type

The `file` command provides information about a file's contents (Figure 1). Because it tests for patterns in the content, `file` cannot be misled by file extensions (Figure 2).

`file` gets its pattern information from the magic file `/usr/share/misc/magic.mgc`. For special cases, you can create your own magic file and pass it in to `file` by calling it with the option `-m <filename>`. If you are particularly interested in MIME files, use the `--mime-type` option.

```
dd@ubuntu: ~/Data
dd@ubuntu:~/Data$ file -i second
second: text/plain; charset=us-ascii
dd@ubuntu:~/Data$
```

Figure 1: By adding the `-i` option, `file` determines that a file is a plain text file.

```
dd@ubuntu: ~/Data
dd@ubuntu:~/Data$ file second.pdf
second.pdf: ASCII text
dd@ubuntu:~/Data$
```

Figure 2: `file` reliably identifies files whose extensions do not match the actual content.

```
dd@ubuntu: ~/Data
dd@ubuntu:~/Data$ file /dev/sda
/dev/sda: block special (8/0)
dd@ubuntu:~/Data$
```

Figure 3: If you call up the information for a device file, `file` shows the major and minor numbers if possible.

```
dd@ubuntu: ~/Data
dd@ubuntu:~/Data$ sudo file -s /dev/sda1
/dev/sda1: Linux rev 1.0 ext4 filesystem data, UUID=333bc6bc-48f0-4c99-89cf-019f
2b6c7130 (needs journal recovery) (extents) (64bit) (large files) (huge files)
dd@ubuntu:~/Data$
```

Figure 4: Because devices in Linux act like files, `file` can also output information on them.

Listing 1: pdflist.sh

```
01 #! /bin/bash
02
03 # Default quit menu
04 menu="E-N-D"
05
06 cd $HOME/Data
07
08 # Search for PDF files
09 for i in $(ls -1); do
10   file $i | cut -d \: -f2 | grep -q PDF
11   if [ $? -eq 0 ]; then
12     # Show selection
13     menu=$(echo $menu $i)
14   fi
15 done
16
17 # Selection menu with Smenu and PDF display
18 while true; do
19   choice=$(echo $menu | smenu -n 10 -t1 )
20   if [ "$choice" = "E-N-D" ]; then
21     exit
22   fi
23   atril $choice
24   clear
25 done
```

You can even access and evaluate device files with `file`. Additionally, it also outputs the major and minor numbers (Figure 3) – the major number specifies the kernel's device driver, while the minor number specifies the individual device managed by the device driver.

Given the appropriate privileges, you can obtain information about the filesystem (Figure 4). To do so, use the `-s <device file>` option. If you only enter the device for the hard disk without the partition number, the output contains block size details, among other things.

`file` usually outputs the information in the form `<filename>: <data>`. You can take advantage of this when using the tool in shell scripts. If you use a `for` loop in the script, you need unique file names. You can obtain these by typing `ls -1`. This gives you one file name per line. The subshell in the loop header of the `for` loop thus provides reliable arguments until a space occurs in the file name. To avoid this, you have to convert or quote the name.

Listing 1 shows a sample script that specifically searches for PDF files and displays them for selection (Figure 5). The dynamic selection menu is created with the help of Smenu.

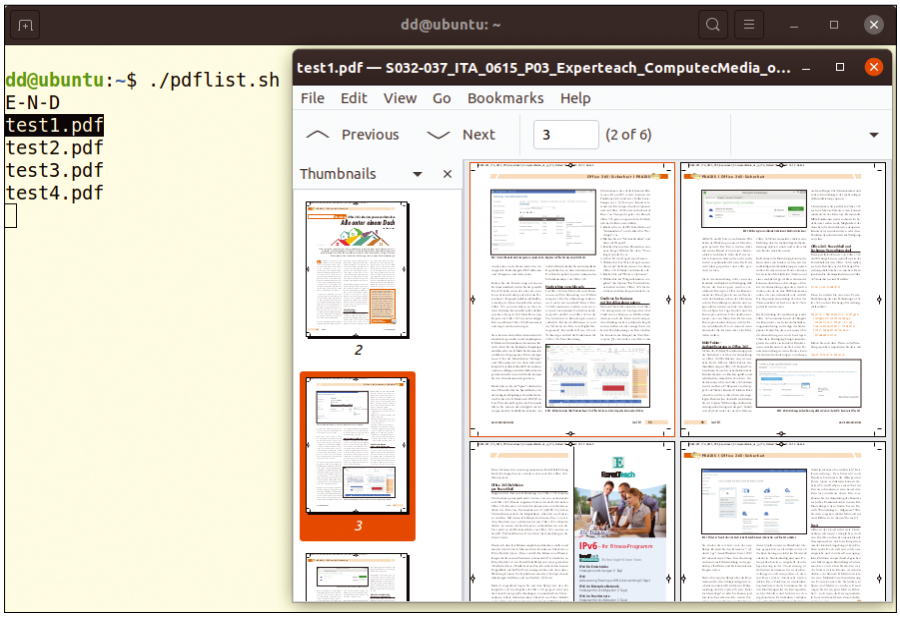


Figure 5: With just a couple of lines of shell code and Smenu, you can create a simple selection menu for a specific file type.

```
dd@ubuntu:~$ stat new
  File: new
  Size: 32              Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 1574028     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/      dd)   Gid: ( 1000/      dd)
Access: 2020-01-28 00:03:12.114682036 -0800
Modify: 2020-01-28 00:03:12.114682036 -0800
Change: 2020-01-28 00:03:12.114682036 -0800
 Birth: -
dd@ubuntu:~$ cat new
Mon 27 Jan 2020 03:35:46 AM PST
dd@ubuntu:~$ stat new
  File: new
  Size: 32              Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 1574028     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/      dd)   Gid: ( 1000/      dd)
Access: 2020-01-28 00:03:16.962691266 -0800
Modify: 2020-01-28 00:03:12.114682036 -0800
Change: 2020-01-28 00:03:12.114682036 -0800
 Birth: -
dd@ubuntu:~$
```

Figure 6: Without specifying any options, `stat`'s output contains a large amount of information.

Status Information

Similar to `ls`, the `stat` command provides file and directory details. Without specifying any other options, `stat` outputs a full set of data for the listed files (Figure 6). Figure 6 also shows the effect of read access – note the `Ac-`

cess line with the date and time information. However, this feature does not work for filesystems mounted with the `noatime` option. `noatime` speeds up data access, because the filesystem does not have to create an entry whenever something is read.

Using `stat -c <format>` you can read specific information about a file (or a filesystem) and evaluate it in a script. Table 1 shows formatting information for `stat`. Figure 7 shows some calls, including querying access rights in numerical form. This information could be useful for an installation script.

Note that `stat`'s output is a bit ambiguous: For example, `Access` means the last read access, but you should note that mount options like `noatime` influence this value. `Modify` refers to the contents of the file, so it may contain the creation date, but always includes the last write access. `Change` shows you information about changing access rights (the owner or similar). The value for `Birth` is currently not determined by `stat` on Linux due to a program error.

Listing 2 shows a small script that reads the access rights of a file and then

Table 1: Stat Format Information

Syntax	Meaning
%a	Access rights in numerical format
%A	Access rights in detailed format
%d	Device number (decimal) for a device file
%t	Major number device file (hexadecimal)
%T	Minor number device file (hexadecimal)
%F	File type
%m	Filesystem where the file resides
%u	Owner UID
%U	Owner username
%g	GID
%G	Group name
%x	Last read access (plain text)
%X	Last read access (Unix seconds)
%y	Last change (plain text)
%Y	Last change (Unix seconds)
%z	Last access (plain text)
%Z	Last access (Unix seconds)

Table 2: Touch Options

Option	Action
-a	Change access time
-m	Change last change time
-t <time>	Use <time> instead of system time
-r <file>	Provides a reference file from which touch takes the timestamp

Listing 2: restrictive.sh

```
01 #! /bin/bash
02
03 # Define $1 as directory,
04 # else cancel
05 if [ -z $1 ]; then
06   exit
07 fi
08
09 # Change to directory
10 cd $1
11
12 for i in $(ls -1); do
13   # Evaluate access permissions
14   stat -c %a $i | grep -q 75
15   if [ $? -eq 0 ]; then
16     # Change if group
17     # or anyone can execute
18     # the file.
19     chmod -v 700 $i
20   fi
21 done
```

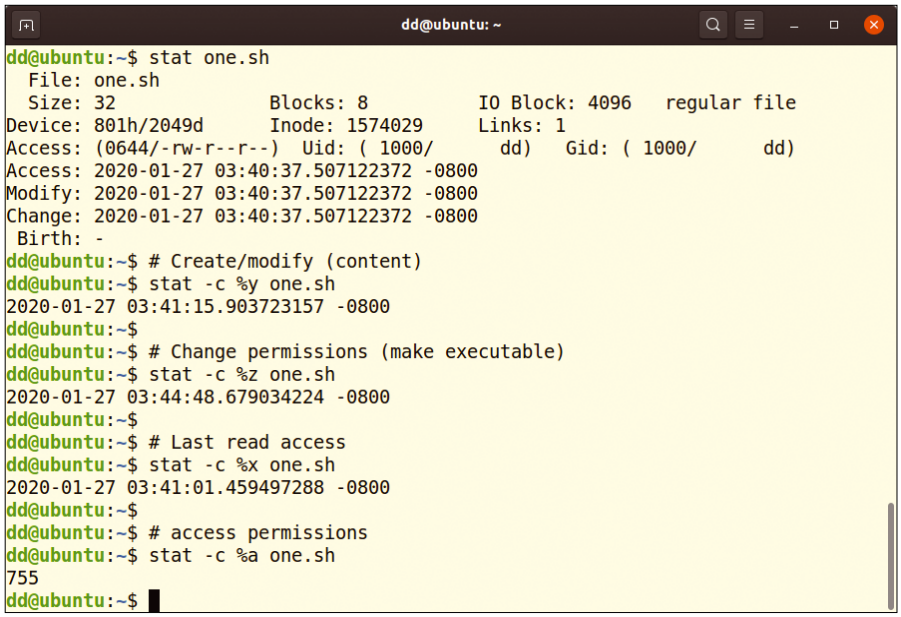


Figure 7: Examples of formatted `stat` queries.

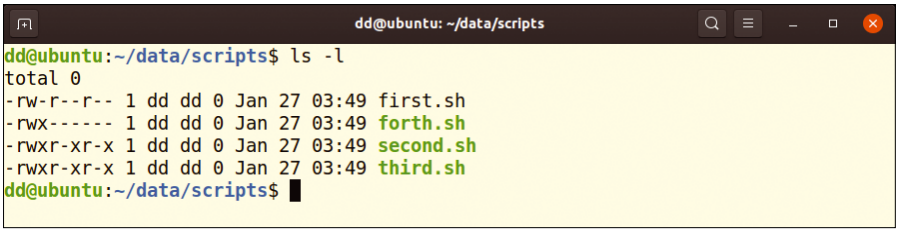


Figure 8: Access rights that are too permissive can give unauthorized persons access to the data under certain circumstances.

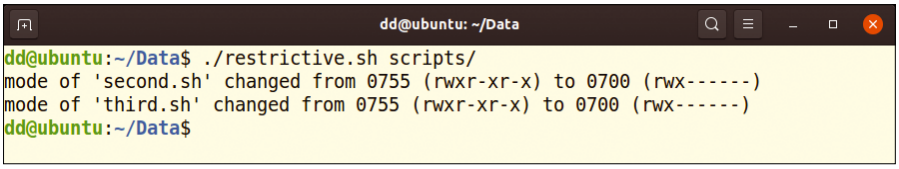


Figure 9: The shell script from Listing 2 showing the changed files.

changes them if they are too permissive. It calls the command to change permissions with the `-v` option so you can see what it is doing in the terminal. Figure 8 shows the database for this; Figure 9 shows the script running.

Changing Timestamps

Applications that work with a file will typically modify the timestamp information. You can do this manually with the `touch` command (see Table 2). If you run `touch` for a nonexistent file name, the system creates a corresponding entry in the filesystem (i.e., it creates a file without any content). If you call `touch <file>` without any options, the program updates all the timestamps in the file to the system time.

The time specification for the `-t` option takes the form of `<MMDDhhmm>`. You can also add the calendar year and seconds to the specification: `<YYYYMMDDhhmm.ss>`. Figure 10 shows how to change the access time using `touch`. Figure 11 shows an example of referencing an existing file for the timestamp. The `stat` command's resulting output shows the special access date set by the command in Figure 10.

Testing Files

Usually, the `test` command is not used in full, but rather as a notation using square brackets and matching options:

```
if [ $? -eq 0 ];
# is the same as
if test $? -eq 0;
```

`test` is used to evaluate the type and timestamps of objects in the directory tree. It returns `0` if the tested condition is true. See Table 3 for `test`'s options.

The `test` command is useful for making a distinction in an `if` construct. The range of applications is extensive. In a script for saving data, for example, it would be possible to check whether a file named `BACKUP.INFO` exists. If so, the script creates a copy of all files that are newer than this file's timestamp. Otherwise, the script creates a full backup.

Listing 3 shows the code for a script that creates a directory named `BACKUPTEST` if it doesn't already exist, quickly performing a common task (Figure 12).

```
dd@ubuntu:~/Data$ stat output2
  File: output2
  Size: 32              Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 1574008     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/      dd)   Gid: ( 1000/      dd)
Access: 2020-01-27 23:23:02.304963764 -0800
Modify: 2020-01-27 23:23:07.433045110 -0800
Change: 2020-01-27 23:23:07.433045110 -0800
 Birth: -
dd@ubuntu:~/Data$ touch -a -t 1001020304.05 output2
dd@ubuntu:~/Data$ stat output2
  File: output2
  Size: 32              Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 1574008     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/      dd)   Gid: ( 1000/      dd)
Access: 2010-01-02 03:04:05.000000000 -0800
Modify: 2020-01-27 23:23:07.433045110 -0800
Change: 2020-01-27 23:23:28.633381393 -0800
 Birth: -
dd@ubuntu:~/Data$
```

Figure 10: Using `touch`, you can modify a file's timestamp.

```
dd@ubuntu:~/Data$ touch -r output2 output1
dd@ubuntu:~/Data$ stat output1
  File: output1
  Size: 32              Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 1586055     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/      dd)   Gid: ( 1000/      dd)
Access: 2020-01-27 23:24:25.710286768 -0800
Modify: 2020-01-27 23:23:07.433045110 -0800
Change: 2020-01-27 23:24:28.438330041 -0800
 Birth: -
dd@ubuntu:~/Data$
```

Figure 11: A timestamp taken from a reference file.

Table 3: Test Options

Test	True, if …
`-e <Object>`	Object exists
True, if <Object> exists and …	
`-b <Object>`	It is a block device file
`-c <Object>`	It is a drawing device file
`-d <Object>`	It is a directory
`-f <Object>`	It is a standard file
`-g <Object>`	The group ID bit is set
`-G <Object>`	The group entries for the process and file match
`-h <Object>`	It is a symbolic link
`-k <Object>`	The sticky bit is set
`-L <Object>`	It is a symbolic link
`-O <Object>`	The query process points to the same owner
`-p <Object>`	It is a FIFO
`-r <Object>`	It is readable
`-s <Object>`	Its size is not `0`
`-S <Object>`	It is a socket
`-u <Object>`	The UID bit is set
`-w <Object>`	It is writable
`-x <Object>`	It is executable
True, if <Object1> exists and …	
`<Object1> -ef <Object2>`	`<Object2>` points to the same object
`<Object1> -nt <Object2>`	Is newer than `<Object2>`
`<Object1> -ot <Object2>`	Is older than `<Object2>`

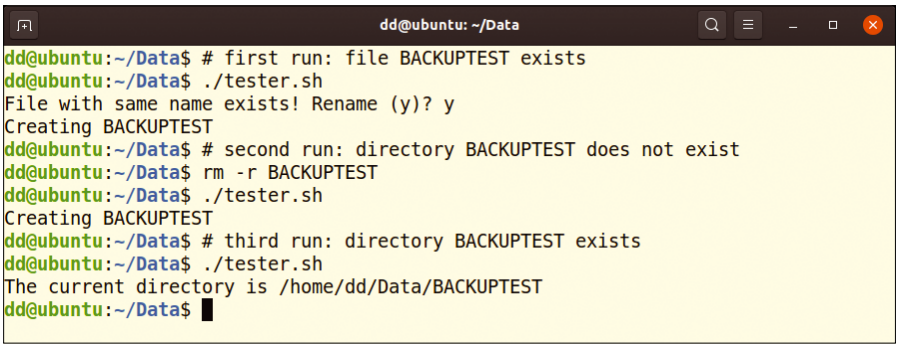


Figure 12: With automation, the `tester.sh` script in Listing 3 handles tasks more quickly and reliably.

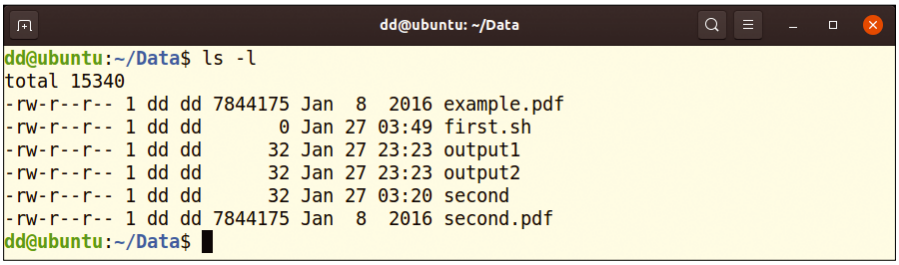


Figure 13: The directory content before running `sortme.sh`.

Timestamps and Rights

The `find` tool not only searches for file and directory names, but it also includes timestamps, access rights, and the file size as a filter if required. See `find`'s man page to learn about the full scope of this command. For the most important `find` options, see Table 4.

If necessary, you can forward `find`'s output to a pipe or process it using `xargs`, which passes the result to other commands, like `tar`. As an example, Listing 4 creates subdirectories for files as a function of their modification date and moves them there. Figure 13 shows the directory's contents before running the script. Figure 14 shows the script in action, and Figure 15 shows the output.

The `for` loop receives the data courtesy of `stat`. Since spaces are used as separators, it replaces them with colons. The output is sorted by date, starting with the newest files.

Listing 3: tester.sh

```
#! /bin/bash

uvznew () {
  # FILE?
  if [ -f BACKUPTEST ]; then
    read -p "File with same name exists! Rename (y)? " we
    if [ "$we" = "y" ]; then
      mv BACKUPTEST BACKUPTEST.file
    else
      echo "Either delete or move the BACKUPTEST file!"
      echo "END OF SCRIPT"
    fi
  fi
  if [ -d BACKUPTEST ]; then
    cd BACKUPTEST
    echo -n "The current directory is $PWD"
    echo " "
  else
    echo "Creating BACKUPTEST"
    mkdir BACKUPTEST
  fi

  return 0
}

uvznew
```

Listing 4: sortme.sh

```
01 #!/bin/bash
02 # $1 = directory to process
03 if [ -z $1 ]; then
04   echo "No input"
05   exit
06 fi
07
08 cd $1
09
10 for i in $(stat -c %y:%n * | sort -r | tr \  \:); do
11   # Populate variables subdir (subdirectory)
12   # and fn (filename)
13   subdir=$(echo $i | cut -d \: -f1)
14   fn=$(echo $i | cut -d \: -f6)
15   # Do not process if directory
16   # restart loop
17   if [ -d $fn ]; then
18     echo "$fn: Skipping directory"
19     continue
20   fi
21   # Create subdir if needed, and
22   # move file to it
23   if [ -d $subdir ]; then
24     mv -v $fn $subdir
25   else
26     mkdir $subdir
27     mv -v $fn $subdir
28   fi
29 done
```

```
dd@ubuntu:~$ ./sortme.sh Data
renamed 'output2' -> '2020-01-27/output2'
renamed 'output1' -> '2020-01-27/output1'
renamed 'first.sh' -> '2020-01-27/first.sh'
renamed 'second' -> '2020-01-27/second'
renamed 'second.pdf' -> '2016-01-08/second.pdf'
renamed 'example.pdf' -> '2016-01-08/example.pdf'
dd@ubuntu:~$
```

Figure 14: `sortme.sh` evaluates the metadata and sorts files into the appropriate directory structure.

Table 4: Find Options

Action	Option	Note
`-type <type>`	Search by type	–
`-size +/-<size>`	Search by size	- = maximum size; + = minimum size; nothing = same size
`-perm <file permissions>`	Search for file permissions	–
`-newer <files>`	Search for files newer than `<file>`	–
`-mtime -/+<N>`	Search for files not modified for `<N>` days	+ = at least; - = within
`-atime -/+<N>`	Search for files not accessed for `<N>` days	+ = at least, - = within
`-execdir <command> "{}" +`	Execute `<command>` for found file	Safer method

In the `for` loop, the `subdir` variable contains the subdirectory to be created and `fn` takes the file name. The script evaluates whether or not `fn` is a directory and, in this instance, aborts processing. The loop then begins with a new pass. This prevents the script from processing a directory.

If, on the other hand, `fn` is a file, the routine then checks again by means of a test whether the subdirectory already exists. If this is not the case, it creates the directory and moves the file to it. If the directory already exists, it simply does the latter. Figure 15 shows a visual overview of `tree`.

```
dd@ubuntu:~$ tree Data
Data
├── 2016-01-08
│   ├── example.pdf
│   └── second.pdf
└── 2020-01-27
    ├── first.sh
    ├── output1
    ├── output2
    └── second

2 directories, 6 files
dd@ubuntu:~$
```

Figure 15: Files end up in the corresponding subdirectories in the new directory structure.

Conclusions

With the appropriate shell commands, you can identify, evaluate, and change various file and directory metadata. As a result, many operations can be simplified using scripts, which avoid errors and save valuable time. ■■■

Author

Harald Zisler has focused on FreeBSD and Linux since the early 1990s. He is the author of various articles and books on technology and IT topics. The fifth edition of his book *Computer-Netzwerke* (Computer Networks) was recently published by the Rheinwerk Verlag publishing company. He also works as an instructor, teaching Linux and database topics in small groups.

Dstat, NetHogs, and nload

Everything Must Go

Every sys admin has a few favorite tools that they always carry with them, if only because they do not want to be without these often overlooked treasures. The gems dangling from Charly's key ring include Dstat, NetHogs, and nload. *By Charly Kühnast*

On my laptop there is a list of topics I want to write about in the near or distant future. There are tools, tricks, and hacks that I often and gladly use, but which I cannot convert into enough text for a whole column. This explains why three small utilities share this column today.

Dstat

Let's start with Dstat [1]. Last issue, I described how to get details about the installed hardware (especially RAM) from Linux. I was promptly asked how to detect "hogs" as quickly and easily as possible – processes that grab resources (such as CPU, RAM, etc.) without so much as a by your leave. I will keep this short, because I have already written about this topic [2]. My secret weapon looks like this:

```
# dstat -cdn -D sda -N enp2s0 -C total ➥
  --top-cpu --top-io --top-mem -f 5
```

Every second, this displays which processes are generating the highest CPU, memory, and I/O load (Figure 1). This command has saved me from working late dozens of times.

NetHogs

Unfortunately, Dstat does not show you which process is generating the most network traffic at the moment. NetHogs [3] fills this gap. On machines with multiple interfaces, it only needs the name of the desired network interface as a parameter. If not specified, it grabs the first interface that is not called `localhost`.

A list of all processes that send or receive network packets appears (Figure 2). I use the *R* and *S* keys to tell Dstat to sort this by incoming and outgoing traffic respectively. NetHogs has a nice graphical add-on, HogWatch [4], that visualizes the data. Unfortunately, HogWatch is no longer actively maintained.

```
root@glas:~# dstat -cdn -D sda -N enp2s0 -C total --top-cpu --top-io --top-mem -f 5
--total-cpu-usage-- --dsk/sda-- -net/enp2s0 -most-expensive- ----most-expensive---- --most-expensive-
usr sys idl wai stl| read  writ| recv  send|  cpu process    |      i/o process      |  memory process
  3   1  96   0   0| 504B   70k|   0     0 |pihole-FTL   0.2|munin-node   27k 326B|systemd-jour 277M
  4   2  95   0   0|   0     0 |2353B 2745B|kworker/1:0  0.1|fping       4157B   0 |systemd-jour 277M
  4   2  95   0   0|   0  4915B|5566B   22k|smoke        0.2|smoke      4187B 151B|systemd-jour 277M
  3   2  95   0   0|   0    58k|2564B 3281B|sshd         0.1|snmpd      1209B   0 |systemd-jour 277M
  3   1  95   0   0|   0   819B|2244B 5018B|pihole-FTL   0.1|snmpd      4021B   0 |systemd-jour 277M
  3   1  96   0   0|   0     0 |2964B 4034B|snmpd        0.2|snmpd      1209B   0 |systemd-jour 277M
  3   1  95   0   0|   0     0 |2693B 6266B|apache2      0.1|snmpd      1209B   0 |systemd-jour 277M
  3   2  95   0   0|   0     0 |4048B 3525B|pihole-FTL   0.2|fping       4157B   0 |systemd-jour 277M
  8   3  88   1   0|   0   102k|  10k  106k|pihole-FTL   3.9|pihole-FTL    53k  30k|systemd-jour 277M
 16   6  78   0   0|   0    76k|  12k  233k|munin        1.8|munin       267k 3187B|systemd-jour 277M
```

Figure 1: Dstat frequently saves Charly from working late.

nload

If you want to see a meaningful net load curve, but don't want to use a graphical tool, nload [5] will do the job. The following command draws the current net load level with cursors on the console:

```
# nload -t 1000 -o 10000
```

The `-t 1000` parameter specifies the update interval in milliseconds (default: 500ms). `-o 10000` tells the tool to cap the graph at 10Mbps, because nload scales it to the interface's maximum speed (Figure 3). If you have a gigabit or faster interface, but a low load, you simply will not see it. ■■■

```
NetHogs version 0.8.1

    PID USER     PROGRAM                DEV        SENT      RECEIVED
   3290 www-data /usr/sbin/apache2      eth0     149.591     12.133 KB/sec
   1812 root     /usr/bin/ssh           eth0       1.474      1.447 KB/sec
   1667 root     /usr/bin/ssh           eth0       0.202      0.159 KB/sec
  13943 root     sshd: root@pts/0       eth0       0.922      0.052 KB/sec
   1130 root     /usr/bin/ssh           eth0       0.036      0.018 KB/sec
    877 root     /usr/bin/ssh           eth0       0.036      0.018 KB/sec
   2579 root     dovecot/imap-login     eth0       0.000      0.000 KB/sec
   1412 root     /usr/bin/ssh           eth0       0.000      0.000 KB/sec
   1317 root     /usr/bin/ssh           eth0       0.000      0.000 KB/sec
   1381 root     /usr/bin/ssh           eth0       0.000      0.000 KB/sec
      ? root     unknown TCP                       0.000      0.000 KB/sec

  TOTAL                                          152.260     13.827 KB/sec
```

Figure 2: NetHogs adds the traffic information that Dstat lacks.

Info

[1] Dstat: *https://github.com/dagwieers/dstat*
[2] "The sys admin's daily grind: Dstat" by Charly Kühnast, *Linux Magazine*, issue 150, May 2013, p. 65
[3] NetHogs: *https://github.com/raboof/nethogs*
[4] HogWatch: *https://github.com/akshayKMR/hogwatch*
[5] nload: *https://github.com/rolandriegel/nload*

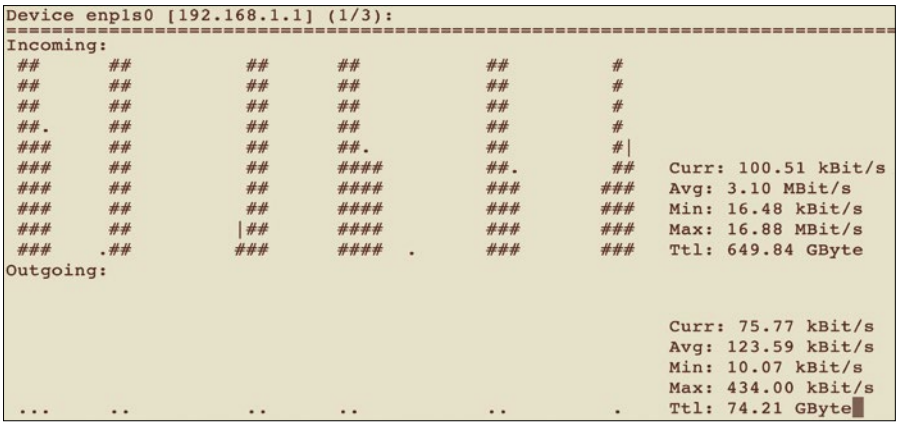


Figure 3: nload provides a meaningful network load graph.

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

Introducing sorting algorithms in Go

Sorted

Whether alphabetical or numerical, bubble sort or quicksort, there is no escape from sorting in computer science. In this month's column, Mike Schilli sorts out the pros and cons of various sorting algorithms in Go.

