TILING WINDOW MANAGERS
THIS CLASSIC TOOL IS ATTRACTING A NEW GENERATION OF FANS

FREE DVD

LINUX PRO MAGAZINE

ISSUE 259 – JUNE 2022

Zero Trust

Reimagining security for today's threats

Analytics Workshop: Data mining with Python and KDD

Wordle with Grep

BlueSeer: ERP solution for the Linux desktop

OpenSnitch: Protect your network with this application-based firewall

10 TERRIFIC OPEN SOURCE TOOLS!

LINUX NEW MEDIA
The Pulse of Open Source

Affordable Business Allrounder

TUXEDO Aura 15 - Gen2

AMD Ryzen 7 5700U
8 Cores | 16 Threads

USB-C 3.2 Gen2
DisplayPort 1.4 & Power Delivery

1,99 cm | 1,65 kg
Thin & Lightweight

4G / LTE
Mobile high-speed web access

100% Linux

5 Year Warranty

Lifetime Support

Built in Germany

German Privacy

Local Support

TUXEDO COMPUTERS 18th ANNIVERSARY

tuxedocomputers.com

OFFICE IN THE SKY

Dear Reader,

Is Microsoft up to its old tricks again?

The past few years have seen some encouraging signs from Microsoft. For instance, they no longer propagate transparent and self-serving falsehoods about Linux at every opportunity. However, recent events indicate that maybe the new Microsoft might be all too much like the old Microsoft.

EU investigators are currently asking around about Microsoft's cloud contracts [1]: in particular, a change to the licensing that occurred back in 2019. The change relates to how the company bills for products like its office productivity suite, which is now bundled as part of the Microsoft 365 package. As far as I understand it, companies pay for a license to use the software on their network, but if they move some of these instances to the cloud, there is an extra charge. In theory, this extra charge applies to all cloud platforms, including Microsoft's own Azure cloud, but in practice, Azure customers are allegedly getting a special discount that offsets much of this extra fee.

If that sounds sneaky, it is. In fact, the regulators are wondering if it is the kind of sneaky that reaches the level of monopolistic practices. In its most extreme form, a monopoly works by confining the customer to a single choice. Another form of monopoly, which is slightly less exclusionary but still very powerful, operates by imposing a penalty on the customer who chooses to stray from monopoly control. Then there is a third kind of monopoly that isn't really any kind of monopoly at all – I'm not sure what to call it, except perhaps an *imaginary monopoly*. In this scenario, a vendor asserts control by projecting a false reality that makes customers *believe* they have no choice even when they actually do.

Microsoft 365 is somewhere between a real monopoly and an imaginary monopoly. Yes, Microsoft does seem to be using its position in the office software market to upload office suite customers to the Microsoft cloud. But why are customers so willing to go along with it? If they want to switch to a different cloud and Microsoft 365 is holding them back, why don't they just give up Microsoft 365?

Companies that worry about acclimating their users to a new productivity suite should stop worrying – seriously, is it really so difficult to use a different word processor or spreadsheet once you have learned to use one. Perhaps more of a problem are the macros written for Microsoft Office that will have to be rewritten for a different API. First of all, this situation doesn't apply to all customers, so any company that is intimidated about switching for macro compatibility reasons should make an honest assessment about how much their company actually depends on Office macros. Secondly, even if your company does use a lot of legacy Office macros, it is worth considering whether this might be a good time to bite the bullet and replace them with something more portable, now that they are being used as a tether to limit customer choice.

Google Docs certainly bills itself as a full replacement for Microsoft's productivity tools. Amazon provides the infrastructure necessary for a collaboration environment, although their WorkDocs product is currently a bit too focused on supporting Microsoft Office. But Amazon certainly has the resources to implement their own complete solution, and you can bet they are working on it now, given the uncertainty with Microsoft.

You might be wondering why hyper-cloud users don't just switch to LibreOffice. The free LibreOffice suite was instrumental in breaking the Microsoft Office monopoly on the desktop. Couldn't they do the same in the cloud? Someday, perhaps, but there is work to be done.