By Mike Schilli

Long before the first computers existed, sorting data occupied mankind. The German-American Herman Hollerith invented an electromechanical machine as early as 1890 that sorted punched cards into different exit shafts to speed up the evaluation of the US census of that era. And even today, computers continue to sort data – whether this be for a list of YouTube video suggestions, the top 100 charts in the music industry by sales figures, or the slowest queries against a MySQL database for performance analysis purposes.

Machines sort at amazing speeds with programs changing the order of arbitrarily formed data structures. Most of the time, they're relying on a simple key as a sorting criterion, often an integer or a character string. This explains why classical reference works on sorting algorithms [1] [2] often only show you how to sort a series of numbers. This is because porting these algorithms to deal with more complex data structures is trivial and can be done by simply defining a mapping of the data structure to a key.

Listing 1: bubble.go

```
01 package main
02
03 import (
04   "fmt"
05 )
06
07 func main() {
08   items := []string{"green", "red",
09     "blue", "black", "white"}
10
11   for i, _ := range items {
12     for j := i + 1; j < len(items);
                 j++ {
13       if items[i] > items[j] {
14         items[i], items[j] =
15           items[j], items[i]
16       }
17     }
18   }
19
20   fmt.Printf("%+v\n", items)
21 }
```

Bubbles Rise to the Surface

As the simplest sorting algorithm, programming beginners often learn how to bubble sort with two `for` loops. In the first loop, the `i` index in Listing 1 [3] works its way from left to right through the elements of an array. For each element examined, another `for` loop starts with the `j` index; it scans all the following elements and checks if the element under review from the first loop is smaller than all the following ones. If the procedure finds a smaller element further to the right, it simply swaps it with the one it is examining. In this way, all elements looked at by the first loop are guaranteed to not have

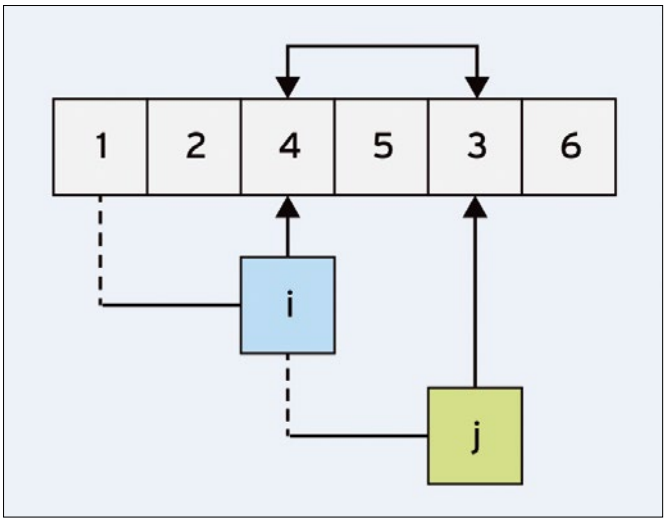


Figure 1: The two loop variables `i` and `j` in the bubble sort work their way through the array and swap unsorted elements.

```
$ ./counting
[3 4 5 3 3 4 4 4 4 5 5 3 3 3 3]
count=[0 0 0 7 5 3]
[3 3 3 3 3 3 3 4 4 4 4 4 5 5 5]
$
```

Figure 2: Three steps during a run of counting sort: Input data, computed index, and final result.

any smaller elements to their right. Finally, when the first loop arrives at the far right end, the array is sorted in ascending order (Figure 1).

In the first `for` loop starting in line 11, Listing 1 shows you the `range` operator; for each element of the `items` array slice, it returns the index, starting at *0*, and the element itself. The loop header ignores the latter by assigning it to the pseudo-variable `_`, since it is only interested in the index of the currently examined element.

The second `for` loop in line 12 starts with the element directly to the right of the currently examined one; the index, `j`, starts at `i+1`. If an element is found at index position `j` that is larger than the one referenced by `i`, the program uses an `a,b = b,a` construct to quickly swap the two, which is a boon in Go compared to other programming languages mandating the use of temporary variables. Listing 1 is easily compiled by typing

```
go build bubble.go
```

and the program will display an alphabetically sorted list of colors (Listing 2).

The bubble sort algorithm is easy to understand, but requires a relatively large number ($O(n^2)$) of steps to complete the sorting task. Smarter methods like merge sort or quicksort offer better performance in general. For small amounts of data, such as arrays with up to five elements, bubble sort works well enough. It sorts the entries in the original array without external storage requirements, it is stable (the order of identical elements remains unchanged), and it handles presorted input data quickly.

Counting Sort

In job interviews, candidates are sometimes asked to name a fast sort algorithm. Before instinctively replying "quicksort," they might do well to consider whether the specific framework conditions of the problem at hand might allow for clever alternatives. For example, if the elements to be sorted consist of, say, just a few different keys, the counting sort algorithm [4] is most likely much quicker (Figure 2).

In Listing 3, the `things` array to be sorted contains just a few repeating values, `3`, `4`, and `5`. Instead of firing up a traditional sorting algorithm, the implementation uses a newly allocated `count` array, which uses counters at index positions `3`, `4`, and `5` to note how often the corresponding values occur in the input array. In this way, a value of *7* is written down at index position `3`, because there are seven `3`s in `things`, and index position `4` in `count` contains a *5*, for the five `4`s in `things`.

Extending Static Arrays

Since the sorting routine does not initially know how many different values will occur in the array to be sorted, it first creates an empty ("count") array. If the `for` loop then wants to create an entry in `count` at a position that exceeds the size of the array, line 16 uses the built-in `append()` function to extend `count` to the required size.

In contrast to typical scripting languages, Go does not work with arrays that grow dynamically; instead, they are allocated with fixed sizes. If a program creates a new array with capacity for three elements using `make([]int, 3)` and later accesses an element outside the range *0* to *2*, Go will throw a run-time error and terminate the program. This also applies to array slices, by the way. They only define a sliding window onto an existing static array. An array can only be extended by the built-in `append()` function, which creates a new array with all additionally required elements appended at the end.

In other words, line 16 creates a new, longer array from the old `count` array, which has been found to be too short, and then reassigns it to the original `count`. The new array differs from the old one by the difference between the desired index (in the `thing` variable) and the last index of the existing array. To do this, the code uses `make()` to create a small array that fills up the difference and feeds its elements one by one (ex-

Listing 2: Color List

```
$ ./bubble
[black blue green red white]
```

Listing 3: counting.go

```
01 package main
02
03 import (
04   "fmt"
05 )
06
07 func main() {
08   things := []int{5, 4, 3, 3, 3, 4, 4,
09     4, 4, 5, 5, 3, 3, 3, 3}
10   fmt.Printf("%+v\n", things)
11
12   count := []int{}
13
14   for _, thing := range things {
15     if len(count) <= thing {
16       count = append(count, make([]int,
17                thing - len(count) + 1)...)
18     }
19     count[thing]++
20   }
21   fmt.Printf("count=%+v\n", count)
22
23   idx := 0
24   for val, run := range count {
25     for x := 0; x < run; x++ {
26       things[idx] = val
27       idx++
28     }
29   }
30
31   fmt.Printf("%+v\n", things)
32 }
```

Figure 3: When a card player picks up a new hand, they insert the new cards in sorted order by making space where the new cards will fit.

panded by the `...` operator) to the `append()` function.

After the first `for` loop, the `count` array contains the counters for the collected keys, as printed out as a checkpoint in line 21. The second `for` loop starting in line 24 iterates over the counters in `count` and fills the `things` array with seven `3`s first, then with five `4`s, and so on. Although the procedure uses an external `count` array for counting, the extra memory required is not relevant for a small number of different keys. Sorting is done in situ (i.e., the original array is modified). The number of required steps is linear ($O(n)$), so the method works orders of magnitude faster than even the most sophisticated generic sorting function for large amounts of data – but only because of the very special properties of the input data, of course.

Insertion Sort

When players of card games like Rummy or bridge pick up a new hand, they often insert new cards into their hand one by one by making room at the position where the new card fits to create a sorted hand. To do this, they move the lower value cards to the left and higher value cards to the right and insert the new card into the space created (Figure 3). This is also how the insertion sort algorithm [1] works, a procedure that sorts in-place with only a few lines of code, but requires some mental dexterity.

The algorithm divides the array to be sorted into two sections: on the left, the elements already sorted so far; on the right, the unsorted elements. To do this, it determines the element to be examined, starting with the second element of the array. In Figure 4 (top), this is initially the number *4*. To the left of it, the previously sorted list contains only the *3*; to the right of it, what is – thus far – the unsorted part then follows, starting with a *1*. Now a loop starts going left to check where to insert the current element into the sorted section. Since the current element (the *4*) is larger than the only sorted element to its left, there is nothing else to do – the algorithm proceeds to the next element.

In the next step in Figure 4 (bottom), the number *1* is the current element, and the loop hops over to the left to check where to insert it in the sorted part. It first sees the *4*, determines that the current element (*1*) must be to the left of it, and moves the *4* one place to the right as shown in step (a), in place of the current element that was previously saved in a variable. Further to the left, it then sees the *3*, which also needs to be on the right of the current element, and then moves the *3* to the position previously occupied by the *4* as shown in step (b). Now there is nothing standing in the way of the *1* previously stashed away in a variable; it can move to the first position in the array as shown in step (c). The sequence is now *1-3-4-*…, and the algorithm proceeds by inspecting the *2* as the current element. Lather, rinse, repeat.

Listing 4 implements the whole thing in a relatively compact way. The `for` loop in line 11 determines the current element, starting at `j=1` (i.e., the second element of the array starting at index position *0*). Listing 4's output with the various sorting

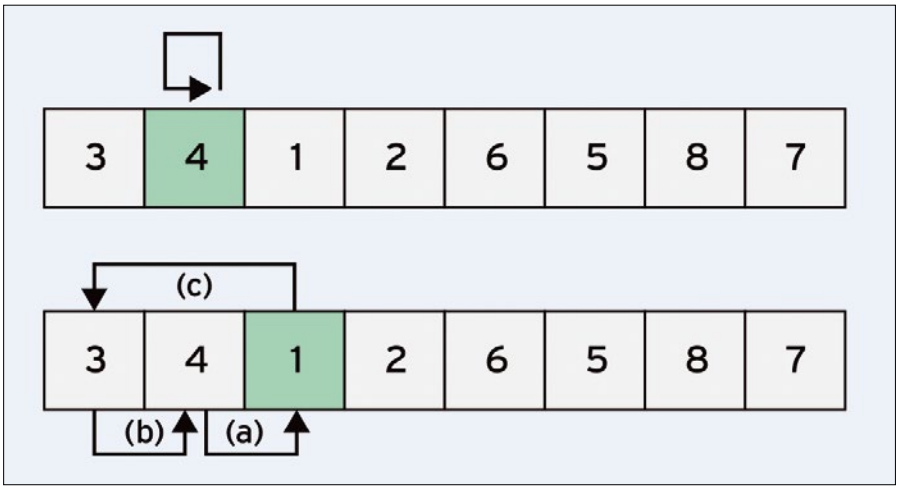


Figure 4: Moving elements in an insertion sort.

```
$ ./insert
[3 4 1 2 6 5 8 7]
key=4
[3 4 1 2 6 5 8 7]
key=1
[1 3 4 2 6 5 8 7]
key=2
[1 2 3 4 6 5 8 7]
key=6
[1 2 3 4 6 5 8 7]
key=5
[1 2 3 4 5 6 8 7]
key=8
[1 2 3 4 5 6 8 7]
key=7
[1 2 3 4 5 6 7 8]
$
```

Figure 5: The insertion sort from Listing 4 sorts the array by finding out-of-place values and inserting them into the right spot.

Listing 4: insert.go

```
01 package main
02
03 import (
04   "fmt"
05 )
06
07 func main() {
08   things := []int{3, 4, 1, 2, 6, 5, 8, 7}
09   fmt.Printf("%+v\n", things)
10
11   for j := 1; j < len(things); j++ {
12     key := things[j]
13     fmt.Printf("key=%d\n", key)
14     i := j - 1
15     for i >= 0 && things[i] > key {
16       things[i+1] = things[i]
17       i--
18     }
19     things[i+1] = key
20     fmt.Printf("%+v\n", things)
21   }
22 }
```

steps and their current `key` elements in each case is shown in Figure 5.

The test loop, which jumps to the left to make room in the sorted part, starts at `i=j-1` in line 14. It continues until it encounters a sorted element that is less than or equal to the current element, or until it risks dropping off on the left end of the array at `i<0`. In each round of the loop, line 16 copies an element in the sorted section one place to the right, and `i` is decremented by one. At the end of the loop, line 19 copies the current element cached in `key` to the vacant position, and the outer `for` loop moves on to the next round.

However, the number of steps required by insertion sort increases proportionally to the square of the number of elements. The algorithm therefore does not work faster than a bubble sort. For randomly structured input data, there are definitely smarter methods that reduce the number of steps from $O(n^2)$ to $O(n*lg(n))$.

Merge Sort

Better methods often rely on the "divide and conquer" principle. This means that they do not solve the whole problem in one go, but divide it into two more easily solvable partial problems, which they then tackle recursively.

What I like about merge sort is its almost trivial implementation based on plain vanilla recursion. The trick is to cut the array in half, sort the two halves individually, and then merge the two sorted stacks into a new one resulting in a larger sorted stack. The individual "sorting" procedure is not really sorting at all: The sorting function again chops the array in half and calls itself against the two halves until it deals with an array with only one element. The func-

Listing 5: merge.go

```
01 package main
02
03 import (
04   "fmt"
05 )
06
07 func main() {
08   things := []int{17, 34, 42, 1, 7, 9, 8}
09
10   fmt.Printf("%+v\n", things)
11   mergesort(things, 0, len(things)-1)
12   fmt.Printf("%+v\n", things)
13 }
14
15 func mergesort(a []int, p, r int) {
16   if p < r {
17     q := (p + r) / 2
18     mergesort(a, p, q)
19     mergesort(a, q+1, r)
20     merge(a, p, q, r)
21   }
22 }
23
24 func merge(a []int, p, q, r int) {
25   left := make([]int, q-p+1)
26   right := make([]int, r-q)
27
28   copy(left, a[p:])
29   copy(right, a[q+1:])
30
31   fmt.Printf("left=%d\n", left)
32   fmt.Printf("right=%d\n", right)
33
34   lidx := 0
35   ridx := 0
36   oidx := p
37
38   for lidx < len(left) &&
39       ridx < len(right) {
40     if left[lidx] < right[ridx] {
41       a[oidx] = left[lidx]
42       lidx++
43     } else {
44       a[oidx] = right[ridx]
45       ridx++
46     }
47     oidx++
48   }
49   for lidx < len(left) {
50     a[oidx] = left[lidx]
51     lidx++
52     oidx++
53   }
54   for ridx < len(right) {
55     a[oidx] = right[ridx]
56     ridx++
57     oidx++
58   }
59 }
```

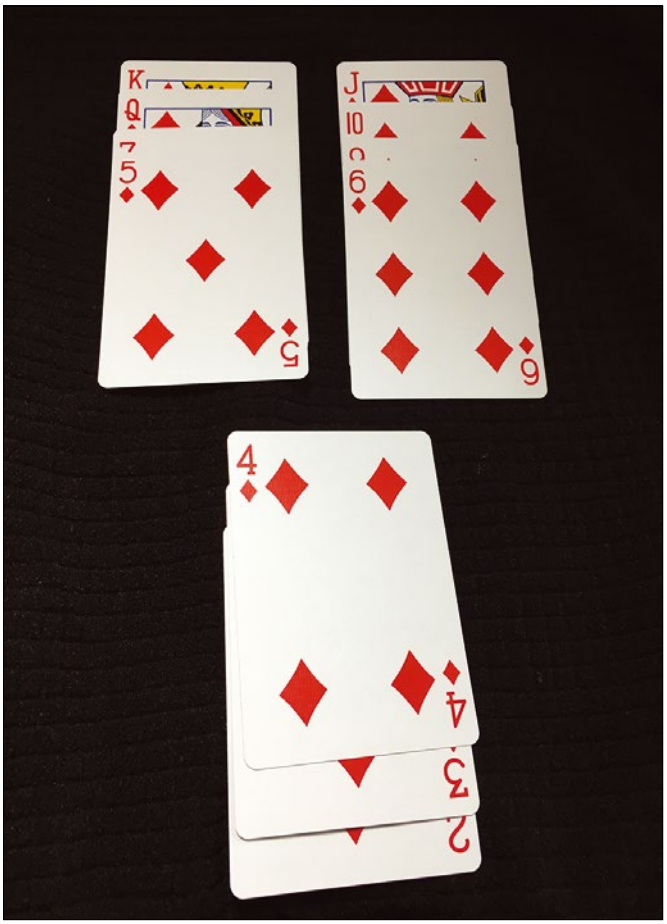


Figure 6: The `merge()` function combines two sorted stacks of cards (on top) to create a new stack that is also sorted (bottom). It always picks the top stack with the lower card for the next step.

```
$ ./merge
[17 34 42 1 7 9 8]
left=[17]
right=[34]
left=[42]
right=[1]
left=[17 34]
right=[1 42]
left=[7]
right=[9]
left=[7 9]
right=[8]
left=[1 17 34 42]
right=[7 8 9]
[1 7 8 9 17 34 42]
$
```

Figure 7: Merge sort puts a field with seven integers in the correct order.

tion declares this as sorted and returns it without change.

The real grit of the procedure is in the `merge()` function, which merges two sorted arrays. To do this, it walks through the elements of the two sorted stacks, compares their two top elements, removes the smaller one, and puts it on the result stack. Figure 6 illustrates the procedure with a card game that combines the two upper sorted stacks to create the stack at the bottom, which ends up to be sorted as well.

In Listing 5, the `merge()` function starting in line 24 expects an integer array `a`, as well as three index positions: `p`, `q`, and `r`. The two partial arrays that `merge()` has to join are available at the positions `p` through `q` and `q+1` through `r`. The function then goes ahead and stores the results in place in the original array `a`.

Listing 6: stringsort.go

```
01 package main
02
03 import (
04   "fmt"
05   "sort"
06 )
07
08 func main() {
09   things := []string{"green",
      "red", "blue", "black", "white"}
10
11   sort.Strings(things)
12
13   fmt.Printf("%+v\n", things)
14 }
```

This presupposes that `merge()` first creates a copy of the left and right array in the variables `left` and `right`. The function uses `make()` to allocate memory for arrays of the desired length. Then, starting in line 28, it calls the built-in `copy()` function and passes it the arrays to be copied as `a[p:]` ("Element at index `p` and following") and `a[q+1:]` ("Element at index `q+1` and following"). Note that we don't want `copy()` to actually copy all following elements, but make use of the fact that `copy()` only copies as many characters as will fit into the target array, specified as the first parameter. But since we've allocated `left` and `right` correctly with their required lengths previously, the number of characters actually copied is also correct.

Then the `for` loop runs through the two arrays starting in line 38, removes the smaller current element from the corresponding stack, and terminates if either one is depleted. The remaining elements of the other array are then copied into the resulting array one by one by the two `for` loops starting in line 49 or 54.

The main `mergesort()` function in line 15, which expects the unsorted array, as well as the index positions of start and end, only needs to chop the array passed to it into two parts, call itself recursively against the two subfields, and call `merge()` to merge the two results. The recursion is repeated until `mergesort()` receives an array with only one element (then `p == r` and the `if` condition is false) whereupon `mergesort()` simply returns, because this one-element array is sorted by definition. Figure 7 shows how the algorithm gradually sorts an array of seven integers and prints out the result.

Built-In

Normally, however, aside from coding interviews, software engineers never really code up their own sorting algorithms in Go. The language offers built-in sorting functions for slices of strings, integers, or floats. After importing the `sort` package (Listing 6), `sort.Ints()`, `sort.Float64s`, and `sort.Strings()` are available, each of which arranges a slice of integers or floating-point numbers or strings in ascending order by sorting the array passed to it in place.

As you can see from the source code [5] on the Go website, the language

Listing 7: sortstruct.go

```
01 package main
02
03 import (
04   "fmt"
05   "sort"
06 )
07
08 type Record struct {
09   id int
10 }
11
12 type Records []Record
13
14 func (r Records)
15  Len() int {
16   return len(r)
17 }
18
19 func (r Records) Swap(i, j int) {
20   r[i], r[j] = r[j], r[i]
21 }
22
23 func (r Records) Less(i, j int) bool {
24   return r[i].id < r[j].id
25 }
26
27 func main() {
28   data := Records{{5}, {1}, {2}, {7}, {3}}
29   sort.Sort(data)
30   fmt.Printf("%+v\n", data)
31 }
```

Listing 8: Results

```
$ ./sortstruct
[{id:1} {id:2} {id:3} {id:5} {id:7}]
```

uses the quicksort algorithm internally. This is a divide-and-conquer mechanism that runs at breakneck speed virtually everywhere. Usually it only takes $O(n*log(n))$ steps to reach its goal. On rare occasions, however, it freaks out completely to the tune of $O(n^2)$ complexity – with a million entries; this could mean the difference between seconds and years.

If the algorithm has to terminate with 100 percent assurance, other methods sometimes turn out to be more advantageous. However, in order to fend off such outliers, standard libraries of programming languages often include procedures that slightly modify the algorithm and avert worst case scenarios.

Sorting Arbitrary Data

Spoiled script kiddies will be rubbing their eyes in amazement: Go's strict type system requires quite a bit of extra overhead before it will sort an array of arbitrary data. It turns out to be impossible to define a sort routine that sorts arrays with elements of a generic data type. Instead, the programmer has to explicitly specify how Go should compare and swap the elements.

Listing 7 defines a fictitious data structure `Record`, which only contains an integer value as an attribute. To sort an array with elements of `Record` types, the programmer has to teach Go:

- How to determine the length of the array with the data types
- How to compare two elements at the positions `i` and `j`
- How to swap two different entries against each other (i.e., everything a generic sort function does with its data internally while it's performing the sort)

For this purpose, Listing 7 contains the `Len()`, `Swap()`, and `Less()` functions, to each of which you need to assign a receiver as an array of `Record` types (i.e., an array of the `Records` type defined above). You can easily determine the length of the array with the built-in Go `len()` function, so `Len()` is taken care of. Two elements are swapped courtesy of Go's practical swapping syntax (`a,b = b,a`), so `Swap` turns out to be easy-peasy. Two elements are compared using the less-than operator (`<`), as shown in line 24, which is the meat of `Less()`. To have Go sort the array in place with quicksort, line 29 calls the `sort.Sort()` function and passes the array with the record elements to it. The correct result, as determined by the algorithm, is shown in Listing 8.

It turns out that a language like Go with strict type checking requires a little more programming overhead than a scripting language that sorts strings, integers, and floats without complaining, or to which you could easily add a homemade sorting function for types you define yourself. This disadvantage is not only made up for by Go's high processing speed, but also by the fact that the compiler complains about accidental type errors – rather than just the program at run time. But, as usual, there's no free lunch. ■■■

Info

[1] Knuth, Donald E. *The Art of Computer Programming, Volume 3: Searching and Sorting*, Addison-Wesley Professional, 1998

[2] Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, MIT Press, 2009

[3] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/233/*

[4] Counting sort: *https://en.wikipedia.org/wiki/Counting_sort*

[5] Sort functions in the Go standard library: *https://golang.org/src/sort/sort.go*

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.

Scraping the web for data

Data Harvesting

Web scraping lets you automatically download and extract data from websites to build your own database. With a simple scraping script, you can harvest information from the web. *By Marco Fioretti*

If you are looking to collect data from the Internet for a personal database, your first stop is often a Google search. However, a search for mortgage rates can (in theory) return dozens of pages full of relevant images and data, as well as a lot of irrelevant content. You could visit every web page pulled up by your search and cut and paste the relevant data into your database. Or you could use a web scraper to automatically download and extract raw data from the web pages and reformat it into a table, graph, or spreadsheet on your computer.

Not just big data professionals, but also small business owners, teachers, students, shoppers, or just curious people can use web scraping to do all manner of tasks from researching a PhD thesis to creating a database of local doctors to comparing prices for online shopping. Unless you need to do really complicated stuff with super-optimized performance, web scraping is relatively easy. In this article, I'll show you how web scraping works with some practical examples that use the open source tool Beautiful Soup.

Caveats

Web scraping does have its limits. First, you have to start off with a well-crafted search engine query; web scraping can't replace the initial search. To protect their business and observe legal constraints, search engines deploy anti-scraping features; overcoming them is not worth the time of the occasional web scraper. Instead, Web scraping shines (and is irreplaceable) *after* you have completed your web search.

Web page complexity, which has increased over the past 10 to 15 years, also affects web scraping. Due to this complexity, determining the relevant parts of a web page's source code and how to process it has become more time consuming despite the great progress made by web scraping tools.

Dynamic content poses another problem for web scraping. Today, most pages continuously refresh, changing layout from one moment to the next, and are customized for each visitor. This makes the scraping code more complicated, without providing, sometimes, the certainty that the scraper will extract exactly what you would see in your browser. This is particularly problematic for online shopping. Not only does the price change frequently, but the price also depends on many independent factors, from shipping costs to your buyer profile, shopping history, or preferred payment method – all of which are just outside of web scraping's radar. Consequently, the best deal found by an ordinary scraping script may be quite different from what you would be offered when clicking on the corresponding link. (Unless you spent so much time and ef-

fort on tweaking the web scraper that you wouldn't have time left to enjoy your purchases!)

Additionally, many web pages only display properly after some JavaScript code has been downloaded and run or after some interaction with the user. Others, like Tumblr blogs, run an "infinite scroll" function. In all of these cases, a scraper may start parsing the code before it is ready for viewing, thus failing to deliver what you would see in your browser.

Changing HTML tags is yet another issue. Scraping works by recognizing certain HTML tags, with certain labels, in the web page you want to scrape. The label names may change after a software upgrade or adoption of a different graphic theme, resulting in your scraper script failing until you update its code accordingly.

Scraping can consume a lot of bandwidth, which may create real problems for the website you are scraping. To remedy this, make your scrapers as slow as possible and scrape only when there is no alternative (i.e., when webmasters don't provide direct access to the data you want), and everybody will be happy.

Finally, a website's business needs, as well as copyright and other legal issues, stand in the way of web scraping. Many webmasters try their best to block automated scraping of their content. This article does not attempt to address the copyright issues related to screen scraping, which can vary with the site requirements and jurisdiction.

Web Scraping Steps

In spite of these caveats, web scraping remains an immensely useful activity (and if you ask me, a really fun one) in many real world cases. In practice, every web scraping project goes through the same five steps: discovery, page analysis, automatic download, data extraction, and data archival.

In the discovery phase, you search for the pages you want to scrape via a search engine or by simply looking at a website's home page.

During page analysis, you study the HTML code of your selected pages to determine the location of the desired elements and how a scraping script might recognize them. Most of the time, HTML tags' `id` and `class` attributes are the easiest to detect and use for web scraping, but that is not always the case. Only visual inspection can confirm this and give you the names of those attributes. You can get a web page's HTML code by saving the complete web page on your computer or right-clicking on *View Page Source* (or *View Selection source* for a selected paragraph) in your browser.

The final three steps (automatic download, data extraction, and data archival) involve writing a script that will actually download the page(s), find the right data inside them, and write them to an external file for later processing. The most common format, which I use in two of my examples, is CSV (Comma-Separated-Values), which is a plain text file with one record per line and fields separated by commas or other predefined characters (I prefer pipes). JSON (JavaScript Object Notation) is another popular choice that is more efficient for certain applications.

Remember to keep both your code and your output data as simple as possible. For most web scraping activities, you will only grab the data once, but may then spend days or months analyzing it.

Listing 1: Scraping Wikipedia Paragraphs

```
#! /usr/bin/python3

from requests import get
from requests.exceptions import RequestException
from contextlib import closing
from bs4 import BeautifulSoup

def simple_get(url):
  try:
    with closing(get(url, stream=True, verify=False)) as resp:
      if is_good_response(resp):
        return resp.content
      else:
        return None

  except RequestException as e:
    log_error('Error during requests to {0} : {1}'.format(url, str(e)))
    return None


def is_good_response(resp):
  content_type = resp.headers['Content-Type'].lower()
  return (resp.status_code == 200
    and content_type is not None
    and content_type.find('html') > -1)

def log_error(e):
  print(e)

wikipedia_page = simple_get('https://en.wikipedia.org/wiki/Alexander_the_Great')
content = BeautifulSoup(wikipedia_page, 'html.parser')

n = 1
for paragraph in content.find_all("p"):
  if paragraph.has_attr('class') and paragraph['class'][0] == 'mw-empty-elt':
    continue
  print(paragraph)
  n += 1
  if n == 3 : break
```

Consequently, it doesn't make sense to optimize the scraping code. For instance, a program that will run once while you sleep (even if it takes a whole night to finish) isn't worth spending two hours of your time to optimize. In terms of your output data, it's difficult to know in advance all the possible ways you may want to process it. Therefore, just make sure that the extracted data is correct and laid out in a structured way when you save it. You can always reformat the data later, if necessary.

Beautiful Soup

Currently, the most popular open source web scraping tool is Beautiful Soup [1], a Python library for extracting data out of HTML and XML files. It has a large user community and very good documentation. If Beautiful Soup 4 is not available as a binary package for your Linux distribution, you can install it with the Python package manager, `pip`. On Ubuntu and other Apt-based distributions, you can install `pip` and Beautiful Soup with these two commands:

```
sudo apt-get install python3-pip
pip install requests BeautifulSoup4
```

With Beautiful Soup, you can create scraping scripts for simple text and image scraping or for more complex projects.

Text Scraper

Listing 1 shows part of a text scraper that I used for a micro-encyclopedia project [2] in HTML and ePUB format, whose entries consist of the first two full paragraphs of the corresponding Wikipedia pages. Listing 1 retrieves just those two paragraphs, with all their HTML formatting, from the individual Wikipedia pages.

I grabbed code for lines 1 to 28 from the web, as it seems to be the standard, well-tested header of most Beautiful Soup-based scraping scripts. I only added the `verify=False` (line 10) to skip verification of SSL certificates, because I was forced to scrape websites that did not have them.

The code loads Beautiful Soup 4 and the other libraries (*requests* and *contextlib*) needed to download web pages and handle errors. The main function `simple_get` in line 8 tries to download the web page from the URL passed as its argument, using two other functions, `is_good_response` and `log_error`. If the download's status code is `200`, the page retrieved is not empty and is in HTML format (lines 23 to 25), `is_good_response` is true, and `simple_get` returns the whole page (line 12). Otherwise, it calls `log_error` to report the problem (lines 16 to 18).

My original code for the text scraper reads a full list of Wikipedia pages from a file. In this simplified version (Listing 1), I just pass one URL to `simple_get` in line 30, saving its result in the `wikipedia_page` variable. Line 31 calls `BeautifulSoup`, telling it to parse the whole page with its HTML parser and save a copy in a structured format made to order for scraping, inside the `content` variable. It is this data structure that makes scraping feasible even for occasional programmers. The `find_all` construct in line 34, in fact, creates a list of all the page elements that are paragraphs (which are marked with the `<p>` tag), and the `for` loop in the same line scans all of them.

I want the first two paragraphs that actually contain text, but Wikipedia pages sometimes have empty paragraphs in unpredictable places. By looking at the page's source code, I discovered that those empty paragraphs have a CSS class `mw-empty-elt`. If the current paragraph has a class attribute with that value, line 35 detects it, and line 36 moves the loop to the next paragraph. Otherwise, the full HTML code of the paragraph is printed, and the counter is incremented. As soon as the loop has found two non-

Figure 1: Map images in their original location on the web (note the image title).

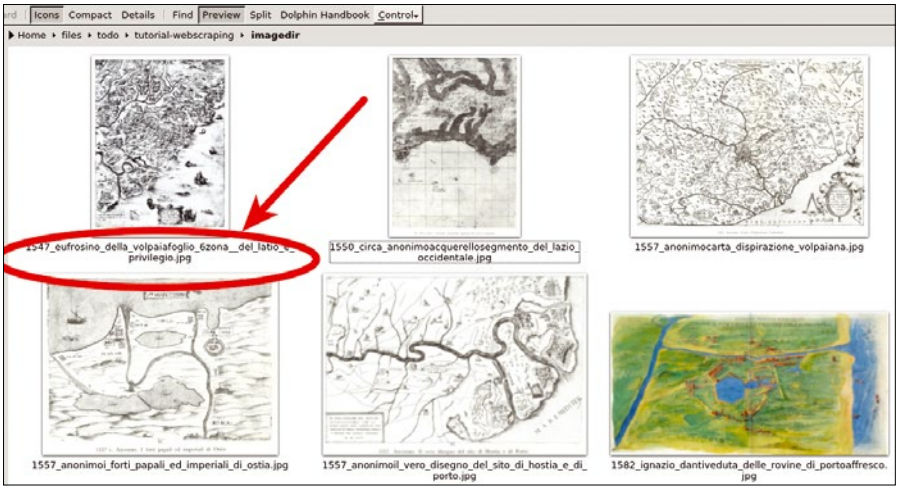


Figure 2: The maps, including image titles, from Figure 1 downloaded to my computer using a simple web scraping script.

Listing 2: Automatic Download of Ancient Maps

```
 1 import re
 2 import urllib.request
 3 import os
 4
 5 cartographypage = simple_get('https://www.codadellacometa.it/studi/cartografie/storica/storica.html')
 6 content = BeautifulSoup(cartographypage, 'html.parser')
 7
 8 for image_link in content.find_all('a', attrs={'href': re.compile("jpg$")}):
 9   image_url = "https://www.codadellacometa.it/studi/cartografie/storica/" + image_link.get('href')
10   local_image_name = os.path.basename(image_link.get('href'))
11   print("Saving", image_link.get('title'), "to", local_image_name)
12   urllib.request.urlretrieve(image_url, "imagedir/" + local_image_name)
```

empty paragraphs, the `n` counter becomes equal to `3`, and the whole script stops in line 39. Of course, I could have detected empty paragraphs by checking the length of their content, but I wanted to show you how to check the value of CSS classes, because these parameters, together with `id` elements that you can process in the same way, are usually the best markers for navigating HTML code.

Image Scraper

You can also use a web scraper for images. I like to collect copies of ancient maps from areas around Rome. Listing 2 shows how I scraped map images from a local university's website (Figure 1) and saved them on my computer (Figure 2). The complete image scraper script includes lines 1-28 from Listing 1, but Listing 2 omits those lines for brevity. The first three lines load three other Python libraries: *re* for the regular expression of line 8, *urllib* to save web pages locally, and *os* to handle paths on the local filesystem. Lines 5 and 6 are identical to lines 30 and 31 in Listing 1: They grab the web page containing the links to all the images and save its `BeautifulSoup` representation in the `content` variable. Even here, the `find_all` function creates a list of only the elements I want to scrape, for the loop in lines 8 to 12.

In this case, those elements are the ones that contain hyperlinks to JPEG images. Such elements are recognizable because they have an `href` attribute and a value ending with the `.jpg` extension. The values of those `href` attributes, however, are not complete URLs that you can directly download. To get the complete URL, line 9 prepends the website address to each `image_link`'s `href` value and saves the result into `image_url`. Line 10, instead, uses the `os.path.basename` function to set the name that the image's local copy will have on my computer. Now line 11 can log to standard output the file name and

Figure 3: The starting point of a city-scraping project: a partial, alphabetical index of Italian cities.

Figure 4: The partial indexes, linked from the page shown in Figure 3, contain all the URLs to the pages of each city.

Figure 5: The Wikipedia page of a small italian town, with the fields to be scraped highlighted.

Listing 3: HTML Source Code from Figure 5

```
<td><a href="/wiki/File:Fara_in_Sabina_dai_Ruderi_di_S_Martino_stretta.JPG" class="image" title="Fara in Sabina --
   Veduta"><img alt="Fara in Sabina -- Veduta" src="//upload.wikimedia.org/wikipedia/commons/thumb/8/83/Fara_in_Sabina_dai_
   Ruderi_di_S_Martino_stretta.JPG
...
<th style="width:80px;"><a href="/wiki/Altitudine" title="Altitudine">Altitudine</a></th><td class="" style="">482 m s.l.m
```

Listing 4: Wikipedia City Scraper

```
 1 import time
 2
 3 fields = ['Superficie', 'Abitanti', 'Altitudine',
             'Provincia', 'Sindaco', 'Cl. sismica', 'Patrono']
 4
 5 city_lists = simple_get('https://it.wikipedia.org/wiki/
                           Comuni_d%27Italia')
 6 content = BeautifulSoup(city_lists, 'html.parser')
 7
 8 li = content.select("ul > li > a")
 9 for link in li:
10   if "Comuni d'Italia (" in str(link.text) :
11     city_list_url =  "https://it.wikipedia.org" + link.
                        get('href')
12     print("LOG: ", link.text, "from ", city_list_url)
13     city_list_letter = simple_get(city_list_url)
14     content = BeautifulSoup(city_list_letter, 'html.
                               parser')
15
16     for row in content.find_all('table')[0].tbody.find_
         all('tr')[1:]:
17       city_url  = "https://it.wikipedia.org" + row.find_
                     all('td')[0].a['href']
18       city_name = row.find_all('td')[0].text
19       print("LOG:\t\t\t", city_name)
20
21       city_source = simple_get(city_url)
22       content = BeautifulSoup(city_source, 'html.parser')
23       print(city_name, "|", end="")
24
25       for f in fields:
26         label = content.find(lambda ttag: ttag.name=='th'
                                and ttag.text==f)
27         if label: print(label.find_next('td').text.
                           strip() + "|", end="")
28       label = content.find(lambda ttag: ttag.name=='th'
                              and ttag.text=='Sito
                              istituzionale')
29       if label: print(label.a['href']+ "|", end="")
30       for img in content.find_all('img'):
31         if "Provincia" in str(img.get('alt')) : continue
32         if "Regione"   in str(img.get('alt')) : continue
33         elif "Veduta"  in str(img.get('alt')):
             print("View= "  + img.get('src') + "|", end="")
34         elif "Stemma"   in str(img.get('alt')):
             print("Crest= " + img.get('src') + "|", end="")
35         elif "Bandiera" in str(img.get('alt')):
                               print("Flag= "  + img.
                               get('src') + "|", end="")
36       print("\n")
37       time.sleep(1.5)
```

the description of the image that is inside its `title` attribute (Figure 1). Finally, line 12 downloads the current image from the website and saves it into the folder `imagedir`, with the name built in line 10.

A Wikipedia City Scraper

For a more complex project, I scraped images and text from two different websites and merged them. I needed basic statistics and contact data for all Italian cities, (almost 8,000). I fetched most of the data from the Italian Wikipedia, and the rest from another source.

First, I needed to get all the URLs for the pages I wanted to scrape. The Italian Wikipedia has one separate page for each city, but not one single page with a complete list of almost 8,000 entries. It does provide several types of partial indexes; I chose the alphabetical one shown in Figure 3 as it was the easiest to use for my needs.

Clicking on any of the links in Figure 3 would open a page as shown in Figure 4, with links to all the city pages for the corresponding letter(s). Consequently, I had to scrape the master index first to get the URLs of the approximately 20 sub-indexes, then scrape each sub-index for the links to the individual city pages, and finally download and parse the individual city pages.

Additionally, I needed to know what to scrape, exactly, from each of those individual pages (Figure 5). I wanted to scrape the images, when available, of the city flag (*Bandiera*), crest (*Stemma*), and view (*Panorama*). I also needed some textual data, namely the Province (*Provincia*) of each city, its mayor (*Sindaco*), elevation (*Altitudine*), area (*Superficie*), and residents (*Abitanti*), plus three other categories (not shown in Figure 5): earthquake risk class (*Cl. sismica*), patron saint (*Patrono*), and official website (*Sito istituzionale*).

Looking at Figure 5, you can see that all the data is available as a cell in the first table of each Wikipedia web page. That table's source code is one messy chunk of HTML. Listing 3 shows just part of the HTML code for *Panorama* (image) and *Altitudine* (text).

Listing 4 shows the script that helped me grab this mass of data (lines 1-28 as shown in Listing 1 are omitted for brevity). Here, I also use the *time* library (line 1) to make the script pause 1.5 seconds (line 37) after downloading each page to reduce bandwidth consumption.

Line 3 is an array containing all the fields in the table that have the same formatting and can therefore be fetched with one common procedure. Using the

Listing 5: Output of Wikipedia City Scraper

```
Fara in Sabina
54,96 km²
13 904[1] (31-12-2017)
482 m s.l.m.
Rieti
Davide Basilicata dal 16-5-2011
zona 2B (sismicità media)
sant'Antonino
http://www.farainsabina.gov.it/
Crest= //upload.wikimedia.org.../Fara_in_Sabina-Stemma.png
Flag= //upload.wikimedia.org.../Fara_in_Sabina-Gonfalone.png
View= //upload.wikimedia.org.../Fara_in_Sabina_dai_Ruderi_di_S_Martino_stretta.JPG
```

methods already described, lines 5 and 6 dump Figure 3's master index inside the auxiliary `content` variable. Line 8 saves inside the array `li` (short for "list index") all the elements of the page that are anchors (`a`) to hyperlinks, placed inside a list element (`li`) of any unordered list (`ul`). Looking at that index page, I knew that the only links I wanted were those whose text had the form `"Comuni d'Italia <SOME letter>` ("Comuni d'Italia" means "Municipalities of Italy"). Consequently, the first thing that the loop starting in line 9 does is check if the current element contains that string (line 10). If it does, line 11 builds the full URL of that sub-index. After the log message in line 12, lines 13 and 14 fetch and parse that URL (which is a page formatted like Figure 4) in the usual way.

Figure 4 shows that the links to the individual pages for each city are contained in the first column of the first table of every sub-index page. The rows of that table are loaded and scanned one by one by the loop starting in line 16. The absolute URL of each page is built in line 17 in the same way as line 11, by prepending the absolute URL to the link found in the first cell of the row. Thanks to `BeautifulSoup`, the notation `row.find_all('td')[0].a['href']` is all that is needed to achieve this. The same technique, applied in line 18 to the text attribute of the first cell of each row, returns the name of the city. In the first cell of Figure 4, `city_name` would have the value *Abano Terme* and `city_url` the value highlighted in the bottom left corner of Figure 4.

Once it knows the city page's URL, the script logs what it found (line 19) and downloads that page into another `content` variable (lines 21 and 22). I can reuse the same variable name (`content`) in each loop without errors, because each instance is local (i.e., only visible) in that loop.

Line 23 starts creating a new line of what will become the final CSV file with all the data: It writes the city name and the pipe character (`|`) without a newline (this is the meaning of the `end=""` statement). The loop in lines 25 to 27 retrieves from the HTML table all the variables listed in the `fields` array of line 3, and highlighted in Figure 5, by using a very powerful Python construct, lambda functions.

Lambda functions are, in their simplest form, short functions without an explicit name, written inline with the code that needs them. In practice, line 26 uses a lambda function to search and load into a `label` variable all the cell headers of the page whose text is equal to one of the fields of line 3. Only when that happens is the `label` variable defined, and this makes the script execute line 27: It loads the table's next data cell of (`find_next`), which contains the field value, strips it of all the initial and trailing whitespaces, and prints it to standard output, followed by another pipe character.

Lines 28 and 29 use exactly the same technique to retrieve the URL of the city's institutional website (`href` field in line 29). I needed separate code for this, because this time I extract a link, not the string associated with it.

The last loop of the script applies yet another version of the same basic trick to grab the URLs of the view, crest, and flag images. Lines 31 and 32 skip the provinces' flags and crests, because I only want those of cities. Lines 33 to 35 print the link found inside the HTML image tags, but only for images whose alternate description (`alt`) includes the words `"Veduta"`, `"Stemma"`, or `"Bandiera"`. Line 36 closes the record for the current city, by adding a newline (`\n`). When I ran this script and stripped out all the `LOG` lines, I got a spreadsheet with rows

Figure 6: The website that provides the official email address.

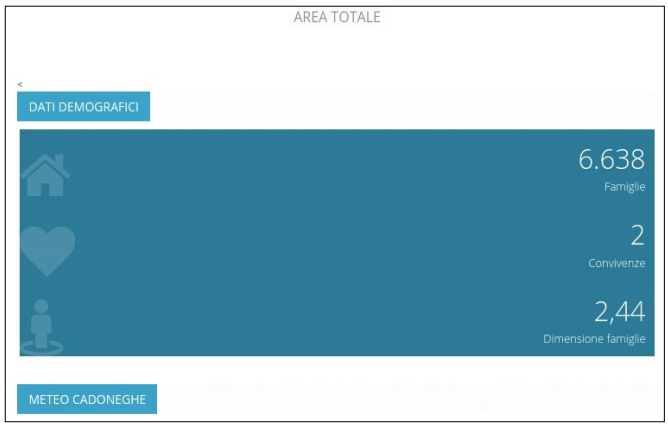


Figure 7: The website where I found the demographic data.

as shown in Listing 5 (for readability, each field is on a separate line and the URLs are shortened).

More City Data

As quick and dirty as it is, the Wikipedia city scraper did its job pretty well. Unfortunately, I needed additional data for each city that was not available on the Wikipedia pages: an email contact address for each city and demographic information. Eventually, I found another portal [3] that published that data, plus other data I did not need. Figure 6 shows the section of that portal that contains the email contact; Figure 7 shows the page with the demographic data.

Listing 6 (again omitting lines 1-28 shown in Listing 1) shows the scraper I wrote to extract the additional data. Since it has the same basic structure as Listing 4, I'll only outline its main parts, leaving the details as an exercise for the reader. This website provides one single list of all the cities as one sequence of 164 numbered pages, whose URLs have the format `https://www.comuniecitta.it/comuni-italiani?pg=N`. The loop starting in line 3 loads those pages one at a time and then loads the URLs of the individual cities' pages from the first table it finds (line 9). When the script loads a city page, the `demobox` section in lines 17 to 24 extracts the demographic data, and lines 26 to 29 detect and print all the email addresses on the page. The result, again, is a CSV text file with fields separated by pipe characters with rows (Listing 7). At this point, the outputs of the two city-scraping scripts can be easily merged, with the Bash `join` command or another script, into one single database with all the data in one coherent format. Since this is a task not limited to web scraping, I leave it as an exercise for the reader.

Listing 6: Email/Demographic Information Scraper

```
 1 import time
 2
 3 for i in range(1,164):
 4   current_page = 'https://www.comuniecitta.it/comuni-italiani?pg=' + str(i)
 5   print("LOG: fetching page number", i, "from", current_page)
 6   city_lists = simple_get(current_page)
 7   content = BeautifulSoup(city_lists, 'html.parser')
 8
 9   for row in content.find_all('table')[0].tbody.find_all('tr')[1:]:
10     city_url  = "https://www.comuniecitta.it" + row.find_all('td')[0].a['href']
11     city_name = row.find_all('td')[0].text
12     print("\n\nLOG:",city_name, "\n\n" + city_name + "|" + "|", end="")
13
14     city_source = simple_get(city_url)
15     content = BeautifulSoup(city_source, 'html.parser')
16
17     demobox = content.find_all("div", {"class": "row statBoxesAlt"})
18
19     for tags in demobox:
20       demoboxDivs = tags.find_all("div", {"class": "col-sm-4"})
21       for singleDiv in demoboxDivs:
22         demofield = singleDiv.text
23         demofield = demofield.replace('\n', ' ').replace('\r', '')
24         print(demofield + "|", end="")
25
26     email = content.find_all("td")
27     for address in email:
28       if "@" in str(address.text):
29         print(str(address.text), end="")
30     print("\n")
31     time.sleep(1.5)
```

Conclusions

The official Beautiful Soup documentation contains additional information, but with these examples, you now know enough to use it productively. If you decide to do large scale web scraping, I recommend checking out how to use shared proxies. You should set your `User Agent` headers, possibly changing their value at random interval, as follows:

```
myheader = {
  'User-Agent': 'Mozilla/5.0 (
  Windows NT 6.3; Win64; x64) ...
```

Add "`headers=myheader`" to the parameters of your `get(url)` calls (for details, see the documentation). This will make your requests look as if they were coming from several normal web browsers, in different locations, instead of one voracious script. Happy scraping! ■■■

Info

[1] Beautiful Soup: *www.crummy.com/software/BeautifulSoup/bs4/doc/*

[2] Micro-encyclopedia project: *http://stop.zona-m.net/2017/12/5000-concepts-for-europe-a-book-proposal/*

[3] Italian Municipalities and Cities: *www.comuniecitta.it*

Author

Marco Fioretti (*http://stop.zona-m.net*) is a freelance author, trainer, and researcher based in Rome, Italy, who has been working with Free/Open Source software since 1995, and on open digital standards since 2005. Marco also is a board member of the Free Knowledge Institute (*http://freeknowledge.eu*).

Listing 7: Sample Output from comuniecitta.it

```
Abbadia Cerreto|   118 Famiglie  |   0 Convivenze  |   2,45 Dimensione famiglie  | info@comune.abbadiacerreto.lo.it
```

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

MakerSpace

A digital picture frame with weather forecast

Reuse and Recycle

A digital picture frame displays photographs and a current weather forecast with just a few hundred lines of Bash and a Raspberry Pi. *By Chris Dock*

Not all that long ago I was thinking it would be pretty neat to buy the small-form-factor SheevaPlug computer [1]. However, it was pretty pricey for a computer for which I had no specific use, so I never bought it. All of that went out the window when the pretty powerful and super affordable Raspberry Pi came out in 2012. Although I didn't have any immediate project in mind, I lined up with everyone else to purchase one.

Once I received my Raspberry Pi, I set out to create projects (e.g., a proxy server, LED cube [2], and networked music device [3]) just to get started. I also experimented with the I2C and SPI protocols and played with devices previously in the domain of the Arduino.

Before I knew it, I had a small collection of these neat little devices. Over time, however, they fell out of use. These classic Raspberry Pis still work, but they aren't as fast and as capable as the most recent Raspberry Pi. While visiting my parents last summer, I saw their old unused TFT monitor in the corner, and it occurred to me how it could be reused. A 15-inch monitor is nothing special when connected to a PC, but it is pretty fabulous when turned into an electronic picture frame.

Most electronic picture frames, usually 8 to 10 inches measured diagonally, simply cycle through photos either in internal memory or on an SD card. The task is well within the capabilities of the Raspberry Pi, but I had another twist that would make this a premium solution. I don't visit my folks as often as I would like, but I would like pictures of their grandchildren always displayed prominently.

What I envisioned was as picture frame connected to common data storage that could be maintained remotely and to which photos could be downloaded automatically. Yet considering the non-technical nature of my parents, the picture frame would need to be self-maintaining.

To justify the counter space for a 15-inch monitor, I would really need to convince my folks of the extra utility this solution would bring. My mom keeps a pretty close eye on the weather forecasts, so I thought the picture frame could also retrieve current weather information.

The picture frame functionality can be summarized in a few words, but implementing the solution would require a fair number of "moving parts":

- LCD monitor
- Raspberry Pi
- SD card
- USB stick
- HDMI-to-VGA adapter cable
- WiFi dongle/Ethernet cable
- Weather data
- Shared Internet data source

Author

Christopher Dock is a senior consultant at T-Systems On Site Services GmbH. When not working on integration projects, Christopher likes to experiment with Raspberry Pi solutions and other electronics projects. You can read more of his work at *http://blog.paranoidprofessor.com*.

Google Drive

My initial Internet data drive was not Dropbox, but Google Drive. Google has a much larger storage space of 15GB associated with each existing email account and has the community-supported `gdrive` client on GitHub [5] that performs much like the Dropbox command-line tool. One slight difference between the two clients is that the Google Drive client operates on file IDs, not file names.

When I started this project, I used the `gdrive` client, but while re-verifying all of the steps, I discovered that the client no longer worked for new email addresses. It turns out the original developer embedded their application ID into the client and either the ID was revoked, or it hit some magical limit at Google and was disabled.

If you prefer to use Google Drive, the source code does exist. It is simply a matter of registering with Google and recompiling the client. If you are comfortable with the Go programming language or would like to learn a bit about it, this might be an interesting project.

The hardware in my case was an old Raspberry Pi B with a WiFi dongle. The setup of the operating system was just a matter of imaging an SD card [4] with an image from the Raspberry Pi site. For this project, it was important that the Raspberry Pi boot automatically and log in to the desktop to display pictures. Although not a requirement, SSH is useful during development and perhaps later for support.

Helper Tools

All functionality would be controlled from a few Bash scripts, with a few utilities to perform some of the tasks:

```
apt-get install curl jq feh
```

`curl` retrieves data with URLs. More specifically, it will be used to retrieve weather data. Information found on the Internet comes in a number of formats, but one of the easiest data formats to use for most web pages is JSON, because it is not so wordy as XML. Even so, it is a bit unwieldy when used from the command line without some additional help, which comes in the form of the command-line utility `jq`, a lightweight JSON processor. This utility allows individual values to be selected out of a JSON structure without a lot of complicated syntax:

```
data.json
{
  "temp": 14.15,
  "pressure": 1009,
  "humidity": 87,
  "temp_min": 12.22,
  "temp_max": 15.56
}
```

Like other command-line tools, the `jq` command can process either a file or an input stream:

```
cat data.json | jq -r .pressure
>1009
```

A graphics program, preferably one that can be run from the command line, will display the photos. The `feh` program does all of this, and the `-F` option ensures the menu and taskbar are hidden. The `feh` image viewer can display a complete slideshow, or it can display a single image. The single-image display makes it the perfect tool and gives complete yet granular control over which images are displayed and for how long:

```
feh -F -x <picture fn>
```

The Raspberry Pi can have a large SD card or even an additional USB pen drive for picture storage, but a mutually accessible attached drive requires the Internet. A number of cloud solutions could make external storage available, but I wanted a solution that also had a friendly web interface.

Data Storage

Dropbox, one of the original data drives on the Internet, provides free disk space (2GB) and a web interface for access. Although 2GB is not a lot of space for many tasks, it should be plenty for holding a few photographs. (See also the "Google Drive" box.)

The `dbxcli` command-line tool [6] lets you access Dropbox as a drive. This client is available for a few platforms other than Linux, such as macOS and BSD. Yet before you can use this client, you must connect it to the Dropbox account with your credentials. The first time you run the client you are prompted for verification (Figure 1).

Entering the URL into your browser in response brings up the login dialog. When you enter your credentials, you will be asked if it is acceptable for this tool to have access to your Dropbox account (Figure 2). Pressing the *Allow* button brings up your verification code in a new dialog. Simply copy and paste the displayed code into your terminal session to finish setting up the `dbxcli` client.

Although it isn't important to know how this client works, it creates a `.config/dbxcli` folder in your home directory that contains an access code to be used as necessary by the `dbxcli` client. Once the code has been given, the client is set up, allowing control of Dropbox files from the command line. Client operations are somewhat limited but provide more than enough functionality to access Dropbox as an Internet-connected disk. The only operations needed for the picture frame are directory listings (`ls`), uploading files (`put`), downloading files (`get`), and deleting files (`rm`).

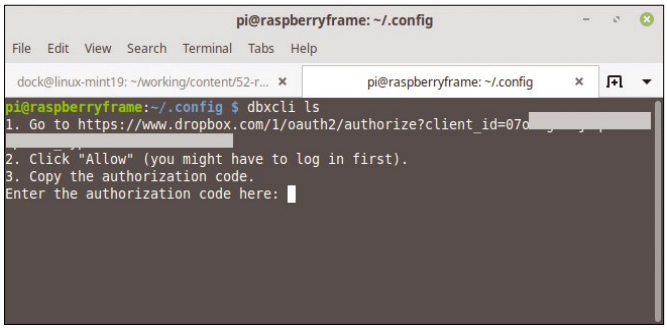


Figure 1: First time running the `dbxcli` client.

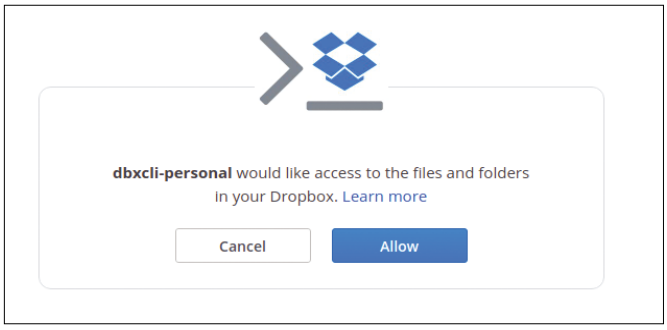


Figure 2: Enabling Dropbox access.

Weather Data

The weather functionality doesn't have very complex requirements: I simply want to get the current temperature and display a brief forecast. Initially I thought about opening up a web page and trying to capture the information there; however, that seemed a bit underhanded as well as difficult. I did find a number of sites that provide free weather information, but OpenWeather [7] was the easiest site to use.

OpenWeather provides different levels of service for retrieving data. Depending on the level of service you select, the data can be as old as a few hours or as recent as 10 minutes. It is possible to get weather forecasts, weather maps, the UV index, weather alerts, or just the current weather. The pricing model depends on the amount of weather or type of historical information needed.

The picture frame will only need at most a few calls per hour, which is considerably less than the 60 calls per second you can make with the free account. OpenWeather is affordable but it also makes its data available over REST API calls, thus eliminating the need to download libraries and compile programs to retrieve data. The one catch is that you have to sign up to use their service.

Once you sign up [8], you receive a key (Figure 3); with this token you are able to download your weather data by the API. You can even create additional keys for specific purposes.

OpenWeather has quite a number of REST calls, each of which fetch different types of weather data. Each API accepts a number of parameters, which makes it easy to customize the results. What is really nice is the consistency of parameters between different REST calls, making it easier to understand when you are looking at new REST calls.

When signing up, you receive an API key, or APPID (Figure 3), which is required for OpenWeather to associate your requests with a valid user for the premium function calls. However, if you do not provide your APPID parameter, you cannot even call their free services.