It appears that The Document Foundation (TDF), maintainer of LibreOffice, isn't really interested in building a universal cloud-based solution. TDF does provide a browser-based, network-ready version of LibreOffice, but they consciously avoid the complexity of integrating cloud storage, authentication, and other technologies needed for a drop-in cloud implementation. A message on the TDF website states, "The Document Foundation is not planning to develop and fund a cloud solution similar to existing products from Google and Microsoft, because this would require selection and integration of the other technologies needed for deployment. This would be a significant growth of scope and not in line with the original mission of the project. The task is therefore left to large deployers, ISPs and providers of open source cloud solutions, and several options are already available on the market." [2]

As the note states, online services based on LibreOffice do exist, but these solutions tend to be from single vendors who are looking to build their own businesses and are not acting on behalf of the whole community. TDF adds, "TDF would welcome provision of a public LibreOffice Online offering by another charity."

Microsoft's recent return to antitrust tactics could be a wake up call for TDF or "another charity" to get working on a free and universal cloud-based productivity solution that will challenge Microsoft's imaginary monopoly in office software. ■■■

Joe Casad,
Editor in Chief

Info

[1] "Microsoft's Tactics to Win Cloud Battle Lead to Antitrust Scrutiny" by Richard Waters, *Financial Times*, 4/13/2022: *https://arstechnica.com/tech-policy/2022/04/microsofts-tactics-to-win-cloud-battle-lead-to-new-antitrust-scrutiny/*

[2] LibreOffice Online: *https://www.libreoffice.org/download/libreoffice-online/*