OpenWeather provides quite a number of ways to define for which city or location you want to retrieve weather. The most unambiguous method would be to find your desired city in the list of cities. You can get a list of all weather stations

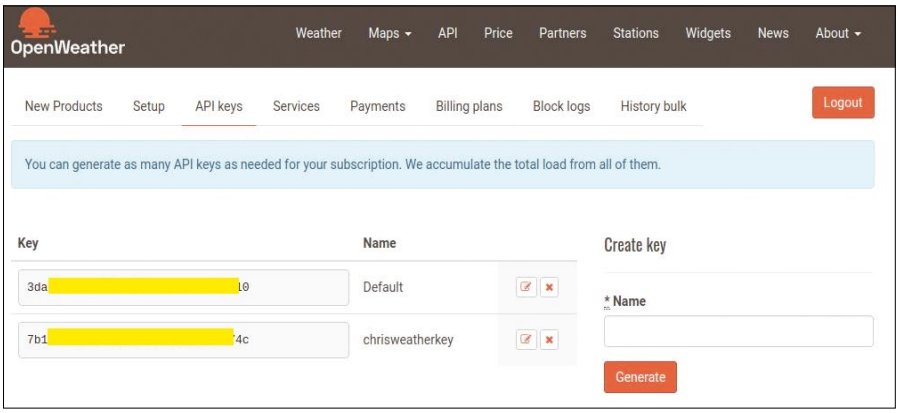


Figure 3: The OpenWeather key maintenance screen.

and their unique IDs, as well as their locations as longitude, latitude, city name, and country [9]. The API allows you to pass in the city name and country, although this doesn't work quite as well for the United States.

The problem I had with their list was trying to distinguish between which entry was Minneapolis, Minnesota, and which was Minneapolis, Kansas. OpenWeather does not appear to make any distinction between the US states, so passing in a popular city name might not return the city data you are expecting. To resolve this problem, I had to use a reverse lookup site on the Internet [10].

To select the current temperature or retrieve a weather forecast, you would use:

```
http://api.openweathermap.org/data/2.5/weather
http://api.openweathermap.org/data/2.5/forecast
```

The parameters in Table 1 can be used with your requests, as well. If you want to keep your network requests to a minimum, you can use the `group` call. Simply pass in a comma-separated list of city IDs to receive a JSON structure for each:

```
http://api.openweathermap.org/data/2.5/group?id=2925533,5037649
```

Although not a comprehensive look at all the calls you can use to retrieve data from this service, it does show all of the important methods I used to generate my weather forecast.

Creating PNG Files

The data from the weather service is returned in JSON format, which is pretty convenient if you are parsing this information by JavaScript but somewhat less convenient for a graphics application.

With the use of `jq`, I easily extracted the values I needed, but that still left the problem of how to display the data. The only data the picture frame displays now is photographs. Although I could probably overlay information to the screen a number of different ways, I chose to create an image from the data with the use of a familiar technology: Apache FOP, a print formatter that can be used in document processing to convert XML data

Table 1: REST API Parameters

Parameter	Description
appid	The value associated with your OpenWeather user account.
q	The city name, which should include the country for clarity (i.e., London, UK) but is not mandatory.
id	The weather site.
units	The default is Kelvin, which isn't so friendly for normal people; however, you can change the units to *metric* or *imperial* (i.e., *units=metric*).
mode	The data returned is in JSON format by default. If you would prefer either XML or HTML, you can select that format with the *mode* parameter.
lang	Most data returned by OpenWeather does not need internationalization; however, a small description field gives a hint about the weather, such as *light rain* or *sunny*. By assigning the *lang* parameter, you can see these weather hints in the language of your choice.

into a PDF file by XSL-FO (extensible stylesheet language – formatting objects). Also, you can use XSL-FO to convert the data into other output formats, including rich text (RTF) or PostScript (PS), but it is just as easy to create a TIFF or a PNG from the same stylesheet.

Because FOP makes it possible to create PNG output files, all of my data on the picture frame will be simple graphic files, simplifying my processing. Apache FOP is open source, so I just downloaded the latest version [11]. With a working Java environment, I could create PNG files.

Raspbian comes preinstalled with OpenJDK, the version of which depends on which Raspbian image you use. My problems appeared the first time I tried to run Java; I kept receiving an error about initialization:

Error occurred during initialization of VM
Server VM is only supported on ARMv7+ VFP

This error turned out to occur because OpenJDK 11 is compiled for a processor chip in the newer Raspberry Pis. Although the processor is backward compatible for older code, the older chip cannot run the OpenJDK 11 compiled for the new processor. The solution to this problem was to remove the current version of the JDK and install OpenJDK 8:

```
sudo apt-get purge openjdk*
sudo apt autoremove
sudo apt-get install ➥
  -y galternatives openjdk-8-jdk
```

Formatting Objects Template

The Apache FOP print formatter XSL-FO combines quite a few technologies from the extensible stylesheet language and XPath. The main difference between Apache FOP and other markup languages such as HTML is separation between the data and the formatting instructions. HTML does not always separate the data to be formatted from the formatting instructions, which makes it difficult to have multiple views of a single source of data.

The XSL-FO template describes in detail how exactly the data should be formatted. The description has the physical output size, margins, and orientation of text. Other supported types of formatting are not dissimilar to those in a normal word processor, which allows you to change fonts, text size, justification, and colors of text or background. Also, you can create lists, import graphics, create watermarks, and make tables of all types.

Listing 1 shows the template necessary to create a small 7cm square document with 2cm margins, in the middle of which appears *Hello World*.

Because of the general complexity of XSL-FO, it would take quite a few pages to describe a simple template properly. A number of books fully describe creating such templates if you are interested in learning more about this technology. You can see my favorite tutorial on the topic online [12].

Source Code

To make REST API calls, parse the resulting JSON output, and coordinate the processes, you could use any of a number of languages. The script or program will power a picture frame that displays new photos every few minutes, so you don't need sub-second response times. With this in mind, I created a Bash script to control and process both the image and weather data.

I enjoy writing Bash scripts, and I am also interested in how others solve various scripting problems; however, I suspect I am in the minority on this subject. Therefore, I will show a few small excerpts of the various scripts, and the rest of the scripts are available as a download [13]. The scripts must handle four tasks:

- Update photos on the Raspberry Pi
- Select and display a picture
- Download the weather forecast
- Generate an image from forecast

These tasks are interesting enough to take a deeper look.

Updating Photos

The simplest way to update the Raspberry Pi might be to wait for some special trigger file to be generated. In this case, you would just list the files in a directory on Dropbox, and when the trigger file exists, you would transfer the files.

Although I might use this method to coordinate a file transfer at work, this solution doesn't exactly scream user friendly. A different method would be to copy over all the files periodically to the Raspberry Pi. In this case, the synchronization is just a matter of removing all existing photos, gathering a list of all images on Dropbox, iterating through that list, and copying over each file. In the future, I might revise the synchronization mechanism, but for now, I will do this once a week.

Listing 1: XSL-FO Hello World

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <xsl:stylesheet version="1.1" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
03     xmlns:fo="http://www.w3.org/1999/XSL/Format" exclude-result-prefixes="fo">
04 <xsl:template match="weather">
05     <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
06       <fo:layout-master-set>
07         <fo:simple-page-master master-name="pnglayout" page-height="7cm"
08                 page-width="7cm" margin-top="2cm" margin-bottom="2cm"
09                 margin-left="2cm" margin-right="2cm" >
10           <fo:region-body />
11         </fo:simple-page-master>
12       </fo:layout-master-set>
13       <fo:page-sequence master-reference="pnglayout">
14         <fo:flow flow-name="xsl-region-body">
15           <fo:block>Hello World</fo:block>
16         </fo:flow>
17       </fo:page-sequence>
18     </fo:root>
19 </xsl:template>
20 </xsl:stylesheet>
```

The first step is to gather up a list of files to be copied over and then to iterate through that list. A rather cute way of doing this is to use both `head` and `tail` to get a specific entry from the list:

```
NAME=`head -${IDX} $WORKFILE | ➥
  tail -1`
```

Downloading the file from Dropbox is then just a matter of running `dbxcli` with the name of the current file to retrieve.

Selecting and Displaying a Picture

Displaying a picture is as easy as running the `feh` program with the name of the file to be displayed over the desktop. The entire goal of the picture frame is to allow for dynamic updating of photos, but even keeping that in consideration, the photos will remain fairly static most of the time; therefore, I maintain the current list of files to display, which is generated each time the files are copied from Dropbox.

Displaying all photographs is just a matter of looping through the list of existing files. Each photo should be displayed for at least a few minutes. I decided that the script shouldn't run continuously; instead, from the list of pictures, it should choose the current image, display it, and exit. The script, then, is run by a crontab file every few minutes. For this to work, the script will need to maintain its state between runs.

The `pickpicture` method in Listing 2 retrieves the current item in the list to be displayed, advances the index, and then displays the current picture.

Downloading Weather Data

Retrieving the weather for a given city is quite easy. The OpenWeather site does the authentication once at the beginning and gives you a time-unlimited key, so you only need to pass the key in with each call:

```
CMD="http://api.openweathermap.org/➥
     data/2.5/weather?➥
     id=${CITYID}➥
     &appid=${APPID}➥
     &units=metric"
curl -s $CMD > $TODAY
```

The script simply makes a call to the OpenWeather URL with the appropriate parameters, and you receive a JSON object in return from the `curl` utility. The returned weather structure is not overly complicated: Simply select each field of interest.

Extracting the values was straightforward except for sunrise and sunset, because these values are stored as a Unix UTC value. The good news is that this can easily be converted to a more user-friendly value with the `date` command:

```
# parse out data from data file
sunset=`cat $TODAY | ➥
        jq -r '.sys' | ➥
        jq -r '.sunset'`
sunset=`date --date=@$sunset ➥
        +'%H:%M:%S'`
```

Generating Forecast Image

Apache FOP was a convenient choice because I can define a template and merge it with a data file, although it glosses over the generation of the template and XML data file. The Bash script parses all weather data and simply creates an XML data file that matches the requirements of the template I have created. With FOP and OpenJDK installed, I generated a PNG file with:

```
fop -xml <xml input fn> ➥
    -xsl <xsl template> ➥
    -png <output fn>
```

In my weather template, I tried to include as much information as possible. The script examined the weather description and selected a matching graphic, which is friendlier than pure text. Unfortunately, the forecast has a lot of information (Figure 4).

Miscellaneous Configuration

The standard default for the Raspberry Pi is for the screen to blank after a certain

Listing 2: Display Bash Code

```
01 pickpicture()
02 {
03    IDXFILE=$PICDIR/index.dat
04    cd $PICDIR
05
06    IDX=`cat $IDXFILE`
07    CNT=`cat $PICLIST | wc -l`
08    NAME=`head -$IDX $PICLIST | tail -1`
09
10    IDX=$(($IDX + 1))
11
12    if [ $IDX -gt $CNT ]
13    then
14       echo 1 > $IDXFILE
15    else
16       echo $IDX > $IDXFILE
17    fi
18
19    showpicture $NAME
20
21    cd $INSTDIR
22 }
```

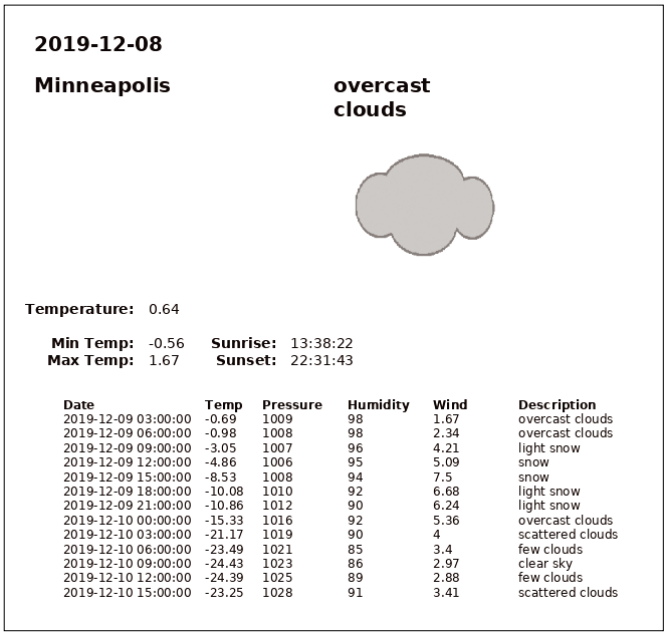


Figure 4: Weather and forecast.

Figure 5: The monitor with its picture frame.

amount of inactivity. Inactivity is inevitable when you don't have a keyboard or mouse attached to the picture frame system. To control this behavior, go to the X server configuration file (i.e., `/etc/lightdm/lightdm.conf` for the Raspberry Pi) and add the line:

```
xserver-command=X -s 0 -dpms
```

This setup might already exist in the default section but will need to be uncommented. According to online sources, you will find this line under the `[SeatDefaults]` section, but in my case, the section was called `[Seat:*]`.

Scheduling

The various scripts that perform the individual tasks are run by the cron daemon (Listing 3). Because only one thing can be displayed on the screen at a time, the crontab entries have been organized in such a way that twice an hour the weather and forecast are retrieved and displayed at times that normal photographs are not.

Lessons Learned

Although creating something new with my old Raspberry Pi was really fun, I did encounter a few problems along the way. My first problem was mainly self-inflicted: I had hoped it would be possible to find an inexpensive, custom-made wooden frame online. Although I found a number of options, the real cost was shipping, which was almost as much as the cost of the frame itself.

In my last minute dress rehearsal, one difficulty turned out to be that an old USB cable I was going to bundle with the picture frame wasn't delivering enough power. Ensure you test the exact hardware you are planning to use.

The most unexpected problem was networking. My initial development used an Ethernet connection, and I never had any problems connecting. Only during my final testing with a WiFi dongle did I encountered networking problems when the network periodically stopped working. After some research, it seems that others had similar problems in the past and have solved this by restarting the networking component. I created a small script to check the Internet connection and restart it if necessary. Running this script once a minute from the root crontab solved my networking problems.

All the necessary logic for displaying the photographs and gathering the current weather forecast can be done with just a few hundred lines of Bash script (Figure 5). This new electronic picture frame should be useful for many years to come. ■■■

Listing 3: Crontab

```
#
# next image
0,10,15,20,25,30,40,45,50,55 * * * * /home/pi/screensaver/showpic.sh > /dev/null 2>&1
#
# get and show current weather
4,34 * * * * /home/pi/screensaver/getweather.sh > /dev/null 2>&1
#
# simply copy everything from Dropbox each week
1 1 1,7,14,21,28 * * /home/pi/screensaver/forcesync.sh > /dev/null 2>&1
```

Info

[1] SheevaPlug: *http://linuxgizmos.com/the-sheevaplug-nas-mini-pc-is-back-with-dual-a53-sheeva64/*

[2] LED cube : *https://www.youtube.com/watch?v=HpUvgoLtos0*

[3] "Streaming lullabies with a Raspberry Pi Zero" by Christopher Dock, *Linux Magazine*, issue 203, October 2017, pg. 26, *http://www.linux-magazine.com/Issues/2017/203/Streaming-with-a-Pi-Zero*

[4] Installing operating system images: *https://www.raspberrypi.org/documentation/installation/installing-images/*

[5] gdrive: *https://github.com/gdrive-org/gdrive*

[6] dbxcli: *https://github.com/Dropbox/dbxcli*

[7] OpenWeather: *https://openweathermap.org*

[8] OpenWeather sign-up: *https://home.openweathermap.org/users/sign_up*

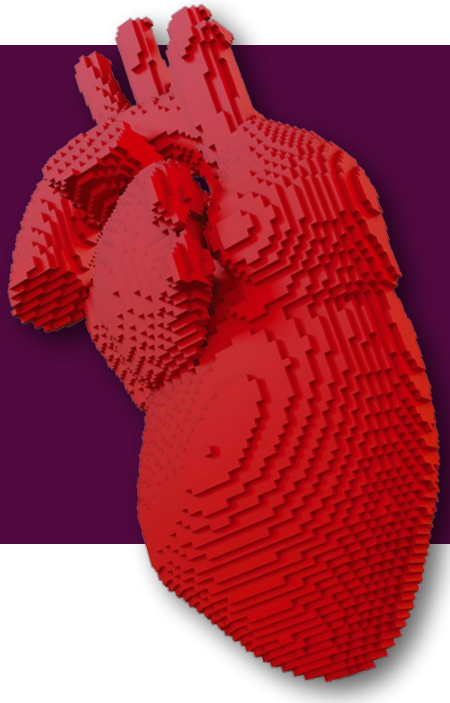
[9] OpenWeather download: *http://bulk.openweathermap.org/sample/city.list.json.gz*

[10] LatLong.net: *https://www.latlong.net/Show-Latitude-Longitude.html*

[11] The Apach FOP Project: *https://xmlgraphics.apache.org/fop/download.html*

[12] Machine generation of PDF files: *https://blog.paranoidprofessor.com/index.php/2016/11/15/machine-generation-of-pdf-files-the-easy-way/*

[13] Code for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/233/*



MakerSpace

An inexpensive open source 3D printer

The Joy of 3D Printing

Printy offers an inexpensive, open source DIY 3D printer kit, with a license that paves the way for future open source solutions. *By Bruce Byfield*

If a single technology defines crowdfunding and open hardware, it would be 3D printing. Not only did 3D printers appear at about the same time as crowdfunding and open hardware, but the demand for cheaper 3D printers has always been high. Even more importantly, 3D printers are a way for other open hardware projects to print cases and replacement parts, overcoming the problems of manufacturing hardware. Kickstarter alone lists 685 projects related to 3D printing over the last decade – however, most are proprietary, only 37 are open source and even those may use obsolete technologies [1]. Jay Lin's Printy project [2] is an attempt to bridge that gap with an inexpensive do-it-yourself open hardware kit (Figure 1) whose crowdfunding campaign will be underway by the time this magazine hits the newsstands.

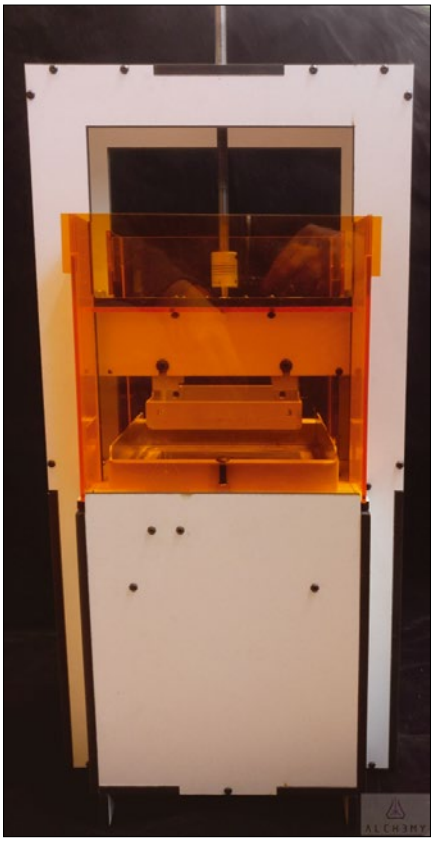


Figure 1: An assembled Printy prototype.

Printy's Crowd Supply page spells out the current situation: "SLA [Stereolithography] 3D printers are not only expensive, they are typically proprietary. For an enthusiast trying to find the right balance between low cost, high print quality, and open source design, the best advice one could offer has long been 'pick one'. The current market has left the 3D printer community stranded in a disconnected ecosystem full of walled gardens. With Printy, we aim to dismantle those walls and help reconnect that community. And what better way to share the joy of 3D printing than by first sharing the joy of building a 3D printer?"

Lin discovered both open source and 3D printers through RepRap [3], a project to build an affordable 3D printer that began in the UK in 2005. "The

idea of printing any object was appealing," he says. "I purchased a Printrbot kit [4], as it struck the right balance between cost and performance. The assembly was great, and I learned a lot about Fused Deposition Modeling (FDM) printers [an obsolete printing technique, 5] via the construction process. That is when I realized that a majority of the joy came from learning about how the printer works rather than what can be printed. Thus, the seeds of Printy were planted. The idea was to make a [modern] Stereolithography (SLA) kit printer [6] to share the joy of construction and learning about this next level of 3D printer technology." In other words, Lin's goal is to build a high resolution 3D printer whose output matches the quality of top-of-the-line proprietary ones.

Coming relatively late into the 3D printer market, Lin was able to learn from the mistakes of others. In particular, Lin was struck by Peachy Printer's [7] crowdfunding campaign. In September 2013, Peachy Printer raised over $650,000 in its campaign to build a $100 3D printer. However, after countless delays, in 2016, the developer was forced to admit that his partner had embezzled $320,000 of the funds raised, putting an end to the project and leaving backers with neither their money nor a printer. Peachy Printer was not the first 3D printer project to collapse – nor even the first crowdfunding campaign to renege – but its size, as well as the initial enthusiasm for it, devastated the development community, including Lin. "I watched my friends' investments into Peachy Printer vaporize in a spectacular way," Lin says. "Since Printy was already in the design phase, I made sure to focus on completing the product before anyone else risked a dime."

Lin continues, "To further reduce buyer's risks, I made use of standard components wherever possible, including the NEMA 17 stepper motor, 8mm linear rails, bearings, endstop switches, Arduino board, and metric screws [Figure 2]. In case any part breaks or goes missing, those components can be easily sourced. A buyer who has all the common components should only need to purchase a few unique components instead of the whole product. Moreover, a hobbyist with access to advanced tools such as laser cutters, soldering equipment, and sheet metal equipment should be able to make those unique components."

To help users, Lin is keeping the bills of materials as spreadsheets on GitHub [8] (Figure 3). The spreadsheets are literally a piece-by-piece inventory of the hardware and tools required to construct a Printy, detailing screws, Allen wrenches, zip ties, safety goggles, and everything else. The last column in each row of the spreadsheets adds exact sizes, as well as advice like "Try to source from the same vendor" – a precaution that could eliminate minute differences in the sizes of the components. The spreadsheets end with images of the components in order to minimize the possibility of confusion.

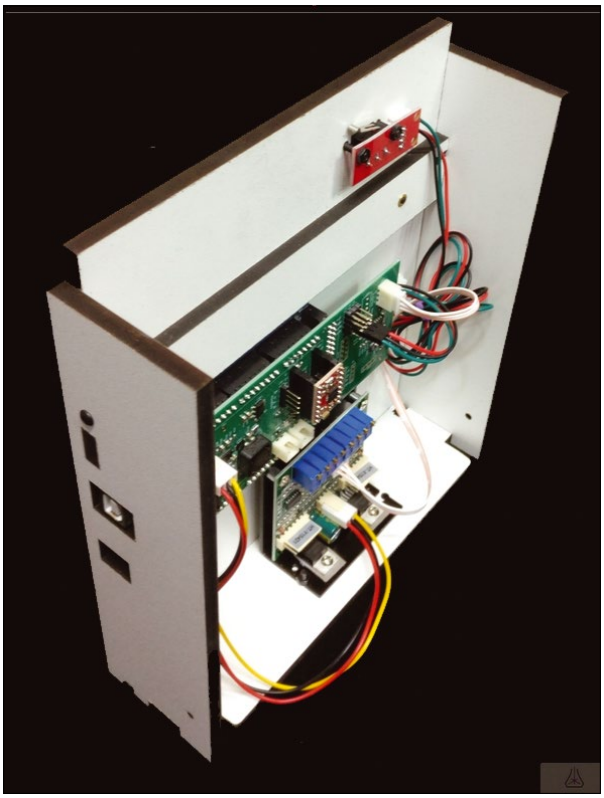


Figure 2: Using easily replaceable standard components makes assembling Printy simpler.

The build instructions [9] are similarly comprehensive (Figure 4). Although I have worked extensively as a technical writer in my time, it is no exaggeration to say that I have never encountered such a thorough set of instructions, even among those written by professional writers. As Lin notes, assembling a Printy includes such key steps as "the placement of the laser and a series of mirrors to guide the laser beam into 2D patterns. To create the third dimension, there is a platform that traverses up and down along a linear rail." All these steps require precision, a fact that is reflected in the instructions.

Printy Specs and Licensing

Table 1 shows Printy's basic specifications as listed on CrowdSupply. These specifications are a trade-off with the cost. For example, the print dimensions are relatively small compared to other SLA printers. Similarly, the reso-

B20 Misc Kit

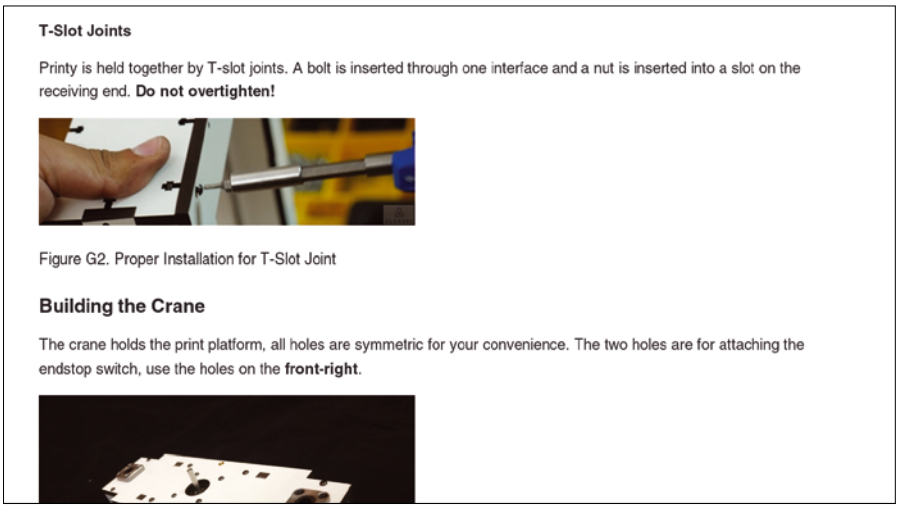
	A	B	C	D
22				
23				
24	Misc Kit	202-0003 Rev C		
25	Item #	Name	Quantity	Comment
26		1 Adhesive zip tie mount	4	Higher quantity package is OK if cheaper
27		2 Double sided tape	1	3M VHB RP32, 0.75in width, >= 36inch cut strip or r
28		3 Zip tie	100	.1" wide >= 3" long. Higher quantity package is OI
29		4 Wire loom	1	2ft
30		5 Laser safety goggles	1	Must block 405nm
31		6 Single sided tape	1	Polyester tape, 3M 1318-1 or equivalent, 0.75in wi
32		7 Metric allen wrench set	1	Must include M2, M2.5, M3. Keep cost <$0.50
33				
34	Mechanical	202-0002 Rev A		Try to source all from the same vendor
35	Item #	Name	Quantity	Comment
36		1 ACME T8x1 200mm with cart	1	T8, ACME, 1mm pitch, 200mm long, 8mm diamete

Figure 3: All of Printy's components have comprehensive lists of parts and required tools.

lution is about half and the speed about a quarter those of top-of-the-line proprietary printers.
The finished product documentation will explain how the skilled and adventuresome can go beyond Lin's design. Lin explains that "anyone can scale each dimension of the laser-cutting DXF files and create a Printy Deluxe. By tweaking the schematics, one can substitute the 14-bit DAC with a 24-bit DAC to enable resolutions that will be the envy of industrial 3D printers. [In addition], the firmware can be ported from 8-bit AVR to a more modern 32-bit ARM processor and achieve much faster print speeds. I plan to document the production steps to aid this group of tinkerers so they can focus on experimentation instead of finding shops that can make their variants." For those willing to make the extra effort, the results should equal or perhaps even exceed the top-of-the-line proprietary printers.

These improvements are possible in part because of the use of the CERN Open Hardware Licence (OHL) [10]. The CERN OHL describes itself as being "to hardware what the General Public Licence (GPL) is to software," and, as Lin explains, is intended for "a project that involves mechanical, electrical, and firmware design." Like the GNU GPL, the OHL allows modifications to a project's elements so long as the modifications are released under the same license. Under the terms of the license, any of the modifications of Lin's basic design can become available to others, which justifies Lin's claim that, technically, "Printy compares favorably against much more expensive machines."

Already, the licensing promises one benefit. Printy's firmware is a fork of Marlin firmware, an open source project that is used even for proprietary printers [11]. Printy's innovation is to adapt a stepper-based firmware into a galvanometer-based machine, according to Lin, expanding the license's usefulness.


T-Slot Joints

Printy is held together by T-slot joints. A bolt is inserted through one interface and a nut is inserted into a slot on the receiving end. **Do not overtighten!**

Figure G2. Proper Installation for T-Slot Joint

Building the Crane

The crane holds the print platform, all holes are symmetric for your convenience. The two holes are for attaching the endstop switch, use the holes on the **front-right**.

Figure 4: Printy's build instructions are detailed and specific.

Table 1: Printy Specifications

Hardware license	CERN OHL v1.2
Software	Based on Marlin firmware, licensed under the GNU GPL
Print dimensions (X x Y x Z)	110x110x150mm
Resolution (X, Y, and Z)	50 microns
Maximum speed (X & Y)	> 600mm/s

By itself, Printy promises to be an inexpensive and open source solution for 3D printing. However, its licensing means that future open source solutions can be more easily developed because of what Lin has already done. "Design," as Lin says, "isn't the end of a project. I find the media, manufacturing, and community building aspects to be equally interesting. I would love to connect with anyone who is inspired to modify and create their own Printy. As for my next project, a desktop metal 3D printer is intriguing." ■■■

Info

[1] Kickstarter 3D printer projects: *https://www.kickstarter.com/discover/advanced?ref=nav_search&term=%223D%20printer%22%20%22open%20source%22*

[2] Printy: *https://www.crowdsupply.com/alch3my/printy*

[3] RepRap: *https://en.wikipedia.org/wiki/RepRap_project*

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

[5] FDM printers: *https://en.wikipedia.org/wiki/Fused_filament_fabrication#Fused_deposition_modeling*

[6] Stereolithography: *https://en.wikipedia.org/wiki/Stereolithography*

[7] Peachy Printer: *https://3dprint.com/133842/peachy-printer-embezzlement/*

[8] Printy's bills of material: *https://github.com/alch3my/printy/find/master*

[9] Printy's build instructions: *https://github.com/alch3my/printy/blob/master/Mechanical/Instructions/BuildInstructions.md*

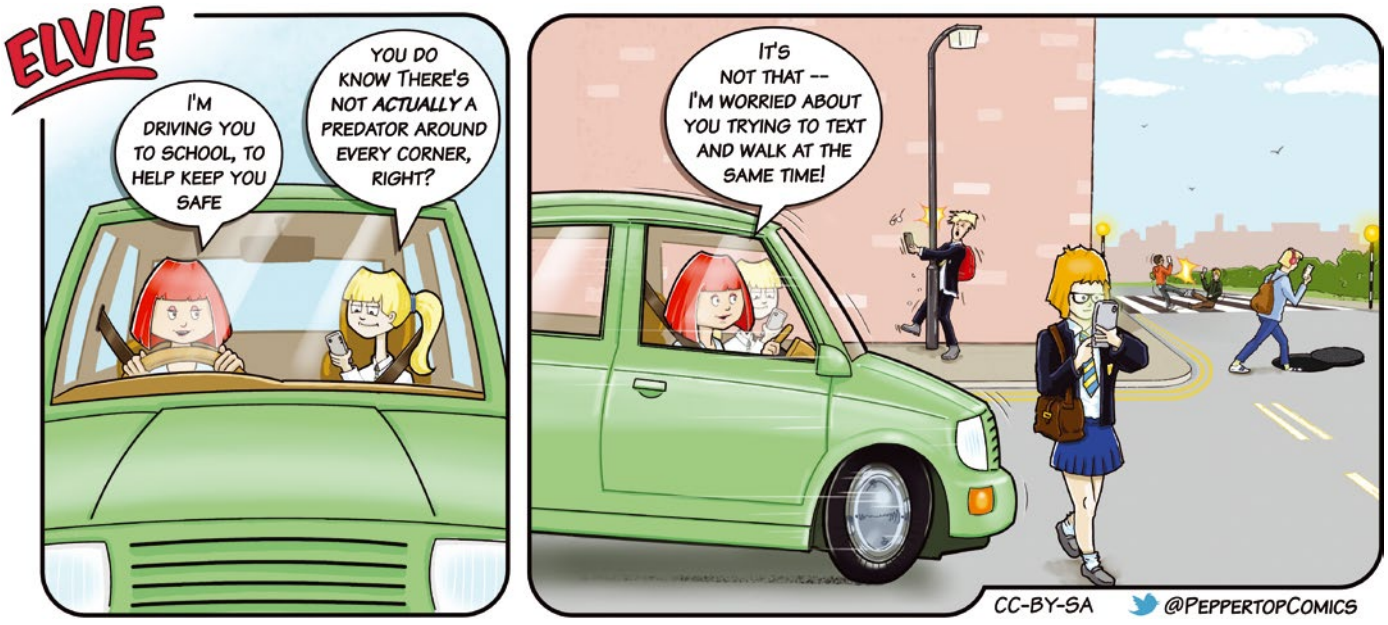
[10] CERN OHL: *https://ohwr.org/project/cernohl/wikis/home*

[11] Marlin firmware: *https://marlinfw.org/*

The popular line on the Internet is that you need to trade away your privacy to participate in the messaging and social networking communities, but the world of free software is all about avoiding such traps. Jami is a messaging app backed by the famous GNU project that protects your privacy and does away with servers altogether. We'll introduce you to Jami and help you get started with the joys of free communication. Also in this month's issue, we compare the Krita and MyPaint graphics tools and describe how to build your own homegrown notification tool using Bash scripts.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE



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

MADDOG'S DOGHOUSE

Maddog remembers when his computer first spoke to him – thanks to Yggdrasil – and the (just!) 150 floppy disk images it took to install Slackware. BY JON "MADDOG" HALL

Early Distributions

Tonight, I was talking to Bill Wright, one of the founders of LinuxFest Northwest (LFNW) [1], a great conference held in Bellingham, Washington, every year, and the conversation came around to early Linux stuff.

I told Bill that I am in the process of cleaning out my house in an effort at what older people like me call "downsizing," getting rid of things that we have collected over the years.

I have a *lot* of stuff that I have accumulated, and one way of reducing the amount is to give away some of these items at FOSSH events like LFNW.

In the course of this, Bill asked me the fateful question of "Do you remember Yggdrasil?"

Of course! Yggdrasil was an early version of GNU/Linux.

I was sitting at my desk at Digital Equipment Corporation (DEC) sometime in November of 1993, and I saw an advertisement in *Dr. Dobb's Journal* for a "Unix system complete with source code for 99 USD!"

I did not believe this, because I knew that a source code license for Unix was still in the tens of thousands of dollars per machine and that no one could ship the sources for that small an amount of money.

Still, $99 was an affordable amount of money for me at that time, and I decided to buy it even if it was a complete sham, so I sent my money away.

After a week or so the CD arrived accompanied by a thin, spiral-bound book. Unfortunately, I did not have a "PC" to run the code on. I had VAX workstations, MIPS workstations, and even Alpha workstations. Why should I have one of those weak, miserable, crappy Intel PCs?

So I mounted the CD in my drive on my Ultrix MIPS workstation, followed the directory tree down to the `man(1)` page directory and started reading the `man(1)` pages. I was impressed.

After I had spent a while looking at the man pages I unmounted the CD and put the package in my filing cabinet.

A couple months later, I facilitated Linus traveling to DECUS in New Orleans, saw Linux for the first time in May of 1994, and convinced Linus to port the code to the DEC Alpha. At the same time, I learned a lot about the state of the GNU/Linux project and some of the players in that market such as a company named "Yggdrasil."

After I returned to my office, I dug the CD out of the filing cabinet and looked closer. I read the manual and hunted down an Intel PC in the DEC Unix group.

The Yggdrasil CD actually used a live filesystem, so you did not have to install it, and when I booted that CD on that (weak, miserable, crappy) Intel PC, not only did it boot, but two graphical windows came up, played some short video segments, and then *the computer spoke to me*.

Sure, the "voice" was only a `.au` sound file playing back, but I was hooked! I *knew* that GNU/Linux was set for greatness.

Unfortunately over time Yggdrasil, like many of the early distributions, faded from sight, but I was then led to another great distribution called Slackware.

By this time, I had purchased my own (weak, miserable, crappy) Intel PC at home and was intensely working with GNU/Linux to understand the system and its limitations, if it had any.

I would dutifully install the Slackware distribution (and of course I wanted *all* of it) from the approximately 150 1.4MB floppy disk images that made up the distribution.

And of course I was too cheap to buy 150 of those floppy disks and too impatient for AOL to send me those floppies one per month (thank you, AOL!), so I installed about 10 floppies of data and started recording over the first ones with the information from the next disk images.

That worked fine until the installation failed on about the 50th disk.

I cried.

Then I saw an advertisement for another distribution named Red Hat, which installed off a CD-ROM! I ran into the night and bought a CD-ROM reader that attached to a sound card (as most CD-readers did in that day) and waited for the Red Hat distribution to show up in my postal mailbox.

In those days, getting GNU/Linux on the disk and able to boot did not solve all problems. On systems with an ISA bus, you had to type in all the device information about your graphics and networking cards and a lot of other devices. Most of the time, you got this from the previous operating system that ran on your computer, but it was never as easy as it is today.

As difficult as the installation was, it was still better than running that "previous operating system." ■■■

Info

[1] LFNW: *https://lfnw.org/*

Chat freely with Jami
Private Conversations

The messenger app Jami offers clients for all popular operating systems and promises maximum anonymity for chats, as well as voice and video calls, by dispensing with central servers. BY CHRISTOPH LANGNER

AOL, ICQ, MSN – the list of these once extremely popular messengers could be extended considerably. Today they are consigned to internet history, although some online providers still try to sell especially short or easy to remember ICQ numbers for thousands of US dollars.

But the world today uses WhatsApp. Over 1.5 billion users use this messenger, which Facebook acquired in 2014 for $19 billion [1]. Away from the mainstream, smaller messengers such as Wire, Signal, and Telegram have carved out a niche for themselves. At the top of the list of reasons why users choose one of these alternatives is the desire for more privacy.

Most messengers require central servers in the background, and this, in turn, means registering with your email address or phone number. This is not true for the open source Jami. In a similar approach to BitTorrent, the Jami messenger uses a distributed hash table to set up its own network and can therefore do without central registration [2]. You only need to register with the service if you want to have a unique username that can be found using the messenger's search function – but this does not mean supplying any personal data. Jami is available for Linux, Mac OS X and Windows, as well as for smartphones with Android [3] and iOS [4], and TV sets with Android TV.

Installation

The developers provide installation files and package sources for Fedora, Debian, and Ubuntu. Listing 1 shows you how to install Jami on Ubuntu 19.04; instructions for other distributions can be found on the project's website [5].

If you don't want to add a package source to your system right away, you can also pick up deb packages for Ubuntu (16.04 to 19.04) and Debian 9 and 10. However, at the time of writing, the links to the 32-bit version of the program pointed into a black hole. Users of Arch Linux will find the program, dubbed *jami-gnome* here, directly in the distribution's repositories.

After the install, you can load the program by selecting the *Jami* entry in the application menu. Jami will not work without a user account; you can create one by clicking on *Create local account*. You can also choose to restore an account that was saved previously (Figure 1). Click on *Advanced…* for the option to create a SIP account, (see the "Jami Today" box).

In the screen that then appears, Jami asks you to enter a profile name. You can enter an image that can also be captured directly via webcam (Figure 2). You can protect the account

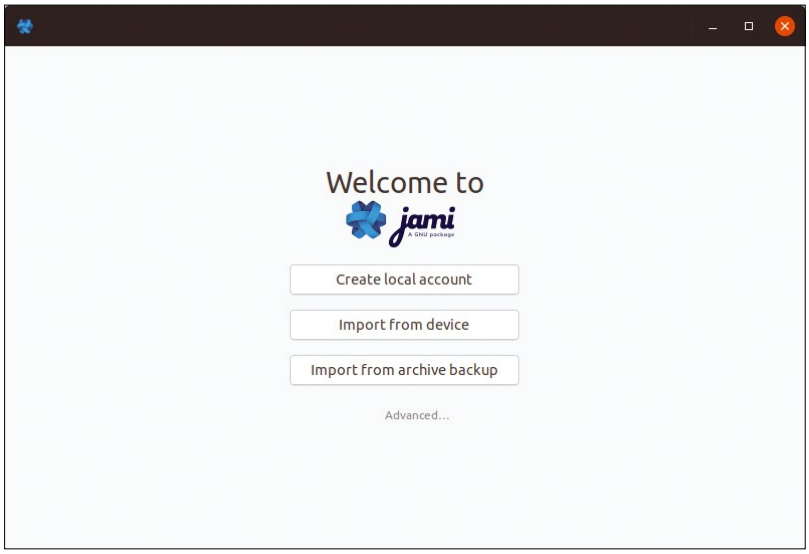


Figure 1: On the first launch, you will need to create a new Jami account, but you do not need to prove your identity to the service with an email address or a mobile phone number.

Listing 1: Install Jami on Ubuntu 19.04