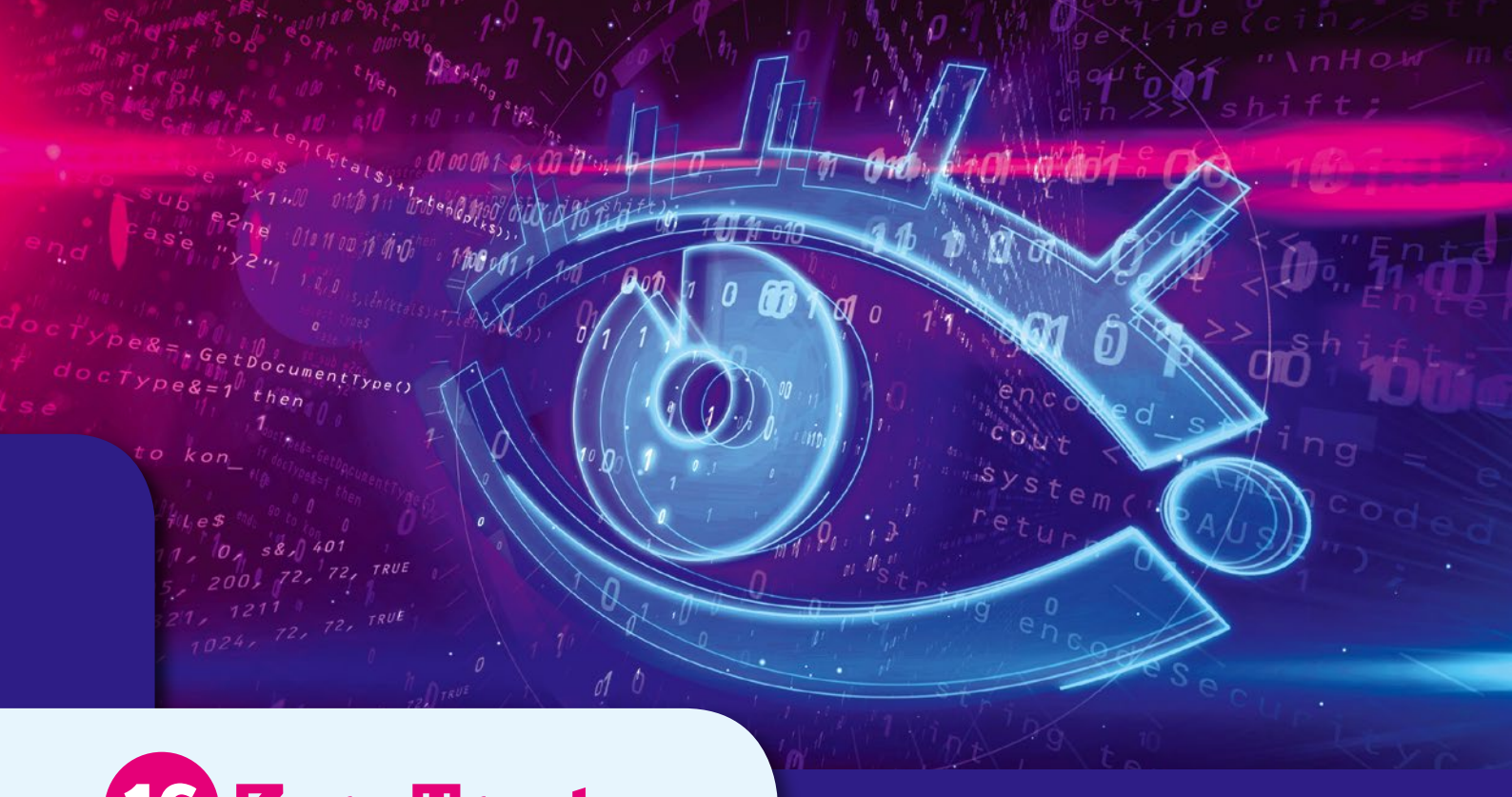
LINUX MAGAZINE
JUNE 2022

NEWS

REVIEWS

COVER STORY

IN-DEPTH

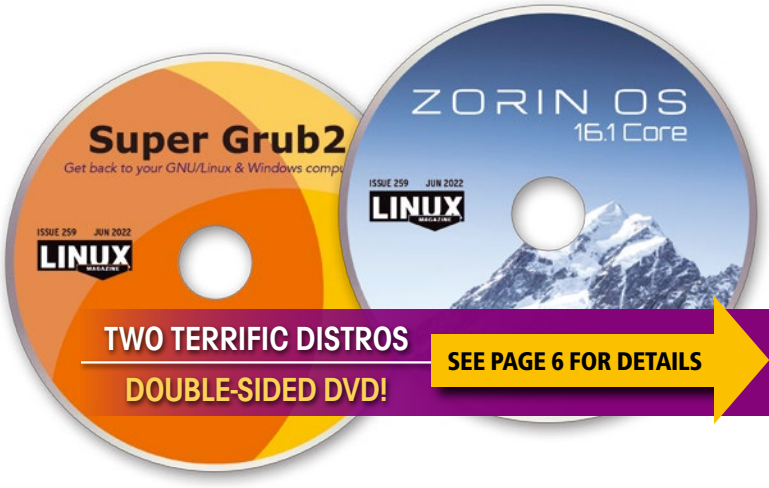


16 Zero Trust

Twenty Years ago, everyone thought a gateway firewall was all you needed to stay safe from intruders, but recent history has told a different story. Today, the best advice is: Don't trust anyone. Your internal network could be just as dangerous as the Internet.

LINUXVOICE

MakerSpace



Zorin OS 16.1 Core and Super GRUB2 Disk

Two Terrific Distros on a Double-Sided DVD!

Zorin OS 16.1 Core
64-bit

Based in Ireland, Zorin OS is a commercial Ubuntu derivative founded by two brothers, Artyom and Kyrill Zorin. In its 13 years of existence, Zorin OS has gained admirers for its minimalist aesthetics. It compares favorably to distros such as Deepin and elementary OS. In particular, Zorin OS is noted for the Zorin Appearance app, a collection of skins that imitate the appearance of other operating systems and distributions.

This month's DVD incudes Zorin OS 16.1 Core, a free download version of the distribution. Its version of Zorin Appearance includes skins for Windows, Windows List, the mobile-device-like Touch, and Gnome Shell, as well as the default modified Xfce desktop. Zorin Appearance aims to ease the transition to Linux by providing a familiar-looking desktop. This intent is popular enough for Zorin to place consistently in the top 10 distributions for page views on DistroWatch.