```
$ echo 'deb https://dl.jami.net/ring-nightly/ubuntu_19.04/ ring main' |
  sudo tee -a /etc/apt/sources.list.d/ring-nightly-main.list
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
  A295D773307D25A33AE72F2F64CD5FA175348F84
$ sudo add-apt-repository universe
$ sudo apt update
$ sudo apt install ring
```

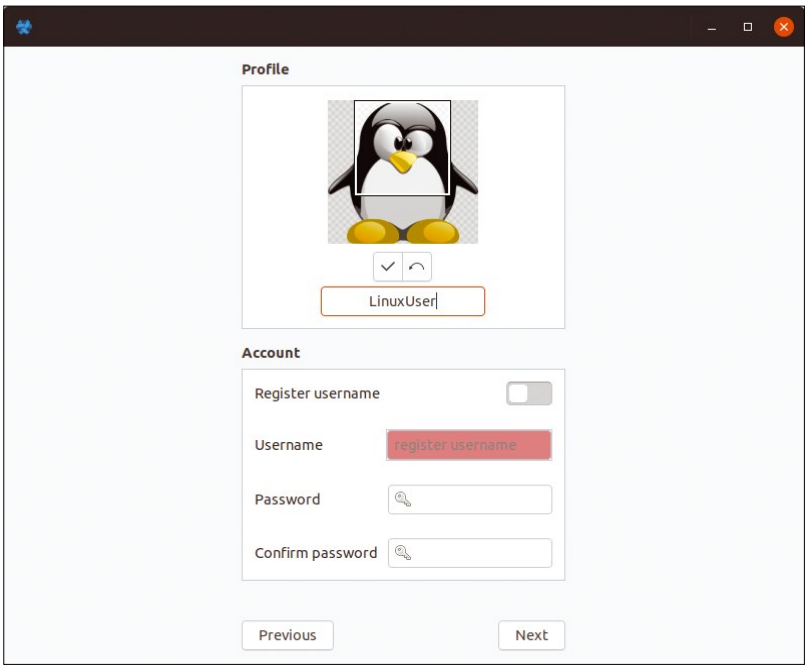


Figure 2: Anyone who wants to can remain completely anonymous with Jami. The *Register username* option is only there to help other people find you through the search feature.

against unauthorized access with a password. Whether or not you register a username is up to you. With a Jami account, you can claim a unique and freely selectable username that gives other Jami users the ability to find you using the program's search function. However, the messenger does not store anything else in the account; the operator cannot log chats or your contact list. Communication is also encrypted end-to-end and therefore protected against sniffing by third parties.

First Chats

At first, Jami shows you very little in the main window. The sidebar contains a search function and, later on, your saved contacts. In the middle, you will find your Jami username – if you use one – or your ID in the form of a cryptic combination of letters and numbers.

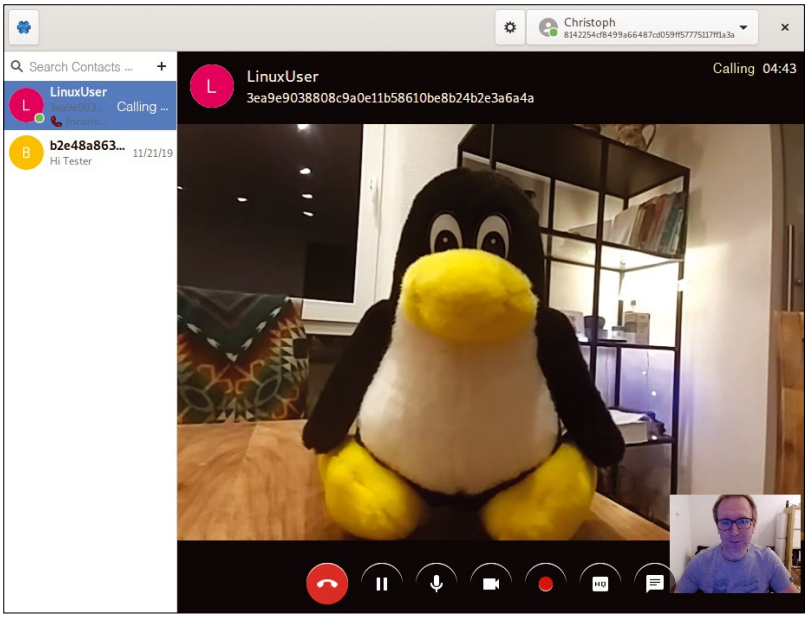


Figure 3: Jami supports audio and video chats, and video chats with multiple callers. The number of callers is only restricted by the bandwidth of the call initiator's Internet connection.

To contact friends or acquaintances, you need to get their IDs or just their usernames. If your contact uses Jami on a smartphone, you can alternatively set up the connection using a QR code, which you call up using the button next to the ID (or name) and then scan it using the Jami mobile app.

When the first message appears, you are asked if you want to accept the contact. From now on, the chat will appear in the sidebar. If your contact uses a Jami account, you will find the chat under that username. Otherwise Jami will only show the cryptic ID; the chat cannot be renamed as yet.

In addition to text-only messages, Jami supports sending files and audio and video conversations with two or more participants in resolutions up to 4K (Figure 3). In the latest version, Jami also gives users the ability to send audio and video messages, much like WhatsApp.

Another new feature of the revised version is support for transferring your own desktop. To do this, start a video call and then right-click on the image. In the context menu you can then choose between sharing a screen section and sharing the entire screen. On Linux, however, screen sharing only works in combination with the X server. Users with Wayland will first need to switch to the classic Linux desktop.

You also have the option to transfer a video file by selecting the *Import file* menu item. The *More Information* option provides statistics and details, such as the frame rate or the audio and video codecs used.

Configuration

Jami hides all settings options behind the gear icon that you find next to the account name in the application's header bar. Go to *General* and you can then configure, say, whether Jami always

Jami Today

What is now Jami used to go by the name of SFLphone and then later on Ring. Originally the program was intended as a plain vanilla softphone for SIP providers. The function still exists, even though the messenger functions are now the main focus. The current "Free as in Freedom" version now offers cross-platform videoconferencing (until now, this function was only available in the Linux version), audio and video messaging in the style of WhatsApp, and a dark theme. In addition, Jami now allows users to broadcast their own desktop or individual windows instead of a webcam image.

starts automatically when you power on the system, and which events it should notify you about. In *Media*, you can define the audio and video devices to be used along with the webcam resolution for video chats. This is also where you enable hardware acceleration.

In everyday use, the last tab, *Account*, turns out to be the most important. This is where you register a Jami username, define an image for the account, and temporarily deactivate or reactivate the account (Figure 4). You will also want to export the account, because a backup of this type is the only way to restore an existing account when you install a new system.

In the lower section of the dialog, Jami offers the option to use the active account on multiple devices, such as a smartphone or a second PC. To do this, select *Link another Device* and enter the password for the account in the following screen. Click on *Export to network* to initiate the connection. Jami will then generate a PIN in the form of an abbreviation like `nix66ilp`. Note that you need to complete the next step quickly, as the PIN code expires after a few minutes.

Now create a new account on the second device, but select the option *Import from device*. Then enter the *Password* and the previously created *PIN*. You will now see the previously created contacts on the second device, too, and will be able to start chats. Since Jami only stores the chat history locally, and not on a central server, you will only see the conversations that took place on the individual device. There is also no archive of the previous chats.

In practice, this means that you will automatically receive incoming messages on all connected devices. However, you will only ever see your replies on the device on which you wrote them. So you can't seamlessly switch between two devices with Jami. Data synchronization across multiple devices would require a central storage location, which Jami deliberately avoids in favor of privacy.

Conclusions

Testing Jami left me with some reservations. Text chats work reliably, but quickly become confusing if you communicate with many people who do not have a Jami account and therefore do not have a username. In the sidebar, there are many chats that only show the contact's ID as a distinguishing criterion. The developers urgently need to improve this, a suggestion has already been posted to the project's bug tracker [6].

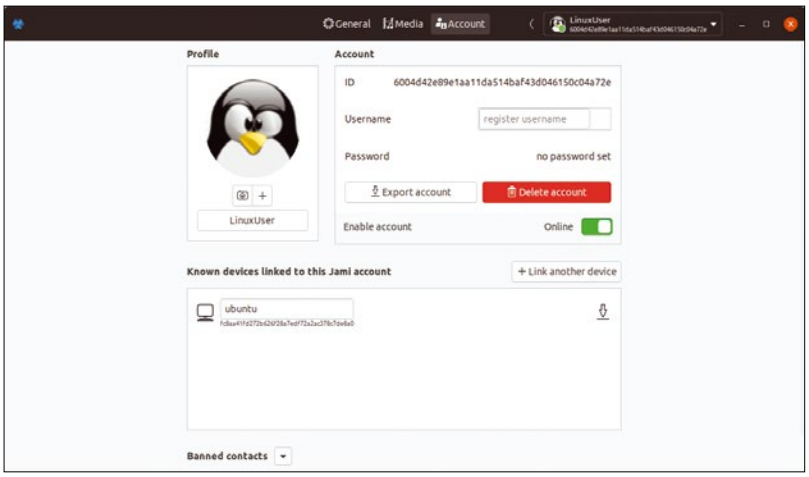


Figure 4: A Jami account can be used on multiple devices. However, because the service does not store messages centrally, you will not see the same message history everywhere.

When sending messages, problems repeatedly occurred in hands-on tests. Jami delivered text messages reliably, even to accounts that were used on several devices (such as on a PC and a smartphone) at the same time. Pictures or documents sent via Jami's encrypted network often got stuck. The client then reported *Initialize File Transfer*, but nothing happened at the recipient's end.

Jami was impressive in video chats with two or more participants. It is one of the few messengers in the package sources of common distributions that support videoconferencing without the need to register with a service. You will definitely want to give the program a chance and experience it for yourself. ■■■

Info

[1] "WhatsApp hits 1.5 billion monthly users": *https://web.archive.org/web/20180209063953/https://techcrunch.com/2018/01/31/whatsapp-hits-1-5-billion-monthly-users-19b-not-so-bad*

[2] Jami: *https://jami.net*

[3] Android: *https://jami.net/download-jami-android*

[4] iOS: *https://jami.net/download-jami-ios/*

[5] Installing on Linux: *https://jami.net/download-jami-linux*

[6] Suggested improvement: *https://git.jami.net/savoirfairelinux/ring-client-gnome/issues/952*

Draw and paint in Linux

Sketch Artist

If you are looking for an open source drawing program, Krita and MyPaint both offer graphic tablet support and brushes. Deciding which one works best depends on your specific needs. BY ALEXANDER TOLSTOY

When it comes to an open source alternative to Photoshop, most people think of Gimp. However, Gimp isn't the only raster graphics (aka bitmap) editor for Linux. For drawing and painting, users quickly realize that they need more than Gimp, including brushes, Wacom-style graphic tablet support, and artistic effects. Krita and MyPaint are two open source alternatives that are specifically designed for digital artists.

Krita [1], once the black sheep of the KOffice bundle, has received a lot of attention as a standalone application. A prominent participant in the Google Summer of Code since 2008, it has grown into mature, production-ready software with paid developers.

MyPaint [2] is a simpler, more basic drawing application with fewer features. However, simplicity can be a great advantage, attracting many talented artists, including David Revoy, the prominent Blender animation specialist and the creator of many MyPaint brushes. Although the latest stable version, MyPaint 1.2, dates from 2017, solid work is being done on version 2.0.

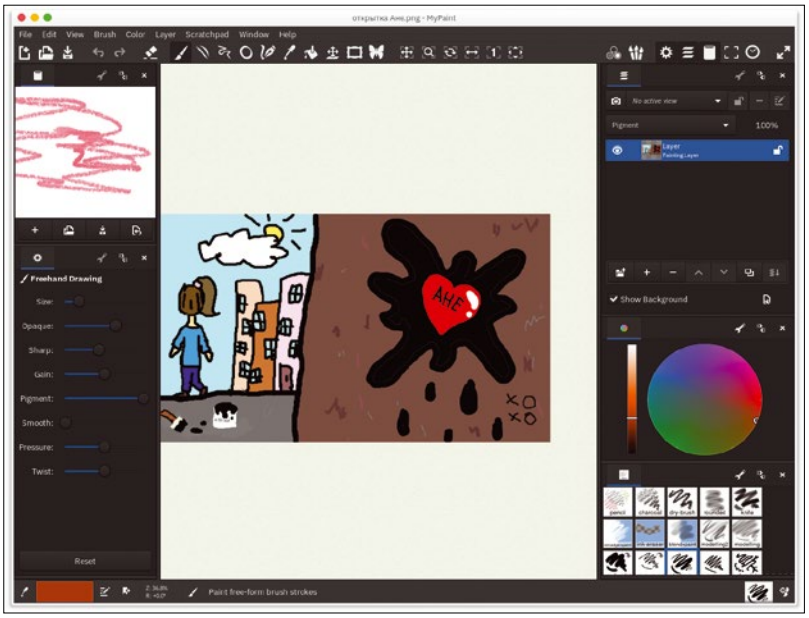


Figure 1: After arranging MyPaint's panels for brushes, layers, and settings, you're ready to draw or paint.

Each application has its devoted users and areas where it performs best. If you haven't used either Krita or MyPaint recently, revisit both apps, in particular the MyPaint 2.0 beta. Both Krita and MyPaint have developed into mature, widely-adopted drawing suites.

MyPaint

With its strong focus on drawing, it makes little sense to use MyPaint for anything else. If you do choose to open a photo in MyPaint, your options are limited: You can use MyPaint's layer and blending options, superimpose text, or add a sketch to embellish the photo. It doesn't offer features found in Gimp, such as healing and cloning brushes, geometric transformation tools, filters, or color correction.

Instead, MyPaint gives you an infinite canvas for drawing and painting; you don't even need to predetermine your image size. With an understanding of how artists work, MyPaint's limited tool selection lets you focus on the content, not the interface.

You may want to customize the initial MyPaint layout so that all essential tools and features are at your fingertips. You can add dockable panels with the brush assortment, color wheel, and brush settings. No matter how many panels you add, the central drawing area remains large and spacious, with panels to the sides (Figure 1). There are several ways of choosing the color (from RGB sliders to various color wheel types), seven sets of artistic brushes from different artists, and the traditional brush settings panel. Brush settings include *Size*, *Opaque*, *Sharp*, *Gain*, *Pigment*, and more. Similar to Gimp, panels can be stacked to form tabs.

MyPaint supports many graphics tablets including Wacom. The pressure sensor works correctly for all supported models. You can also tweak ad-

vanced settings; the *Edit | Edit Preferences* dialog is largely dedicated to pressure handling and assigning custom actions to keys and buttons. Global Pressure Mapping (Figure 2) is a configurable curve that changes brush behavior. Adjusting the curve can subtly change how the digital brushes behave, mimicking real-life brushes.

Drawing with MyPaint is quite comfortable. You use the *F* and *D* keys to change the current brush size, the mouse wheel controls zooming, and dragging with the left mouse button pressed while pressing the spacebar lets you move around the canvas. After 10 seconds of inactivity, unsaved work is backed up in the OpenRaster format (`.ora`) and placed in `~/MyPaint`, minimizing the chance of losing your work.

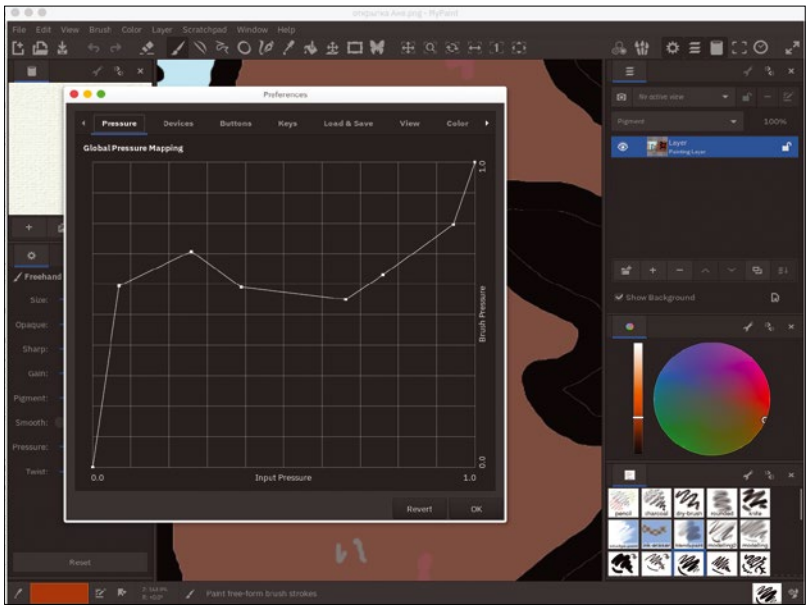


Figure 2: Global Pressure Mapping in MyPaint lets you experiment with brush behavior.

Krita

With its main toolbar offering numerous features, Krita looks very professional. If you connect your tablet and start drawing, you'll notice that Krita is well-tailored for artists. A right mouse click (most likely implemented in your pen as the extra button) brings up the radial palette with a set of brushes, a color selector, scale control, and access to extra brushes.

Krita offers a breathtaking number of brushes: pens, pencils, inks, chalks, airbrushes, markers, scratchers, textured brushes, pastels, and much more. All brushes are divided into groups and powered by 17 brush engines. Select a brush by pressing the *Choose brush preset* button on Krita's top toolbar. Then press *Edit brush settings* and configure opacity, blending, size, ratio, pressure, and more. The number of settings depends on the brush engine, ranging from fewer options with the Quick Brush Engine to several options with the Pixel Brush Engine (Figure 3). The number of possible combinations of brush settings is close to infinite.

Unlike MyPaint, Krita has cloning and healing brushes, as well as a variety of image transformation tools. It is possible to use Krita for general image manipulation instead of Gimp, but one should keep in mind the following limitations:

- Although the cloning brush works well, the Smart Patch tool (used for healing) is too automated and can deliver mixed results.
- Krita can lag and show poor performance when working with large images.
- Color correction tools are very limited.
- There is no printing support.

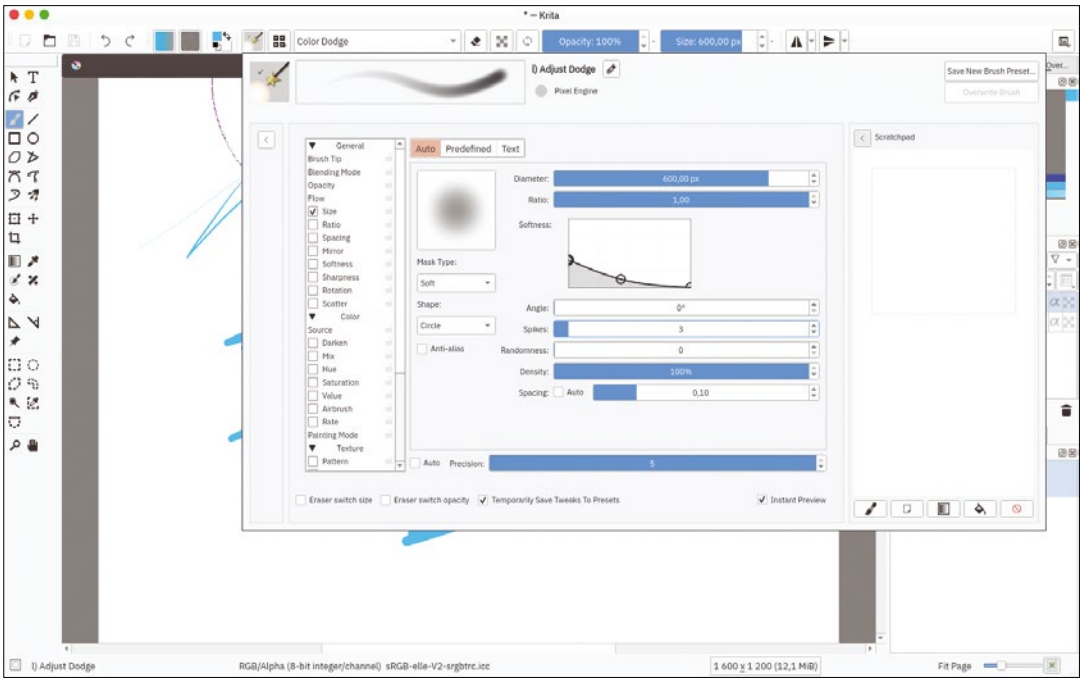


Figure 3: Krita sports several brush engines, some of which hide an impressive number of options.

Early versions of Krita featured a limited set of built-in effects and filters, which prevented the application from competing with Gimp equally. Recent Krita versions support GREYC's Magic for Image Computing (G'MIC) – a rich set of artistic, geometric, abstract and enhancing filters (Figure 4). You will find these filters under the Tools menu.

Krita offers more than just drawing tools. Additional advanced features allow you to create animations and comic templates, as well as offer support for vector objects. Under *Settings* | *Configure Krita*…), you will find a noticeably larger set of options than found in MyPaint. You may want to adjust the RAM limit and swap file location, change the color space to 32-bit RGB, or apply a custom International Color Consortium (ICC) profile to your screen. Regardless of the menu, Krita has more available tools and options than you might expect. A good example is color blending for layers and brushes (Figure 5). Apart from *multiply*, *screen*, and a dozen other expected variants, Krita also allows blending things by lightening, darkening, and mixing based on different color models and extra algorithms. It's quite sophisticated; consequently, once you master Krita's options, you will have reached an unprecedented level of precision when it comes to graphics.

Interoperability

Both MyPaint and Krita can export and save files in OpenRaster (MyPaint's default), PNG, and JPEG. Krita offers additional options, including its default native format (`.kra`) as well as PSD, XCF, and a bunch of traditional raster formats (XPM, TIFF, etc.).

Although both MyPaint and Krita support OpenRaster, you should be aware of OpenRaster's limitations. For instance, Krita's support of selection masks for layers will be lost if you save the file as OpenRaster. To preserve this, you should use Krita's default `.kra` format, which supports all of Krita's features.

While MyPaint is not designed to import Photoshop PSD and Gimp XCF files, Krita can handle these types of files with some limitations. Krita's reverse-engineered PSD importer supports layer groups and masks, blending modes, and transparency. It does not support text layers and vector objects. Still, it's better than nothing.

Input Devices