If you like Zorin OS, you might consider Zorin Lite, another free download for older computers. Alternatively, you might want to consider paying for Zorin Pro, whose Zorin Appearance also includes macOS, Windows 11, Windows Classic, and Ubuntu (meaning Unity). In any version, Zorin makes life simpler for new Linux users and will be appreciated by those who prefer a beautiful and well-designed desktop.

Super GRUB2 Disk
32- or 64-bit

As you probably know, GRUB2 is the most common bootloader for Linux. By editing GRUB2, you can start a system in a different operating system or using a different kernel. By using Super GRUB2 Disk from a live external drive, you can dual boot different Linux distributions, BSD, macOS, and Windows on common form factors.

With Super GRUB2 Disk, you can edit GRUB in a number of ways. The most common uses include listing available bootable drives and adding or deleting a Linux distribution or an operating system. You can also use Super GRUB2 Disk to detect or recover GRUB2 configuration files, as well as enable LVM, PATA, and RAID support. In addition, experimental support is available for booting an external drive. All these options and more are documented on the Super GRUB2 Disk website (*https://www.supergrubdisk.org/wiki/SuperGRUB2Disk*).

You should keep Super GRUB2 Disk on hand in case of emergencies. However, you may also find it the most convenient way to edit GRUB2 under any circumstances.

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.

vendor neutral · **global community** · non-profit · increased salaries trusted in more than 180 countries · professional certification detailed exam objectives · online testing · free learning materials individual skills credentials · **multiple languages** · high availability certifications valid for 5 years · Linux · open technologies · FOSS our mission is to promote the use of open source by supporting the people who work with it · demanded IT skills liberating people · Open Source · 200,000+ certification holders proven and reliable · personal and economic growth opportunity DevOps · economic and creative opportunities for everybody security · **accessible exam prices** · BSD · booming job market distribution neutral · increase your bonus pay · cybersecurity international standard · future proof career · **hundreds of partners** plenty of career paths · need for developers · virtualization open source hiring will rise · recommended for professionals improved workplace productivity · covers all major distributions **become more attractive to employers** · ramp up your career member based organization · elected board of directors prove your skills · higher earning potential · **your future is open**

Better career options. It's what we do.

Linux Professional Institute's mission is to promote the use of open source by supporting the people who work with it. That's why we offer exams in multiple languages and in more than 180 countries. Read what drives us at lpi.org/why.

Find out about the value of Linux Professional Institute certification at lpi.org/value-of-certification/new-to-linux.
More information and all LPI certifications on lpi.org

Linux Professional Institute

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

Microsoft Expands Their Windows Subsystem for Linux Offerings with AlmaLinux

It wasn't an April Fool's prank when Microsoft announced the addition of AlmaLinux (*https://almalinux.org/*) to the line of distributions available for WSL. From the Microsoft Store, it's now possible to download a version of AlmaLinux to run on Windows.

AlmaLinux now joins the growing list of Linux distributions for WSL that includes Ubuntu, OpenSuse Leap, Kali Linux, Debian, Oracle Linux, and Suse Linux Enterprise. This server-centric Linux distribution came into being soon after it was announced that CentOS would be migrating to CentOS Stream back in 2020.

A quick search in the Microsoft Store makes it easy to download and install AlmaLinux 8 for WSL. Just make sure your host system is a PC with either ARM64 or x86 architecture, includes more than 4GB of RAM, and already has Windows Subsystem for Linux installed. The download of AlmaLinux is a svelt 83MB and can run on either Windows 10 or 11. If you don't already have WSL installed on Windows, open Powershell (with elevated privileges) and issue the command `wsl --install`.

Although this has yet to be announced on the official AlmaLinux blog (*https://almalinux.org/blog/*), the distribution is officially available in the Microsoft Store and can be added to WSL for free.

Debian 11.3 Released with Numerous Bug and Security Fixes

The developers of Debian have been hard at work patching several security vulnerabilities and fixing bugs for the latest point release of their venerable Linux distribution. This new release fixes numerous security issues surrounding Apache's Log4j (such as CVE-2021-4104, CVE-2022-23302, CVE-2022-23305, and CVE-2022-23307) and includes other security patches for the likes of ClamAV, FLAC, GLibc, Golong, XTerm, atftp, eguardian, glewlwyd, gnupg2, and htmldoc.