Both applications are designed to work with pressure-sensitive graphics tablets. MyPaint in particular suffers without a dedicated input device. If you don't have one, you can try to draw with a mouse, but you will not get very far. This deficiency shows in many of its features, such as Scratchpad, which offers a separate canvas space where you can make sketches, mix colors, and do various auxiliary tasks without cluttering the main canvas.

Krita is far less hardware-dependent. Without a tablet, you can still master comics, animate your artwork, and manipulate images. If Krita had more developed color management tools (e.g., shad-

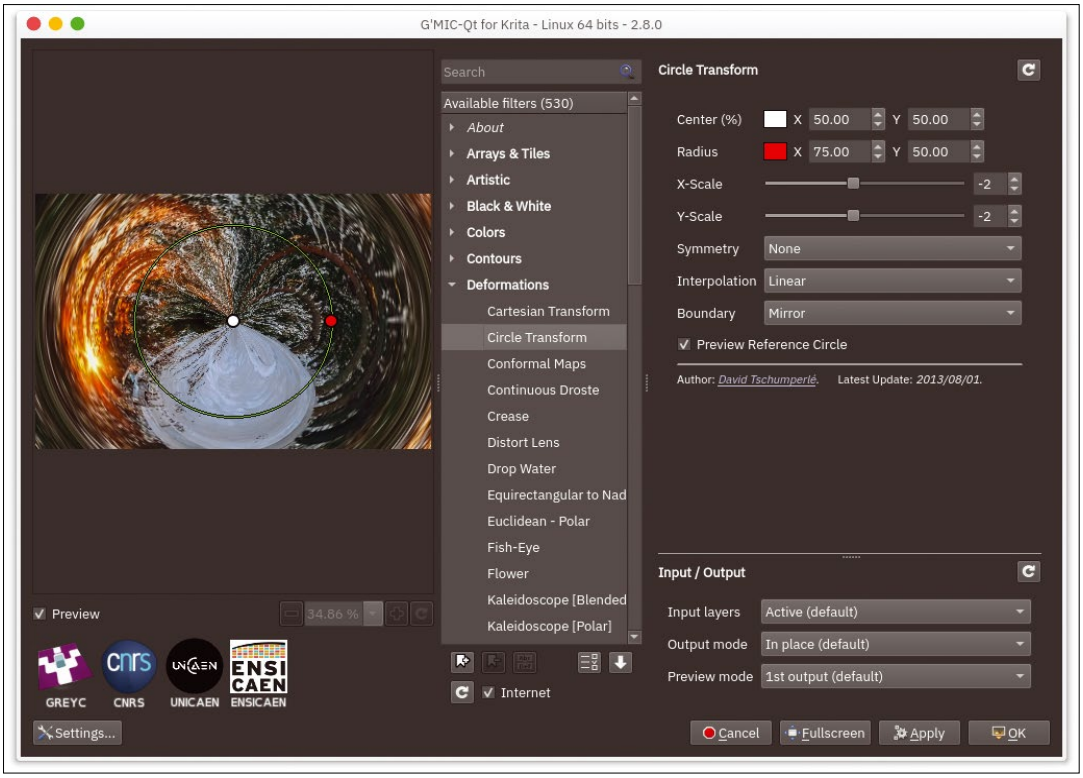


Figure 4: Krita isn't just for drawing: G'MIC support lets you play with artistic filters and effects.

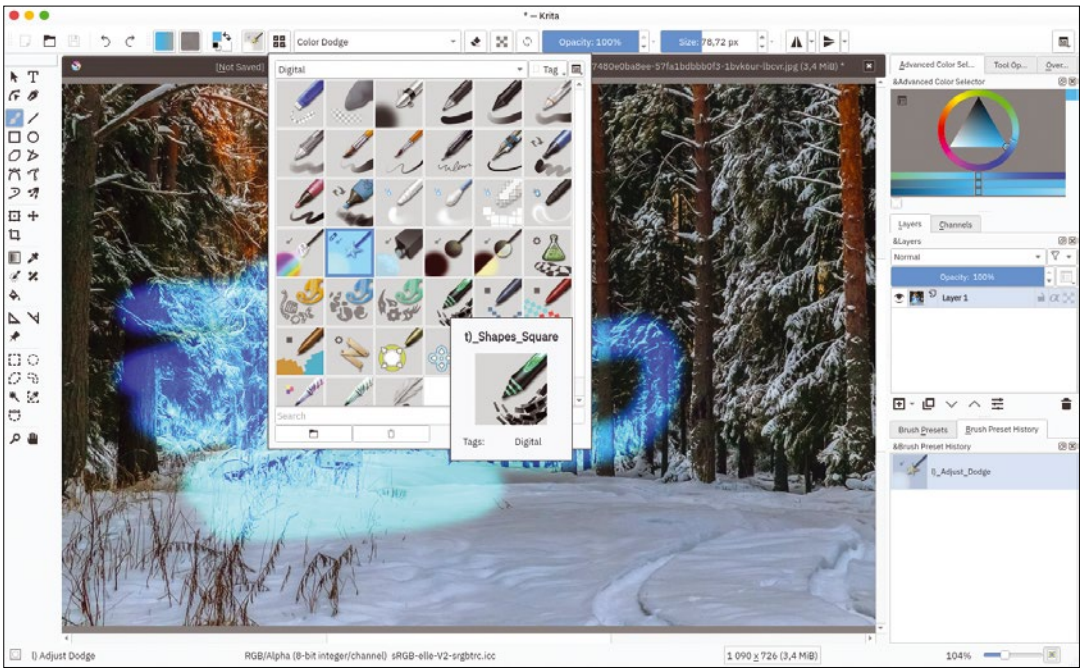


Figure 5: Krita offers a large selection of brushes, including cloner and healer brushes.

ows and highlights), a better healing brush, and a decent print preview dialog, it could compete directly with Photoshop.

Making a Decision

Choosing between MyPaint and Krita depends on your individual needs. If you just need a robust mediator between your drawing tablet and Linux, then MyPaint is preferable. While it has fewer brushes than Krita, some artists think MyPaint brushes are better and more true-to-life. MyPaint's purpose is to provide the most authentic drawing experience. With this narrow focus, you can't criticize MyPaint for falling short on things it was not meant to do. Additionally, MyPaint settings are easier to navigate and use. For instance, changing the pen pressure in MyPaint is very intuitive.

Krita offers a complete, end-to-end solution for sketching, painting, and creating graphics from scratch. While MyPaint is superb in simple drawing, Krita also lets you draw comics, textures, animated scenes, retouch photographs, distort and transform any raster images, and apply artistic effects. However, Krita can't fix pale, underexposed, or blurred images, and it does not allow you to print. A decade ago, Krita did support printing, but it was so buggy that the developers decided to remove this feature.

In terms of performance, Krita has greatly evolved in recent years with improved OpenGL acceleration for the canvas, as well as the ability to calculate brush dabs on several CPU cores at one time. However, both contenders have important gaps. MyPaint's infinite canvas isn't that infinite if it consumes your available RAM without notification, and there is no dedicated setting to remedy this. Using large brush sizes in both MyPaint and Krita can slow down your computer. However, since Krita relies on hardware acceleration, your mileage can vary here depending on your hardware, drivers, and OS. Therefore, it is worth testing Krita on Windows, Linux, and macOS for comparison. MyPaint performance is somewhat less affected by your setup: It is fast with smaller sketches and slower with larger ones.

For beginners or artists who just want to draw and paint, I would recommend MyPaint. For more advanced users or anyone who wants to learn something new, Krita offers a versatile all-in-one bundle, which resembles Adobe Photoshop in terms of GUI without its paid subscription or other limitations. ■■■

Info

[1] Krita: *https://krita.org*

[2] MyPaint: *https://github.com/mypaint/mypaint/releases*

The Author

Alexander Tolstoy is a long-term Linux enthusiast and a tech journalist. He never stops exploring hot new open source picks and loves writing reviews, tutorials, and various tips and tricks. Sometimes he must face the bitter truth thanks to the inhuman `fortune | cowsay` command that he thoughtlessly installed in `~/.bashrc`.

Adapt the appearance of the GRUB boot menu, boot screen, and KDE splash screen

Variations on a Theme

Power users can adjust the look of their desktops in openSUSE with just a few clicks. We'll show you how to customize the GRUB boot menu, the boot splash screen, and the KDE start screen.

BY PETER KREUßEL

The splash screen and other startup images are a familiar sight for any Linux distribution. Although the desktop background and color scheme can be quickly customized to suit your tastes, the GRUB boot menu (Figure 1) and the boot splash screen (Figure 2) are far more difficult to customize. This article describes how to modify and unify your system by changing the GRUB boot menu and the boot and KDE splash screens to fit your personal preferences. The following examples are based on openSUSE, but the procedures are similar for other Linux systems – especially KDE-based systems.

Start with YaST

If you're working in openSUSE or another SUSE variant, you can use YaST to select a different theme for the GRUB boot menu. Use the graphical tool in *System/YaST Bootloader* (Figure 3). Select the `theme.txt` file for your choice of theme. YaST will do the rest of the work.

Unfortunately only the default theme is preinstalled on openSUSE. However, there is a *grub2-theme-breeze* package for a retrofit that matches the plain, monochrome, original KDE start screen. Pling.com [1], the former OpenDesktop.org, has a choice of around 350 themes. Unpack the zip archive you download from the site and – working with root privileges – move the subfolder with the `theme.txt` file into the `/boot/grub2/themes/` directory.

Then start the YaST *Bootloader* module and open the *Kernel Parameters* tab. Click on *Browse* to the right of the input box for the *Console theme*, and navigate to the theme folder you copied to `/boot/grub2/themes/` in the file browser. When you get there, select the `theme.txt` file and press *OK* in YaST. After rebooting the system, you can stand back and admire the new look of the boot menu (Figure 4).

Editing a Theme

Creating your own GRUB theme is somewhat complicated. It is easier to adapt an existing theme to suit your requirements. To create a background image and colors that you like, you need to be familiar with a few elements of the GRUB theme description language [2].

First copy the `/boot/grub2/themes/openSUSE/` folder to your home directory for editing. Then

Figure 1: Even the first sign of life from openSUSE, the GRUB menu, is green.

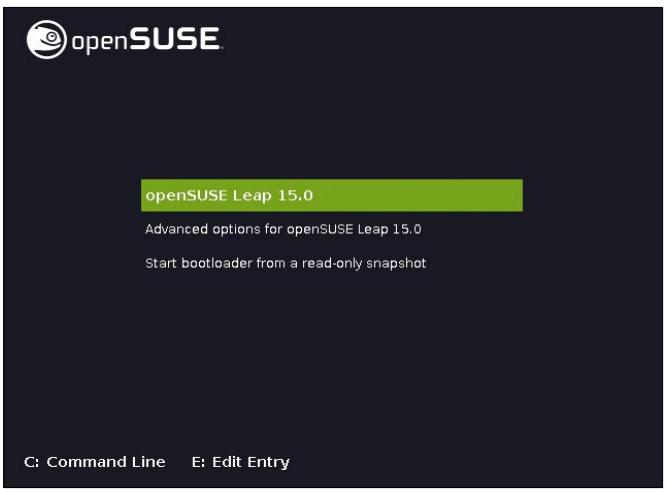
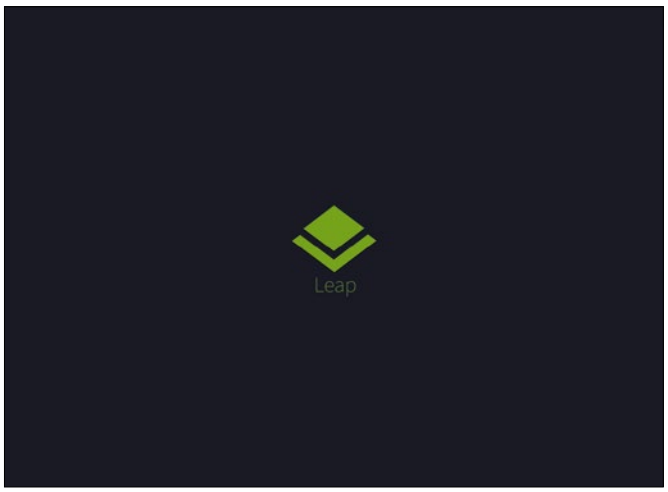


Figure 2: OpenSUSE only offers a uniform green for the boot splash screen.



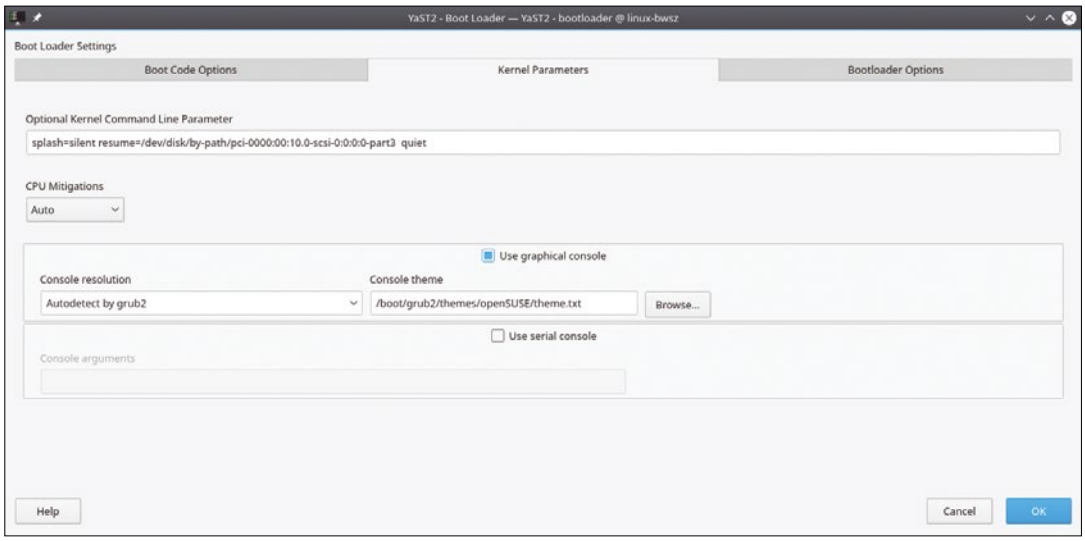


Figure 3: In the center tab of the YaST *Bootloader* module, select the `theme.txt` configuration file of a GRUB theme installed on your system.

open the `theme.txt` file in a text editor. The file consists of five parts : a definition of global properties at the beginning and the sections `+ image {}`, `+ boot_menu {}`, `progress_bar {}` and `+ hbox {}`. You define the openSUSE logo, the countdown time before auto-booting, the boot menu itself, and the bottom line with the keyboard shortcuts.

To adjust the colors of the GRUB menu, you need to change some statements in `theme.txt` and the PNG files in the themes folder. The hexadecimal value to the right of `desktop-color:` defines the background color. The KDE KColorChooser program returns the appropriate values with the help of the default KDE color picker. Figure 4 shows a GRUB menu, where the values for `desktop-color` in the global header, as well as `fg_color`, `bg_color` and `border_color` in the `+ progress_bar {}` section, have been adjusted.

The colors of the bars that highlight a selected menu item are not based on a setting in `theme.txt`, but on bitmaps. This means that 3D effects can be realized, although the openSUSE theme does not use any. To adjust the color of the highlight bar, open the `highlight_c.png` file in Gimp (Figure 5). The `slider_*` files are responsible for displaying a scrollbar, which only appears if there is not enough space for all the items in the start menu.

In Gimp, click on the upper color box just below the tool palette (highlighted in Figure 5). In the *HTML notation* field of the Gimp color selector, enter hexadecimal values as in `theme.txt`. Use the *Fill* tool (also highlighted) to color the formerly green areas to reflect your wishes. The `highlight_c.png` file consists of only one pixel. Open in Gimp and zoom in until the scale is large enough.

The important thing to know about 3D designs is that the `highlight` and `slider` settings result from the values for `selected_item_pixmap_style` (=`highlight_*.png`) and `scrollbar_thumb` (=`slider_*.png`).

The file names for the `highlight_*` and `slider_*` files include final characters defining the position on a map: `_c`, `_w`, `_n`, `_s`, `_e`, `_ne`, `_nw`, `_se`, and `_sw`. For instance, `highlight_c.png` refers to the center of the screen, and `_w`, `_n`, `_s`, and `_e` refer to the four cardinal directions.

A background image is far more appealing than a monochrome background. The image must be a PNG or JPEG image with a color depth of 8 bits that you copy into the theme folder. In `theme.txt`, add `desktop-image: "file.png"` at the start of the global block.

Figure 4: A cosmetic color change for the boot menu does not require any in-depth knowledge of the GRUB configuration syntax.

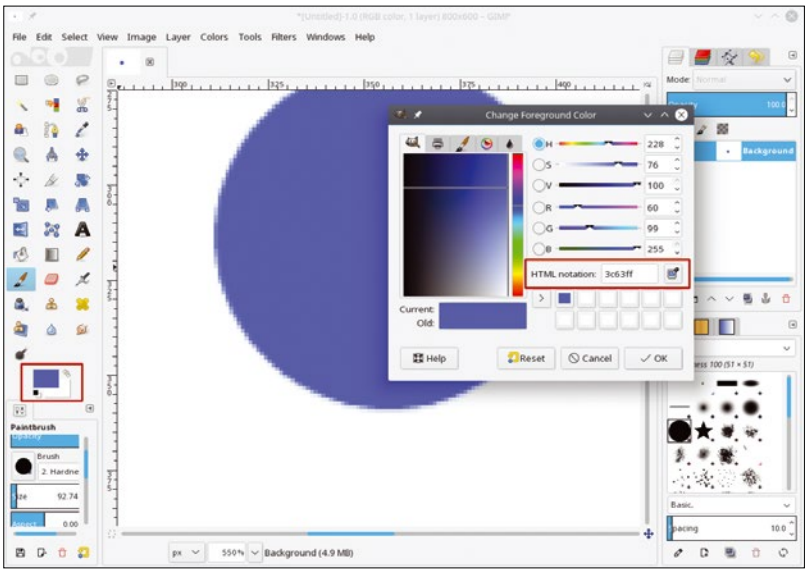


Figure 5: In graphical mode, openSUSE's GRUB configuration uses bitmaps for the highlight bar. Gimp gives you an easy option for changing the color.

Working with administrative privileges, copy the theme folder to `/boot/grub2/themes/` and select the customized `theme.txt` file in the YaST *Bootloader* module. Even if you create new versions where the path to `theme.txt` does not change, you still need to open the YaST module each time and update the GRUB configuration by pressing *OK*. All other aspects of GRUB themes are explained in the GRUB manual [2].

Boot Animation

After booting the system with the bootloader, the boot splash screen (Figure 2) appears; it remains visible for a few seconds during system startup. This splash screen is produced by Plymouth, a tool that starts early in the boot process and displays a graphical image. Numerous Plymouth themes are available on Pling.com [3]. You can change the splash screen with a simple command.

Unpack the downloaded archive in `/usr/share/plymouth/themes/`. If several subfolders exist there, you will recognize the actual theme folder by the fact that it contains a configuration file with the `.plymouth` extension. Then run the `plymouth-set-default-theme -l` command and look for the new theme. The `plymouth-set-default-theme -R` command then installs the theme.

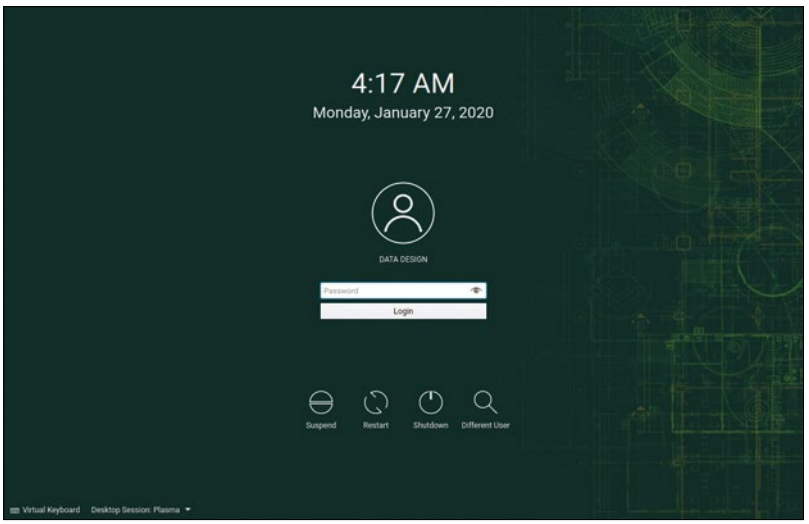


Figure 6: The green background of the KDE login screen can be changed with just a few clicks in the system settings.

You do not need to reboot the system to view the new theme. Install the *plymouth-x11-renderer* package, launch `plymouthd` as root, and then call `plymouth --show-splash`. The Plymouth screen is displayed in a window. Press the right arrow key to continue the animation. However, the window cannot be closed so easily; it is programmed to remain in the foreground. To get rid of it, press Ctrl+Alt+Esc on KDE and click on it. Before performing another test, run `plymouth --quit`.

Plymouth comes with a set of plugins that load PNG files. Unfortunately, these files are not documented. The naming conventions from the demo themes (packages *plymouth-theme-<XXX>*) can be used if necessary. However, almost all the themes available online use the *Script* plugin preinstalled by openSUSE, which comes with good documentation [4].

Scripting Language

The *Script* plugin evaluates the instructions in a `theme-name.script` in the theme folder. The specially developed scripting language is extensive. So let's have a look at the minimalistic *Simple-Image* [5] theme, which displays a centered image at boot time and does not use any animation.

In as-delivered condition, the graphic shows an Arch Linux logo. To replace it, edit the `img.png` file in the theme folder or replace it with a PNG with a resolution of 1920 x 1080 pixels. You must reinstall the theme after each change to the image. With an absolute minimum of script code, *Simple Image* is a good basis for experiments.

The content of the `simple-image.plymouth` theme configuration file is shown in Listing 1. The configuration provides a name and description and selects the *Script* plugin. The `[script]` configuration block for the plugin specifies the directory for the image to be displayed and the name of the script file. To create a new theme, simply adjust these paths and the file name of the `.plymouth` file itself to match the theme name you have chosen.

You can see the `simple-image.script` file in Listing 2. It creates an image object and calculates the screen center (lines 1 to 4). In computer graphics and also in Plymouth scripts, a sprite is a moving image object (line 6). Lines 7 and 8 place its coordinates at the computed center of the display. The last line defines `RefreshFunction` as the `refresh_callback`, which is called at each step of the boot process. Here it simply sets the opacity of the image to `1` and the stacking order to `15`.

You can extend this rudimentary framework by elaborating on the `refresh_callback()` function. You could successively increase the `Opacity` from

`0` to `1`, fade in more images, or move sprites over the screen with algorithms of arbitrary complexity using `SetX()` and `SetY()`. The script plugin documentation lists all the available commands.

Yet Another Splash Screen

The display manager starts after the boot splash. For KDE, openSUSE installs SDDM (Figure 6). When starting KDE, you will then see a second splash screen (Figure 7). The background image of the KDE login manager can be easily changed in the desktop environment's settings tool. This is all it takes to adjust the login screen to match the rest of the graphical design, as the login screen controls use a neutral look by default.

The KDE start screen, on the other hand, does not easily fit into a customized look. Newer KDE versions include a download function for themes in the system settings, as in many other places. This is still missing from openSUSE Leap 15.1, but it's not difficult to download and install themes manually from Pling.com [6]. Just unpack the zip archives in the `~~/.local/share/plasma/look-and-feel/` directory; you might need to create `look-and-feel/` first.

The start screens are written in QML, an interface description language provided by the Qt GUI framework used by KDE. To even touch on the Qt feature set would go beyond the scope of this article. However, QML startup screens can be "hacked" in a similar way to Plymouth boot screens by changing the graphics in the `Theme-Name/contents/splash/images/` folder.

As soon as you hover the mouse pointer over the preview of a *Start Screen Design*, a play symbol appears that starts the boot animation for testing purposes. This makes it quite easy to see which graphics from the `images/` subfolder appear at which position in the start screen.

Conclusions

Adapting the KDE desktop to your own taste is a breeze. The GRUB boot menu, the boot splash animation, and the KDE start screen are a bit trickier. To install themes found on the Internet, all you have to do is choose the right folder to store the themes, and, in the worst case, you may need to know the right YaST module or command.

It is not quite as easy to create your own themes for these three system design elements. The best way to get started is to modify existing themes using the templates available on the Internet. Even with minor adjustments, you can achieve a customized, seamless look for the system. ■■■

Figure 7: Writing a theme for the KDE start screen requires QML programming skills if you want to do more than simply replace the graphics that are displayed.

Listing 1: simple-image.plymouth