Photo by Kenny Eliason on Unsplash

Although this release doesn't bring to light any new features, it's still a very important release, and all Debian 11 users should not hesitate to upgrade (especially those who use Debian with the Apache webserver).

The 11.3 release ships with kernel 5-10.0-13, which is patched against the Dirty Pipe vulnerability.

As far as user-facing software, it's important to remember that Debian focuses primarily on stability, so many apps might seem out of date. For instance, the version of LibreOffice shipped is 7.0.4-2, which is an LTS release. Although this version might be missing some of the new features of LibreOffice 7.3, it's a very stable version, so it's on-brand for Debian.

Get your copy of Debian 11.3 (*https://www.debian.org/download*) and make sure to check out the full release notes (*https://www.debian.org/News/2022/20220326*) for the distribution.

The First Alpha of Asahi Linux Is Available

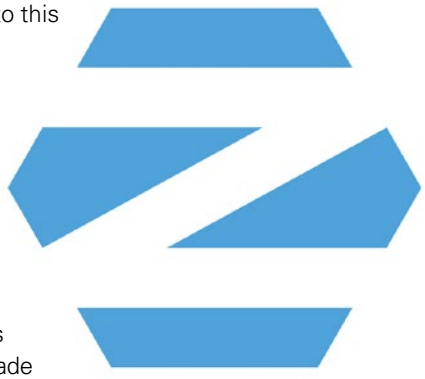
For anyone looking to install Linux on Apple Silicon, that task has been next to impossible … until Asahi Linux came into being. The announcement of the project came some time ago, but only recently have the developers finally announced the release of the alpha version of the OS.

This first release will contain bugs and doesn't include all features that will be found in the final release. Some of the features that do not yet work include DisplayPort, Thunderbolt, HDMI, Bluetooth, GPU acceleration, video codec acceleration, neural engine, CPU deep idle, sleep mode, camera, and the touchbar. Rest assured, however, that common features (such as WiFi, USB, NVMe, and the keyboard) function as expected. As well, some apps (such as Chromium and emacs) do not yet function properly.

For those looking to kick the tires of Asahi Linux, you will not have to first jailbreak your device, as the installer works out of the box, nor will the distribution affect the security level of your macOS install. You can install Asahi Linux by upgrading to the latest version, opening the terminal application, and issuing the command `curl https://alx.sh | sh`. You should also make sure to read official release notes for the alpha version of Asahi Linux (*https://asahilinux.org/2022/03/asahi-linux-alpha-release/*).

Zorin OS 16.1 Released with a New Kernel for Better Hardware Compatibility

Zorin OS 16 was only released seven months ago, yet the developers have been busy at work to bring the first point release for the desktop distribution. The biggest change to the operating system is the inclusion of the 5.13 kernel; though, that kernel is not patched against the Dirty Pipe vulnerability. However, immediately upon installation, an update will patch the kernel, so your desktop isn't in danger of falling prey to this nasty bug.



The 5.13 kernel brings better hardware compatibility for newer hardware (such as NVIDIA RTX 3050 GPUs, 12th Generation Intel Core processors, Sony PlayStation 5 DualSense controllers, the Framework laptop, and Apple's Magic Mouse 2).

Also included with 16.1 is the 7.3 release of LibreOffice, the Mesa 21.2.6 graphics stack, and all the usual goodness that comes along with Zorin OS (such as professional-grade creative apps, and different desktop layouts).

It should also be noted that the developers of Zorin OS decided to send all profits of sales to aid Ukraine, from release day (Thursday, March 10) until March 17. Although by the time you read this that effort will have ended, it's worth mentioning that the developers are doing what they can to send aid to the people of a war-torn nation.

Read the official Zorin OS 16.1 release notes (*https://blog.zorin.com/2022/03/10/zorin-os-16-1-released-support-for-ukraine/#support-the-people-of-ukraine*).

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

What Is an Inode?
• Jeff Layton
If you are reading or learning about high-performance computing (HPC), where storage is a very important consideration, having a basic introduction to an inode is fairly important.