```
[Plymouth Theme]
Name=Arch Linux Simple Image
Description=This is a plymouth theme which simply displays an image
ModuleName=script

[script]
ImageDir=/usr/share/plymouth/themes/simple-image
ScriptFile=/usr/share/plymouth/themes/simple-image/simple-image.script
```

Listing 2: simple-image.script

```
01 image = Image("img.png");
02
03 pos_x = Window.GetWidth()/2 - image.GetWidth()/2;
04 pos_y = Window.GetHeight()/2 - image.GetHeight()/2;
05
06 sprite = Sprite(image);
07 sprite.SetX(pos_x);
08 sprite.SetY(pos_y);
09
10 fun refresh_callback () {
11   sprite.SetOpacity(1);
12   spr.SetZ(15);
13 }
14
15 Plymouth.SetRefreshFunction (refresh_callback);
```

Info

[1] GRUB themes: *https://www.pling.com/browse/cat/109/order/latest/*

[2] GRUB theme file documentation: *https://www.gnu.org/software/grub/manual/grub/html_node/Theme-file-format.html*

[3] Plymouth themes: *https://www.pling.com/browse/cat/108/order/latest/*

[4] Plymouth script plugin: *https://www.freedesktop.org/wiki/Software/Plymouth/Scripts/*

[5] Simple-Image theme: *https://github.com/barskern/plymouth-theme-simple-image/*

[6] Start screen designs: *https://www.pling.com/browse/cat/488/order/latest*

FOSSPicks

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

Graham has just recorded a one-off podcast featuring the wonderful open source VCV Eurorack emulator, often written about here. He's now strongly considering doing a more regular synth-related podcast. BY GRAHAM MORRISON

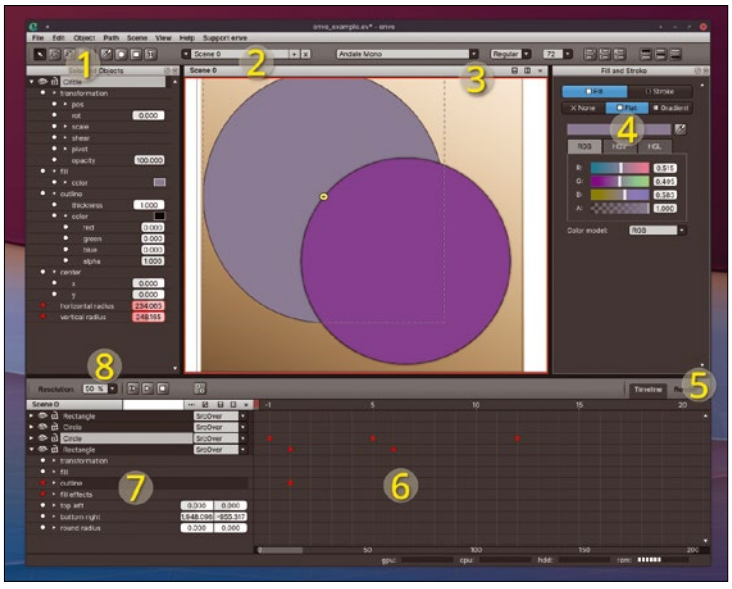
Drawing and animation

enve

Back at the dawn of home computing, there was a brilliant drawing and animation application for the Apple II and Commodore Amiga called Fantavision. It had been ported from the Apple to the Amiga, which was something of a surprise because Electronic Arts' Deluxe Paint ruled the platform and was often bundled with a new Amiga. Most users wouldn't need anything else. But there was something that set Fantavision apart, and this made it unique in a world full of pixel art – it was a vector drawing application. Images were constructed by drawing lines from one point to another. This was more arduous than the virtual spray paint of Deluxe Paint, but it meant that your images were resolution independent, much like SVG in the modern era.

There was another advantage: Vector images like these are much easier to animate, because you simply move the start or end point and take a snapshot of their location at a specific frame. A tweening algorithm then generates the frames in-between, making the resultant animation far more flexible and potentially smoother than the equivalent bitmap animation. It's what made Flash so hugely successful, and it's the same principle at the heart of this excellent new drawing and animation tool, enve. Enve is a cross between Fantavision, Inkscape, and Blender. Its drawing tools include paths, circles, rectangles, and text, and you can create a different fill, either gradient or solid, and change the outside stroke. You can also group these objects together, and perform raster/bitmap effects on them, including blur and drop-shadows. These tools and editing options cover all the principal requirements for drawing and design and are equally as capable of producing complex illustrations as they are simple sketches.

But the real power comes from enve's ability to animate these components. This is achieved from the timeline beneath the editing window, where you can see frame numbers stretched across to the right and objects listed vertically on the left. The object list, also available from the main view, is where you can drag and drop the hierarchy of the objects and unfold them to show all their parameters, from radius values to fill types. The clever part is that, by clicking on the small record button to the left of each value, you save this value at the current frame. Change both the frame and the values, and enve will smoothly animate the transition between one set of values and the next, creating a beautifully smooth and crisply rendered animation. When you want to output the final result, the *render* tab lets you generate a PNG or MPEG animation of the scene, at whatever resolution or frame rate you need. It's an application that may only be in a beta and early release state, but it already works brilliantly and can finally replace the immortal Fantavision.



1 Editing tools Though not a huge list, there are enough tools to create any primitive you need. **2 Scenes** Split a project into multiple scenes to be rendered separately. **3 Canvas** Edit images just as you would in Inkscape by grouping, ungrouping, and stacking the objects. **4 Fill, stroke, and gradients** Alongside these, there are special fill and rendering effects from the right-click menu. **5 Render** Lets you select any resolution and frame rate for your scene. **6 Timeline** Every parameter in your scene can be animated by taking a keyframe of its value on any frame. **7 Parameters** Choose which parameters to edit and which you want animated. **8 Scale** Lower the resolution for previews, and there's GPU acceleration if you have the hardware.

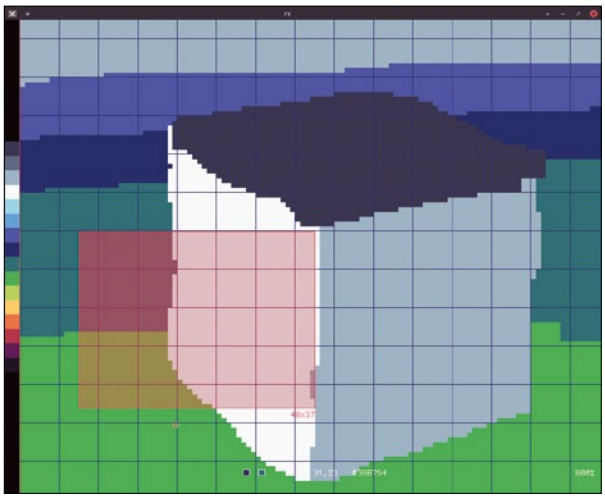
Project Website
https://github.com/MaurycyLiebner/enve

Pixel editor

rx

Even if you're by no means a graphic artist, there is simple pleasure to be found playing with a pixel editor. What could be more satisfying than clicking your pointer to create large chunks of color, usually within a very limited grid of potential locations and an equally limited palette? It's the opposite of having too much choice, and even when you choose not to click *save*, your time is never wasted. Of course, at the other end of the talent scale, if you are capable of putting two decent pixels together, there's never been a better time to be into pixel art. The world of new retro gaming is hugely popular and features modern interpretations of old 8-bit hardware limitations within the confines of modern hardware. It's also surprisingly hard to create pixel art with a fully fledged editor like Gimp, because not only does it offer too much choice, pesky modern conveniences like anti-aliases, feathering, and dithering can often mess up your perfect creations.

For this reason, rx is an excellent choice of pixel editor. Its user-interface takes a lot of inspiration from the Vim text editor, which also means it's not obvious how to quit or do much of anything until you've read the help section. You need to learn just a few keyboard shortcuts, all of which are very similar to Vim's. There are visual (selection), insert (editing) and command (:) modes, and navigation is via *H*, *J*, *K*, and *L*. You can still use the mouse to select colors and to draw on the canvas, and a big feature in rx is animating your creations by pressing Return to create a new frame. Your animation will start to play, allowing you to draw updates in real time, much like animation in Deluxe Paint on the old Amiga. It's a great way to mess around, and if you're an artist, produce brilliant results while writing your game code in Vim!



Rx's command mode lets you do many non-obvious things, like set a grid, manipulate pixels, and add to the palette.

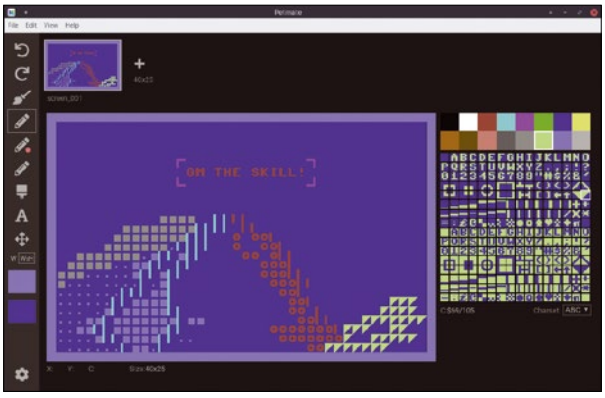
Project Website
https://rx.cloudhead.io/

Character art

Petmate

As brilliant as rx is for editing pixels (see above), if you're looking for the definitive low-fi tool for creating images, you can't beat getting back to the native 1980s experience. On the various Commodore home computers of the early-mid 1980s, the VIC-20, Commodore 64, PET, and humble Plus/4, images weren't always made from pixels directly, but out of text characters. Back in the day, this was both a good way to save RAM and an easy way to create pictures without an image editor. Commodore even printed its esoteric character glyphs on the front of its keyboards, which meant any user at any skill level could get involved. And lots of people did, making the resulting PETSCII images, as they became known, synonymous with homegrown, typically BASIC-written games and demos.

Decades later, getting access to PETSCII characters to be able to create images on modern hardware was challenging. However, a brilliant application called Petmate has finally made it easy. When launched, its color palette and design is utterly reminiscent of the Commodore 64 after a reset, complete with its character set on the right and an empty Commodore screen on the left. You can start drawing by simply selecting a color, then a character, and then by clicking and drawing in the main view. Unlike drawing with a regular brush, these characters work best when they're well placed alongside other specific characters, and this is where PETSCII requires more skill than creating similar images with a normal pixel editor. However, it does mean you will be able to export your images to work on real hardware, and while the default character set is directly from Commodore's ROM, you're free to change it, as well as the palette, to create your own set of new limitations.



Export your Commodore 64 character images as a PRG executable, or assembler code to run on the real hardware.

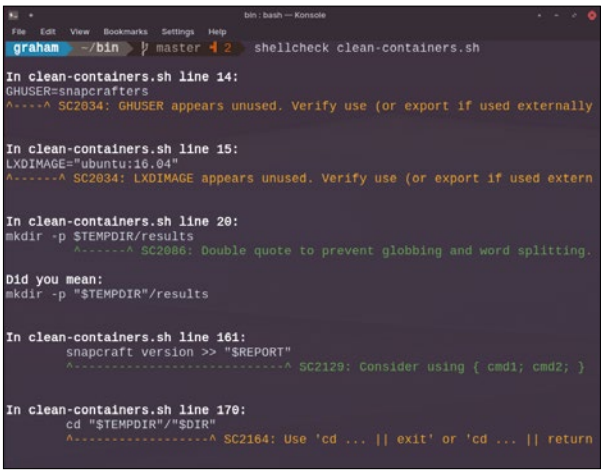
Project Website
https://nurpax.github.io/petmate/

Script analyzer

ShellCheck

One of the best things about the command-line shell is that when you find yourself typing the same commands over and over, or when you know you're going to iterate over a command dozens of times, you can dump the whole thing in a script that can be run again and again. Thanks to the way many shells are structured, you don't need to be much of a programmer to get any of this to work,. You can often paste a few commands into a script without knowing anything other than `#!/bin/bash` and that a return signal of *0* means a command executed correctly, rather than incorrectly. (Don't forget that with `if` statements!) As time goes on, however, scripts will often get augmented with other bits of code, and before long, they've become a vital part of your personal or professional infrastructure. That's when you need to start looking at their contents more seriously.

ShellCheck is a script-checking tool that's intended for beginners but also happens to be useful for solutions that have outgrown their initial single use cases. It's also a great way to improve your script knowledge. It can be run via an online form, and even from within your favorite script editor, but it's often simpler to start with the command itself by passing your script as its only argument. If you get no advice, then ShellCheck has found nothing drastically wrong with your script, but that's unlikely because ShellCheck is very good at identifying errors. It warns you when you don't use a variable, when you should be using double quotes, when to remove dashes, and when to add a return case, alongside dozens of other typing, portability, style, and conditional errors. And unlike some of the scripts we tested, it works perfectly.



If you write any `sh`, `bash`, `dash`, or `ksh` scripts, ShellCheck will offer suggestions on how to improve them.

Project Website

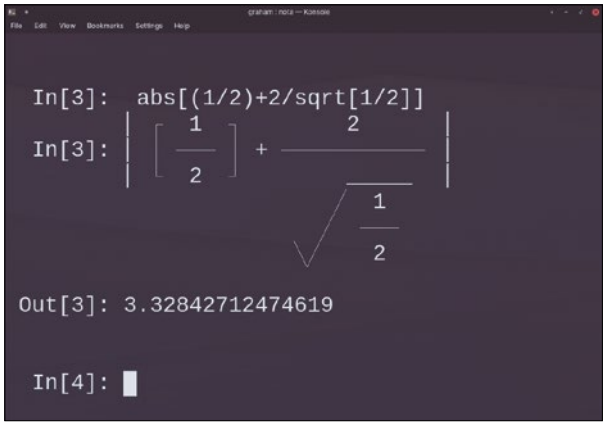
https://github.com/koalaman/shellcheck

Terminal mathematics

Nota

There are lots of commands you can use from the terminal for performing mathematical functions. There's `expr`, for example, which performs addition, division, and subtraction and compares two values, and there's `bc` for more complicated expressions where high precision is important. Many terminal emulators, including Bash, even include their own mathematical interpreters, letting you perform some simple sums without even summoning a command. All of these methods are functional, but they're not terribly exciting or fun to use. The same can't be said for Nota, a terminal calculator with "rich notation rendering." What this means in practice is that you get a character-based graphical representation of your calculation, whether it's a simple sum or a more complex expression.

Unlike many other tools, Nota's calculations are the product of an interactive session, and when you start the tool you're asked to enter the various inputs you wish to process. But the clever part is that it's not expecting just numbers, but commands that use its own simple language. You can create variables, for instance (*a=1*), and use them in later inputs. Inputs can be decimal, hex, binary, named identifiers, and hidden within parentheses. Each input is always evaluated, so you can see what's happening. This process lets you create a function exactly as you might remember it – or code it. The output is always visually appealing, such as putting *1* over *2* when the input is *1/2*, or an ASCII radical sign when evaluating *sqrt[1/2]*. Combine this with the variables, and you can process complex input much like you would on a piece of paper, only with each step being evaluated and with beautifully rendered text output. This is what makes Nota so interesting, and our new default terminal-based mathematical tool.



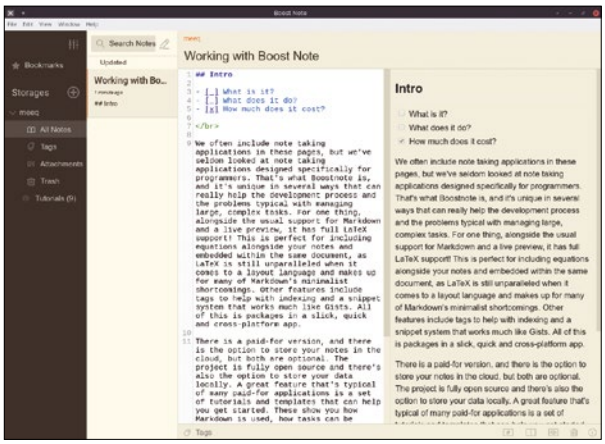
The only additional feature we'd like to see in Nota is the ability to process input from the command line, rather than from its interactive session.

Project Website

http://codes.kary.us/nota/nota

Coder notes

Boost Note



Alongside your Markdown notes, you can also include attachments and thanks to KaTeX, even LaTex-formatted equations.

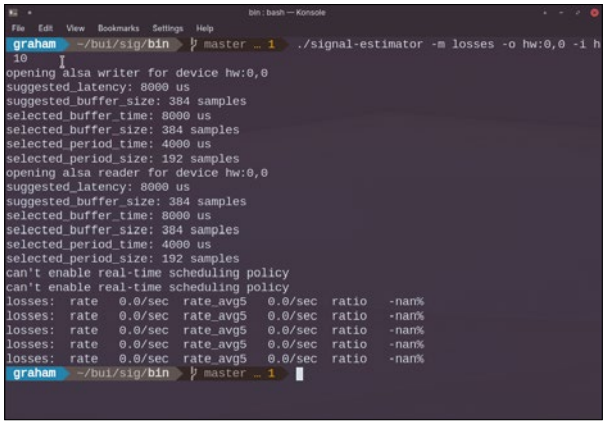
We often include note taking applications in these pages, but we've seldom looked at a note taking application designed specifically for programmers. That's what Boost Note is, and it's unique in several ways that can help improve the development process and address the problems typical with managing large, complex tasks. Alongside the usual support for Markdown and a live preview, it has full LaTeX support. This is perfect for including equations alongside your notes and embedded within the same output document. LaTeX is still unparalleled when it comes to layout and makes up for many of Markdown's minimalist shortcomings, but it's not the only trick, either. Other features include tags to help with indexing, and a snippet system that works much like Gist's. All of this is packaged in a slick, quick, and cross-platform app.

There is a paid version, and there is the option to store your notes in the cloud – which may be a good choice when the accompanying mobile app is released, but both cloud storage and subscriptions are optional. The project is fully open source, and there's also the option to store your data locally. A great feature that's typical of many paid-for applications is a set of tutorials and templates. These show you how Markdown is used, how tasks can be produced and tracked, which keyboard shortcuts to use, and how to use the storage options. The templates include a framework for brainstorming, a bug fix, meeting notes, and a weekly planner; they're a great way to get started. Several light and dark themes are included, and you can edit the syntax highlighting colors for your chosen types of code. You can also export directly as either HTML or as the raw Markdown, so you can always get hold of your data, regardless of where it's stored.

Project Website
https://boostnote.io/

Audio testing

signal-estimator



Check whether your Bluetooth audio is adding more delay than your USB audio.

When you work with audio, whether for a recording or to engage in real-time chat, it's often difficult to answer the simplest questions: How will this sound? Why is there a delay? That's because there are two stages in this process, and it's not always clear what impact one can have on the other. Stage one is the analog to digital conversion (ADC). This is what takes your raw audio and turns it into something your computer can work with. The second stage does the opposite, transforming the digital audio on your computer to a sound you can actually hear – digital to analog conversion (DAC). These two stages are familiar to us all, but they still involve a tricky balancing act, and their performance can be difficult to measure without either special equipment or great effort.

The `signal-estimator` tool aims to make all this easier. It's a simple tool that's been designed to send a signal to an output that's directly connected to an input, a so-called loopback. This gives it complete control and oversight over your audio signal chain, allowing it to measure both outgoing and incoming delays (latency) and quality. Quality is measured with both a glitch detection algorithm and an algorithm that detects signal loss. It also allows you to measure latency on external equipment, which is important when considering the precise timing of a performance. With the audio connection defined from the command, and the mode selected, `signal-estimator` will continuously output either the latency or the signal loss measurement, allowing you to change the physical connection or connect more to the signal chain to see, rather than hear, what the effect might be.

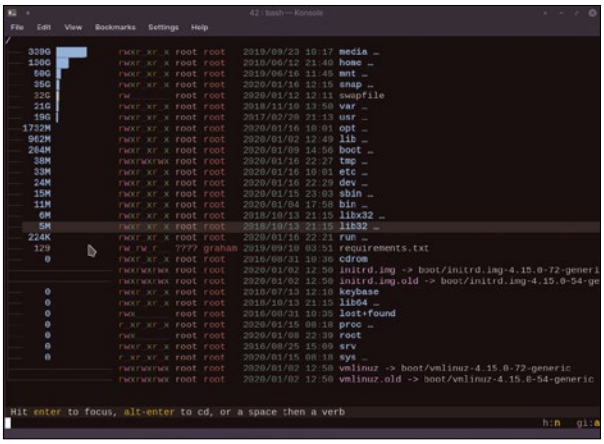
Project Website
https://github.com/gavv/signal-estimator/

File navigation

broot

Despite how complex our filesystems have become, often holding hundreds of thousands of files with dozens of directories across multiple sources and filesystems, our tools have changed very little. We jump around locations with `cd`. We still use `ls` to list the contents of a directory, sometimes carefully sorted with extra arguments or piped into a pager, or `grep` for searching. Failing that, `find` becomes the equivalent of a Google search for things you can't locate, or more commonly, can't remember whether you still have. What we need is a better way of doing these things, built for the age of the terabyte, and that's exactly what broot attempts to be. It's a command-line filesystem navigation tool that's written to be as intuitive as the tools we know while upgrading the experience to deal with the huge sizes and quantities of information we now need to parse when finding files and directories.

After a painless installation process, broot is actually started by typing the much shorter `br` on the command line. This transforms your terminal into a file and directory browser. Unlike an app like Midnight Commander, though, its output feels more like an interactive `tree` and `ls` session than a desktop application. You can move the cursor up and down the list, press Enter to select a folder and Alt+Enter to `cd` into that location. Launch with `br -dp` and you get permissions and dates alongside the file entries.



Use `mv`, `cp`, `rm`, and `mkdir` just as you would on the command line, only without losing the context of the file and directory view.

From within broot itself, you can enter a command mode by pressing `:`. This allows you to enter commands like `s` to show the sizes of folders, or `e` to edit a file. It's very quick and works well and is a great alternative to losing track of exactly where you are on your filesystem.

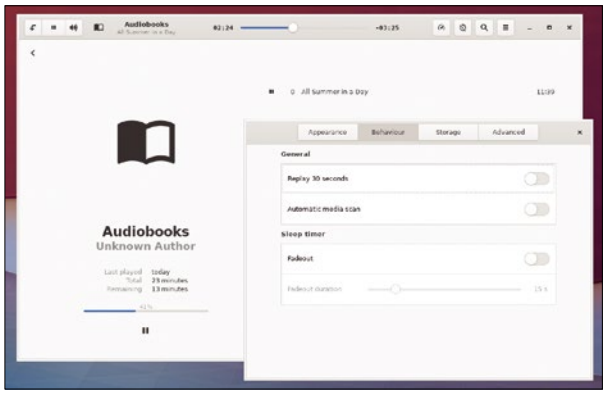
Project Website
https://github.com/Canop/broot#installation

Audio book player

Cozy

Audio books have become hugely popular, and there's no doubt they're a great alternative to reading when you're traveling or otherwise can't read. Unless you're using a proprietary service, you can generally use any audio player to listen to them, and this means audio books have no special requirements – they can be managed and played just like albums and other pieces of music. But they can also benefit from some special considerations, especially when it comes to organizing a library, and extended playback control, such as chapter support. That's what the beautifully minimal Cozy offers – a minimal, distraction free Elementary-inspired GTK+3 environment from which you can select what to listen to and play it back.

Cozy operates a lot like an older version of iTunes, long before it became cluttered and full of distractions. This is a good thing, because the original iTunes design was excellent. A folder can be automatically monitored for audio books, and you can also drag and drop audio files into the main window to add them to your library. From the library, you can then sort by author, reader, and title – not all of which may be available from a free online source. After playback has commenced, you can change the playback speed to between 0.5 and 2.0 times the original, and the quality of the audio processing here is excellent. If you need to rapidly skip through a book, rather than slowly enjoy it, 1.6 speed should be perfectly digestible for most people. On the other end of the scale, if you want to fall asleep listening to a book, there's a timer, and the option to stop playback after the current chapter. It works perfectly and is well worth installing if you prefer your books read to you.



With sleep timers, speed control, and an optional dark theme, Cozy is perfect for listening to books in bed.

Project Website
https://cozy.geigi.de/

Puzzle platformer

VVVVVV

It's not quite open source, but the source code to VVVVVV is now freely available for personal use, and most importantly, freely available to study.

While nearly everything we look at here is, and should be, FOSS (i.e., open source, as defined and hosted by the Open Source Initiative), this game is not. Its source code has been released under a "personal use only" and "does not include media assets" license, which is something we wouldn't normally want to encourage. However, 10 years after its initial proprietary, trailblazing, and award-winning commercial release, VVVVVV is still a brilliant and important game. It's a retro-puzzle-platformer heavily inspired by the impossible platformers of the 1980s. The six Vs in its title represent the initials of six scientists who are missing within the vaults of a huge cave system. You play as one of these Vs, Viridian, so you need to track down the other five. You move left and right and jump to get across the 2D screens, and jumping can also switch you to the ceiling or the reverse side of platforms to help you navigate the 400 rooms and increasingly complex levels. You die a lot, but there are often multiple respawn points on each level.

Its visuals look like something you might have played on a 48KB Spectrum, while its audio is just about on-par with an Atari 800XL. Each screen has a name, reminiscent of Jet Set Willy, and the whole effect is fantastic. While there's obviously a strong appeal to nostalgia, the game itself works as a modern puzzler, very much in keeping with the indie game development scene, where you die often and make progress slowly, all the while getting more hooked on the gameplay. And that's what makes the release of its source code important, even if it's not open source. Game development is hard, and finding gameplay examples from successful releases that you can study is almost impossible – or was, until the release of VVVVVV.

Project Website
https://github.com/TerryCavanagh/VVVVVV

First person shooter

Red Eclipse 2

Thanks to its slow regeneration times and ammo collection, Red Eclipse forces a more strategic gameplay.

Red Eclipse 2 is a major update to the first Red Eclipse, an open source first-person shooter that's been around for over a decade. The original game was focused on online multiplayer mode, with addictive, fast gameplay that felt very much like the best bits of Unreal, especially when you played it competitively with your friends. It's a huge achievement that a game like this can survive so long, and more importantly, that a game like this has enough support to keep growing and developing. This new version includes a new UI and heads-up display. The graphics engine has also been re-engineered to include real-time dynamic lighting and shadow, ambient occlusion, volumetric lighting, global illumination, HDR, bloom, and tone mapping, as well as water and glass reflections and refraction. The end result is a fantastic looking game that looks much more modern than its previous 1.6 release. There's also a much more dynamic movement system that even incorporates a "parkour" type of mechanic alongside impulse boosts and dashing. You can practice your new-found parkour skills in a new single player training mode that challenges you to use different moves. This and the practice levels with bots are the only single-player experiences in the game.

Online gameplay has been refocused and intensified by only offering a limited set of maps, with seven main deathmatch maps and one race map included by default. But there's also a tutorial level for beginners. The game's permissive licensing means you can now download the game for free from Steam, too. This allows you to take advantage of Steam's multiplayer functionality, as well as the convenience of the game being listed and updated alongside any other games you may own. But you can also download, or even build, the game manually and run it without any other configuration.

Project Website
https://www.redeclipse.net/

Creating a simple laptop notification

The Linux Box of Tricks

With Bash, keeping a task simple often means using several different tools that all do their jobs well. Here's an easy, effective way to create a notification for when your laptop is unplugged.

BY PAUL BROWN

If you trawl forums where newbies hang out (like I do), you'll often come across people yearning to learn Linux and asking things like: "Where can I learn about [insert some technology here]?" "Some technology" being anything ranging from the very basic stuff, like "the shell" or "the filesystem," to the quite advanced, like "systemd" or "Docker."

That, I think, is the wrong question to ask and, hence, tutorials often get the answer wrong and center on one tool and one tool alone. This forces the creator of the tutorial to jump through hoops to be able to do everything they have set out to explain with that single tool. It can be taken to the extreme with the author writing about how you can use LibreOffice Writer to open a socket to a remote host – probably not the best way to get data from the network.

A better question may be what are the best tools to carry out this specific task? The answer is usually whatever is easiest and most effective, regardless of whether you have to use four or 40 different tools to achieve your aim. And even with an easy task, you often have to resort to several things in the Linux box of tricks to make sure the project is effectively tackled.

In this article, let's look at how to make a warning system that is as simple as possible. We will make a notification pop up from your tray when your laptop is unplugged from the electrical outlet. That's it. Sounds easy, and it kind of is – except that, to avoid over-complicating things or getting into serious programming, you need to use at least four different tools.

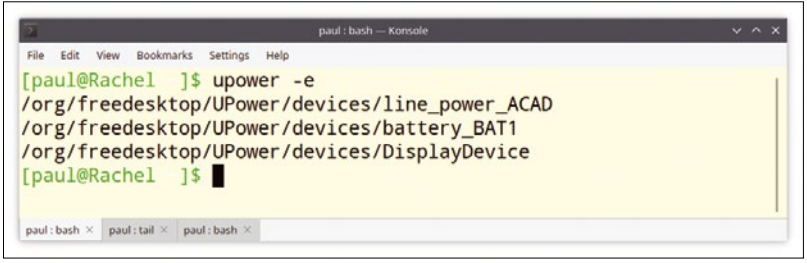


Figure 1: UPower tells you, among other things, what power sources your device counts on.

The rationale for this utility could be that, while desktops often do show that the laptop is unplugged by popping a battery icon into the tray, this is often easy to miss. If it plays a sound, it makes the warning easier to catch, but if you are listening to music, have the sound down, or are hard of hearing, you may miss that, too. If you have the icon, the sound, *and* a pop-up notification, it is your own silly fault if you inadvertently drain your battery.

UPower

The first tool you're going to have to get familiar with is UPower [1]. UPower is a utility developed by freedesktop.org and is designed to, among other things, give you information about what is happening with the power sources on your machine.

Apart from an API to reach this information from several programming languages, UPower also comes with a handy human interface, ideal for quick and dirty scripting.

Try this, for example:

```
upower -e
```

This is a list of power sources available to your laptop. On your particular machine, they may have slightly different names from what you see in Figure 1, but you can be pretty sure that the entry for the battery will have the word "battery" in it and the entry for the electricity coming from the power outlet will have the words "line" and "power" in it.

But UPower's `-e` option doesn't tell you to which power source your machine is connected. For that you use the `-i` option. Following from the example shown in Figure 1:

```
upower -i /org/freedesktop/
  UPower/devices/line_power_ACAD
```

will show whether the electrical power source is connected. Figure 2 shows that it is – check the line that says `online: yes`.

Likewise,

```
upower -i /org/freedesktop/
  UPower/devices/battery_BAT1
```

will show you information about your battery, including the make, model, serial number, level of charge, and all sorts of technical information. (Remember that the exact name of the devices you want to query will probably be slightly different on your machine – again check `upower -e` to see the exact names for your computer).

The third thing you'll find useful for this project is UPower's `-m` option. This monitors changes in the state of the power supplies.

Run

```
upower -m
```

and then disconnect your laptop from the outlet. UPower will print information about the changes (in this case, that it is switching from line power to battery) to the terminal.

Now you know what power sources are available (`upower -e`), you can find out whether they are connected or disconnected (`upower -i source`). When that status changes (`upower -m`), you have the foundations of your project.

systemd Units

The next thing you need are some systemd units. Units are systemd's equivalent to the old init scripts: You describe a series of rules in them and systemd makes them happen. The rules lay out what state you want the system to be in and when. For example, a unit may tell systemd that you want networking at a certain point in the boot process, or that systemd should mount external USB storage devices when they are connected, or hundreds of other things.

Systemd provides several kinds of units [2]. Some run services, others set tasks to run at certain times or on certain days or mount devices, and there are some that monitor files and directories. The latter should prove useful, since in today's task you want systemd to monitor the changes in the power source.

But first you need something to monitor. As mentioned above `upower -m` pushes out changes to the power sources to the terminal and doesn't stop until you hit Ctrl+C. You can use this to pipe the output to a file using something like

```
upower -m > upower.log
```

However, if you run this from the command line, it will tie up your terminal. The best way to run this is as a systemd service. As systemd units cannot process Bash commands chained together, you have to wrap the command in a script (Listing 1).

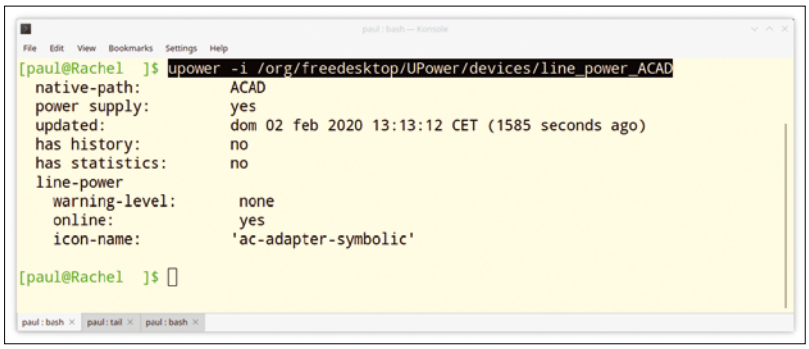


Figure 2: You can use UPower to check the state of the power sources.

Save that as `upowerlog.sh` to a directory on your path (I saved it to `/home/paul/.local/bin/`), make it executable with

```
chmod a+x /home/[username]/
  .local/bin/upowerlog.sh
```

and you can then run it from a service unit like the one you see in Listing 2.

In case you are not familiar with systemd units, they are split into sections. The `[Unit]` section often contains information about the unit for humans to read and directives that tell systemd how the unit is related to other services in the system. If this service needs another service to run for it to work correctly, for example, you would put that here; if the service needs a task to run before it starts, you would also put that here; and so on.

In this case, all you have are the optional `Description` and `Documentation` directives. The former tells a human user what the service does, and the latter contains a link to where you can read up more about the unit itself.

The `[Service]` section contains information that tells systemd what to do. In `upowerlog.service`, the service is a `simple` service, which means it just needs to start and runs until it is told to stop or the

Listing 1: upowerlog.sh

```
#!/bin/bash

upower -m > /home/[username]/.local/share/upower.log
```

Listing 2: upowerlog.service

```
01 [Unit]
02 Description= logging changes in power supply to laptop
03 Documentation= http://www.linux-magazine.com/Issues/2020/233
04
05 [Service]
06 Type= simple
07 ExecStart= /home/[username]/.local/bin/upowerlog.sh
```

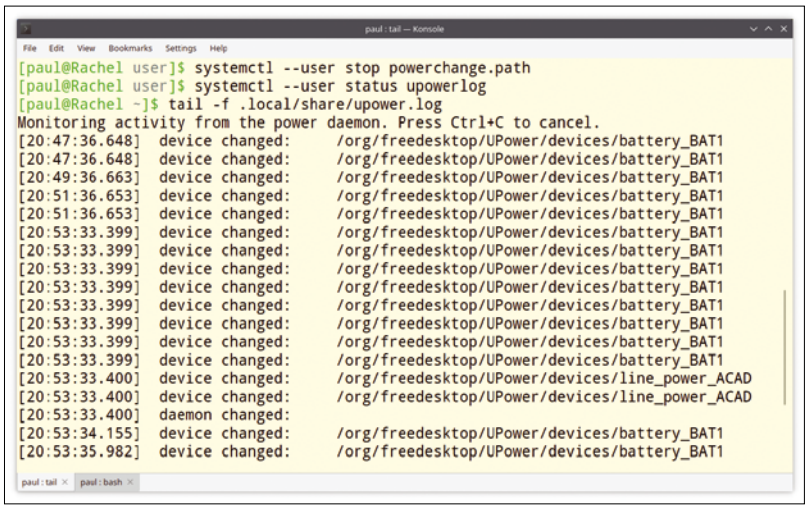


Figure 3: Tracking changes in power source as a service.

system powers down, while the `ExecStart` directive tells systemd what it has to run, in this case, the `upowerlog.sh` script.

You don't have to go to the hassle of becoming root and saving your units in `/etc/systemd/system/`, which is where they usually live. Non-root users can have their own services, too!

Save the contents of Listing 2 as `upowerlog.service` in `/home/[username]/.config/systemd/user/`. You can then start it with:

```
systemctl --user start upowerlog
```

and you'll see a file called `upower.log` pop up in `/home/[username]/.local/share/`.

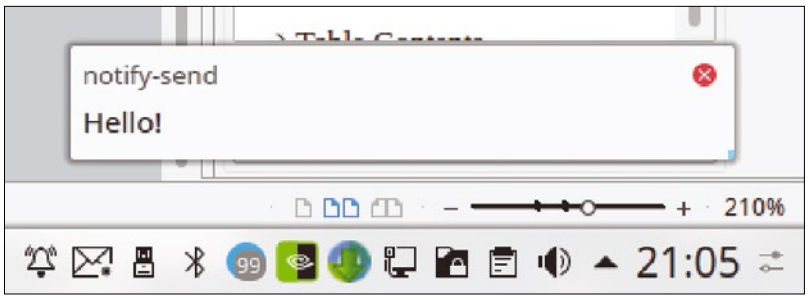


Figure 4: You can send custom notifications to your desktop using `notify-send` from the shell.

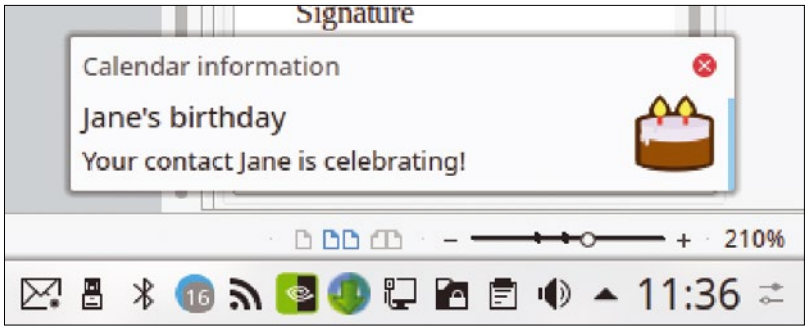


Figure 5: You can make your notifications more visually attractive with icons and custom titles.

You can then see its contents change with

```
tail -f /home/[username]/➥
  .local/share/upower.log
```

Plug in and then unplug your laptop several times and you'll see something like what is shown in Figure 3.

Stop the service with

```
systemctl --user stop upowerlog
```

while you build the monitoring part of your service.

Notifications

Sending a notification to a standardized Linux desktop is very easy:

```
notify-send 'Hello!'
```

This will show a notification with the message "Hello!" popping up from your tray (Figure 4).

You can, of course, spruce the notification up a bit. The `-a` option, for example, allows you to change the title of the pop-up box from the default `notify-send` to whatever you want, and the `-i` flag lets you set an icon. If you choose an icon from `/usr/share/icons`, you don't need to specify the path. You can also leave out the extension. Another thing you can do is specify how long the notification will be shown with the `-t` option. The duration is specified in milliseconds.

The following line

```
name="Jane"; notify-send -a ➥
  "Calendar information" -i food-cake ➥
  -t 5000 "$name's birthday" ➥
  "Your contact $name is celebrating!"
```

will show what you can see in Figure 5.

To show a notification when the computer is disconnected from the electrical outlet, you can do the following.

Use `upower -e` to find the name of the line power source:

```
upower -e | grep line
```

Use that to get information about it:

```
upower -i $(upower -e | grep line)
```

Grab the line that tells you if it is online:

```
upower -i $(upower -e | ➥
  grep line) | grep online
```

Extract the "yes" or "no" and remove excessive spaces:

Listing 3: powerchange.sh

```
01 #!/bin/bash
02
03 if [ $(upower -i $(upower -e | grep line) | grep online | cut -d':' -f 2 | tr -d ' ') = 'no' ]
04 then
05   notify-send -a "Power source information" -i dialog-warning -t 5000 "Laptop unplugged"
       "Check you haven't kicked the charger by accident."
06 fi
```

```
upower -i $(upower -e ➥
| grep line) | grep online
| cut -d':' -f 2 | tr -d ' '
```

And fold the whole lot into a script like the one shown in Listing 3.

Save the script as `powerchange.sh` in `/home/[username]/.local/bin/` and test it with your laptop both plugged in and unplugged. When you run `powerchange.sh` with your laptop unplugged, you will see a notification like the one shown in Figure 6.

Path

The final step is to set up the systemd units that will monitor and trigger the execution of `powerchange.sh`.

You use `path` units to monitor files and directories. As you will be using `upowerlog.sh` to update the `upower.log` file, `upower.log` is what you want to monitor. The `path` unit shown in Listing 4 does that.

We already talked about the `Description` directive when explaining `upowerlog.service` and the `PathModified` directive is self-explanatory: It points to a file or a directory and is triggered if said file or directory is modified.

There is another new directive in `powerchange.path` worth looking at in a little more depth, though. Line 3 shows a `Wants` directive that points to the first unit you created, `upowerlog.service`. This ensures that if `upowerlog.service` is not started when you run `powerchange.path`, systemd will get it started. This means you don't have to run `upowerlog.service`

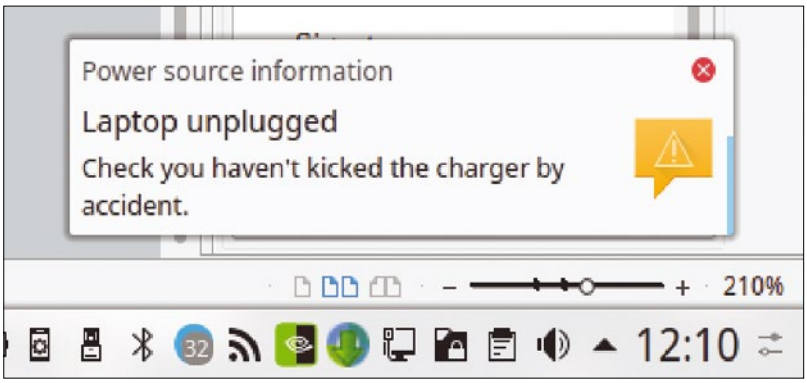


Figure 6: `powerchange.sh` shows a notification when the laptop is unplugged.

yourself; systemd will do it for you the moment it parses the `Wants= upowerlog.service` line.

You will notice that there is no call to `powerchange.sh`, the script we created in Listing 3. That's because path units do not execute scripts themselves. Instead they rely on a companion `service` script (usually with the same name, but with the `.service` extension) to do that for them. After all, that is what `service` units are for: starting scripts or programs.

`powerchange.service` looks like what is shown in Listing 5. Again, this service is self-explanatory and all it does is run the `powerchange.sh` script from Listing 3.

Listing 4: powerchange.path

```
01 [Unit]
02 Description= monitoring upower log
03 Wants= upowerlog.service
04
05 [Path]
06 PathModified= /home/[username]/.local/share/upower.log
```

Listing 5: powerchange.service

```
01 [Service]
02 Type= simple
03 ExecStart= /home/[username]/.local/bin/powerchange.sh
```

Save both `powerchange.path` and `powerchange.service` to `/home/[username]/.config/systemd/user/`, and you can now run `powerchange.path` with

```
systemctl --user start powerchange.path
```

and check that everything works by unplugging your laptop. The notification shown in Figure 6 should pop up, warning you that your machine has been disconnected.

Walkthrough

To summarize what's going on:

1. When you start `powerchange.path`, systemd also starts `upowerlog.service` thanks to the `Wants` directive in `powerchange.path`.
2. `upowerlog.service` runs `upowerlog.sh`, which starts dumping the output of `upower -m` into `/home/[username]/.local/share/upower.log`.
3. Each time `/home/[username]/.local/share/upower.log` changes, `powerchange.path` calls `powerchange.service`.
4. `powerchange.service` runs the `powerchange.sh` script that checks to see if the line power source is connected. If it is not, it sends a notification to the desktop.

Conclusions

I hope this has helped illustrate how you can leverage a wide variety of seemingly unrelated tools to create one single project.

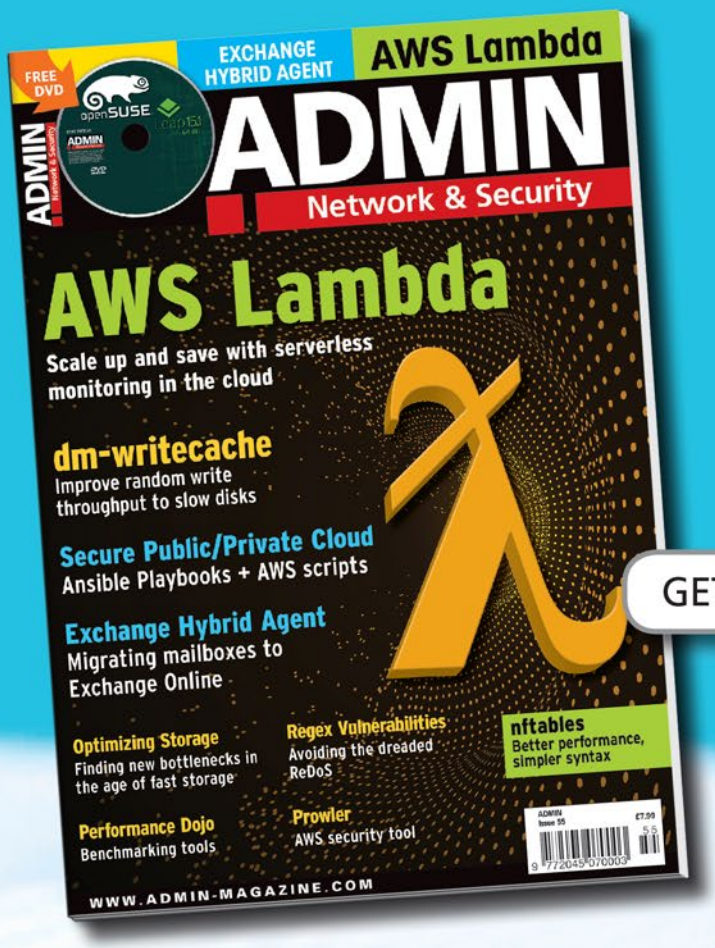
In any case, systemd units, the UPower system, `notify-send` (and other command-line-to-desktop utilities), and Bash's daisy-chaining of terminal commands are all worth exploring further. They provide you with a rich set of tools you can use to complete all kinds of projects. ■■■

Info

[1] UPower: *https://upower.freedesktop.org/*

[2] systemd units: *https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files*

REAL SOLUTIONS *for* REAL NETWORKS



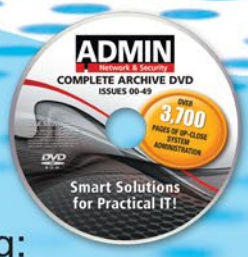
THE NEW IT

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

SUBSCRIBE NOW!

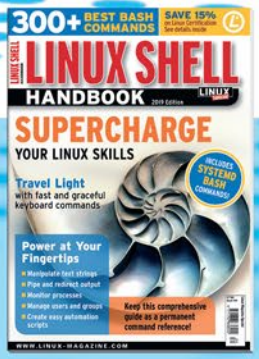
shop.linuxnewmedia.com

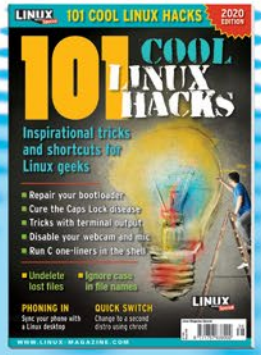
GET IT FAST with a digital subscription!



Check out our full catalog:

shop.linuxnewmedia.com





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*.

KubeCon + CloudNativeCon Euope 2020

Date: March 30-April 2, 2020

Location: Amsterdam, Netherlands

Website: *https://events.linuxfoundation.org/kubecon-cloudnativecon-europe/*

The Cloud Native Computing Foundation's flagship conference gathers leading technologists from leading open source and cloud native communities to further the education and advancement of cloud native computing.

DevOpsCon

Date: April 21-24, 2020

Location: London, United Kingdom

Website: *https://devops.jaxlondon.com/*

JAX DevOps is a four-day conference for software experts featuring in-depth knowledge of the latest technologies and methodologies for lean businesses. Join the software delivery revolution for accelerated delivery cycles, faster changes in functionality, and increased quality in delivery.

LinuxFest Northwest

Date: April 24-26, 2020

Location: Bellingham, Washington

Website: *https://linuxfestnorthwest.org/conferences/2020*

LinuxFest Northwest features presentations and exhibits on free and open source topics, as well as Linux distributions & applications, InfoSec, and privacy; something for everyone from the novice to the professional! This annual Open Source event co-produced by Bellingham Linux Users Group and the Information Technology department at Bellingham Technical College.

Events

CloudFest 2020	March 14-19	Europa-Park, Germany	https://www.cloudfest.com/
SUSECON 2020	March 23-27	Dublin, Ireland	https://www.susecon.com/
KubeCon + CloudNativeCon Europe 2020	March 30-April 2	Amsterdam, Netherlands	https://events.linuxfoundation.org/kubecon-cloudnativecon-europe/
Open Networking & Edge Summit North America	April 20-21	Los Angeles, California	https://events.linuxfoundation.org/open-networking-edge-summit-north-america/
Strata Data Conference	April 20-23	London, United Kingdom	https://conferences.oreilly.com/strata-data-ai/stai-eu
DevOpsCon	April 21-24	London, United Kingdom	https://devops.jaxlondon.com/
LinuxFest Northwest	April 24-26	Bellingham, Washington	https://linuxfestnorthwest.org/conferences/2020
Linux Storage, Filesystem and Memory Management Summit	April 27-29	Palm Springs, California	https://events.linuxfoundation.org/lsfmm/
IcingaConf 2020	May 12-14	Amsterdam, Netherlands	https://icingaconf.com/
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/
Open Souce Camp on Bareos	June 19	Berlin, Germany	https://opensourcecamp.de/
ISC High Performance 2020	June 21-25	Frankfurt, Germany	http://ct.isc-events.com/click.php?id=138
Embedded Linux Conference North America	June 22-24	Austin, Texas	https://events.linuxfoundation.org/embedded-linux-conference-north-america/

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to *edit@linux-magazine.com*.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at:
http://www.linux-magazine.com/contact/write_for_us.

NOW PRINTED ON recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Authors

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
© Sergey Nivens, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 3090 5128

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
2721 W 6th St, Ste D
Lawrence, KS 66049 USA

Publisher
Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

Printed in Nuremberg, Germany by hofmann infocom GmbH on recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 2721 W 6th St, Ste D, Lawrence, KS, 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 2721 W 6th St, Ste D, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany.

Approximate

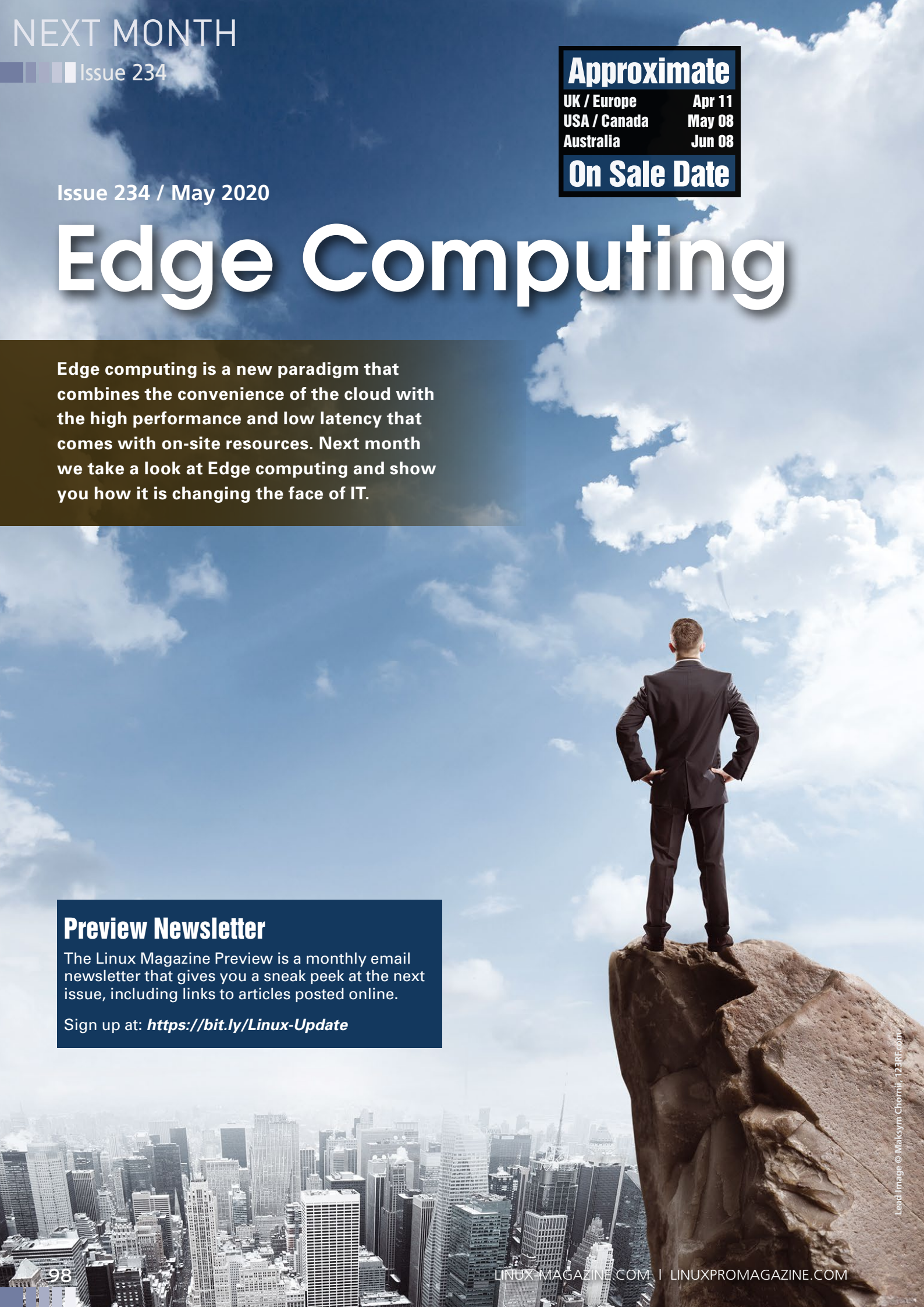
UK / Europe	Apr 11
USA / Canada	May 08
Australia	Jun 08

On Sale Date

Issue 234 / May 2020

Edge Computing

Edge computing is a new paradigm that combines the convenience of the cloud with the high performance and low latency that comes with on-site resources. Next month we take a look at Edge computing and show you how it is changing the face of IT.

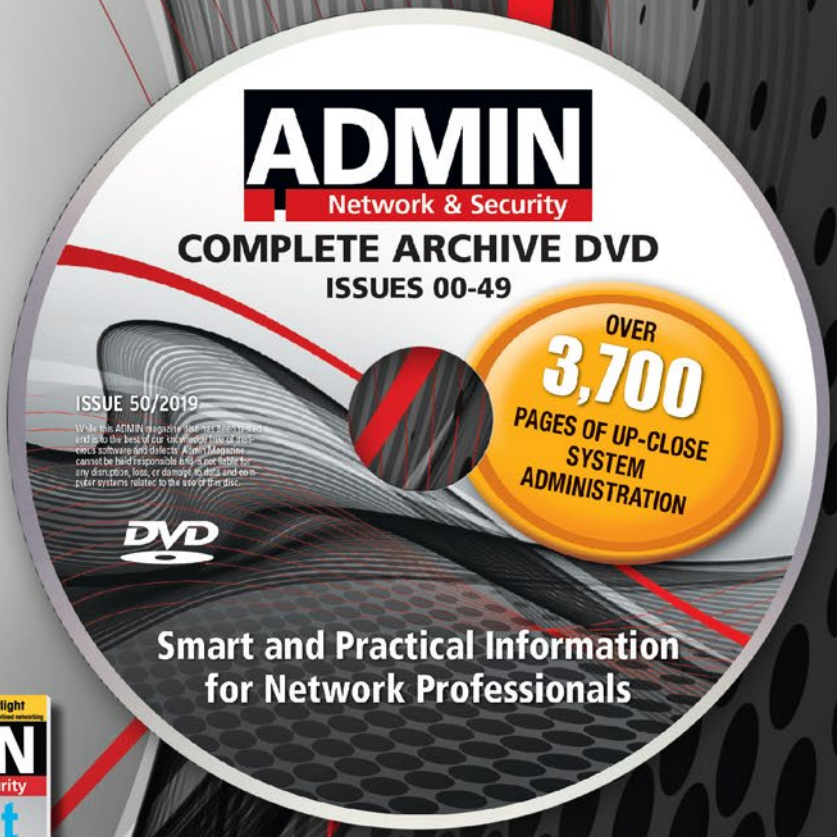


Preview Newsletter

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***

9 Years of ADMIN on One DVD



This searchable DVD gives you 50 issues of ADMIN, the #1 source for:

- network security
- system management
- troubleshooting
- performance tuning
- virtualization
- cloud computing

Clear off your bookshelf and complete your ADMIN library with this powerful DVD!

ORDER NOW!

shop.linuxnewmedia.com

HETZNER

EXCELLENT WEBHOSTING

FOR BEGINNERS & EXPERTS

READERS' CHOICE
HOSTING & SERVICE PROVIDER
AWARDS 2019 PLATIN

INCLUDES DOMAIN AND WORDPRESS INSTALLER

IDEAL FOR SIMPLE WORDPRESS SITES

e.g. Web Hosting Level 1

- ✔ 1 domain included
- ✔ Unlimited traffic
- ✔ 10 GB disk space
- ✔ 50 sub-domains
- ✔ 100 email accounts
- ✔ 1 database
- ✔ Daily backup of user accounts
- ✔ FTP and FTPS access
- ✔ Free Symantec Basic SSL Certificate
- ✔ No minimum contract
- ✔ Setup fee $9.20

monthly $ **1.80**

e.g. Web Hosting Level 19

- ✔ 1 domain included
- ✔ 20 Add-on domains (Registration fee-based)
- ✔ Unlimited traffic
- ✔ 300 GB disk space
- ✔ Unlimited sub-domains
- ✔ Unlimited email accounts
- ✔ Unlimited databases
- ✔ Daily backup of user accounts
- ✔ FTP and FTPS access
- ✔ Free Symantec Basic SSL Certificate
- ✔ No minimum contract
- ✔ No Setup fee

monthly $ **18.50**

IDEAL FOR RESOURCE HEAVY WEBSITES

www.hetzner.com