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

Exploiting, Detecting, and Correcting IAM Security Misconfigurations
• Stefano Chierici
Three IAM security misconfiguration scenarios are rather common: allowing the creation of a new policy version, the modification of a role trust policy, and the creation of EC2 instances with role passing.

Workspace ONE for Endpoint Management Empowered
• Jens-Henrik Söldner
VMware Workspace ONE provides a secure and user-friendly digital workplace. We look at the features, components, and architecture of Workspace ONE, as well as application management and simplification of the integration of end devices through user self-enrollment.

Zero Trust as a Security Strategy
• Martin Loschwitz
Acceptance of zero trust models such as BeyondCorp by Google or LISA by Netflix lags in Europe, where endpoint security is king. We examine why this situation must change by looking into the principles of modern zero trust concepts.

Red Hat Adds Common Criteria Certification for RHEL 8.2

Red Hat Inc. has further strengthened its RHEL platform by adding Common Criteria Certification for RHEL 8.2.

For those who aren't familiar with Common Criteria Certifications, the goal with these types of certifications is to assure a product meets specific security criteria for specific computing environments. These certifications go through rigorous validation using standardized and repeatable testing via a third party.

RHEL 8.2 was certified by the National Information Assurance Partnership (NIAP) after the testing and validation were handled by Acumen Security (a US government-accredited lab). The operating system was tested and validated against the Common Criteria Standard for Information Security Evaluation (ISO/IEC 15408) against version 4.2.1 of the NIAP General Purpose Operating System Protection Profile. This process included the Extended Package for Secure Shell (SSH), version 1.0.

Paul Smith, senior vice president and general manager, Public Sector, North America, Red Hat, said of the certification, "This first Common Criteria certification for Red Hat Enterprise Linux 8 shows that Red Hat continues to maintain crucial IT security certificates for its next-generation operating system as well as the fact that the world's leading enterprise Linux platform can now provide a more secure and more intelligent platform for critical and classified deployments while retaining the flexibility, scalability, and innovation of Linux."

For more information, check out the official Red Hat announcement (*https://www.redhat.com/en/about/press-releases/red-hat-adds-common-criteria-certification-red-hat-enterprise-linux-8*).

Linux Kernel 5.17 Has Finally Arrived

The latest Linux kernel has arrived and it's chock full of surprises for new hardware, performance enhancements, and security fixes.

Chief among the new additions is support for the AMD P-State driver, which is a performance scaling driver that introduces a new CPU frequency control mechanism for AMD Zen-based CPUs. This new driver will offer vastly improved power efficiency and will go a long way to aid the performance on the Steam Deck.

Other hardware additions/improvements include updates for next-gen AMD GPUs, more improvements for Apple Silicon, initial support for Intel Raptor Lake S graphics, support for custom fan curves for some ASUS ROG laptops, a new driver for x86 Android tablets, support for Intel's new "platform firmware runtime update" (which allows for partial firmware updates without rebooting a system), and a new driver for the Lenovo Yoga Book.

Of note in the latest release is a fix for the Spectre v2 vulnerability, which affects a large range of processors from Intel, AMD, and ARM. This was actually one of the reasons for the delay because an embargo on public disclosure of the AMD patch for the vulnerability meant that the kernel automated testing process found a number of "fixes for the fixes." On this (*https://lkml.iu.edu/hypermail/linux/kernel/2203.0/06186.html*), Linus Torvalds (the creator of Linux) said, "None of this was really surprising, but I naïvely thought I'd be able to do the final release this weekend anyway." Torvalds continued, "We also really don't have any reason _not_ to give it another week with all the proper automated testing."

Although kernel 5.17 might not be the most glamorous release, thanks to the Spectre v2 fixes, it's a crucial one. Of course, it's now up to distribution maintainers to make this kernel available for users.

Get the latest news in your inbox every two weeks

Subscribe FREE to Linux Update

bit.ly/Linux-Update

openSUSE Conference

June 2- 4, 2022

events.opensuse.org

Zack's Kernel News

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

By Zack Brown

Retargeting the Magic SysRq Key

Andrzej Pietrasiewicz pointed out that not all computers had a "Magic SysRq" key on their keyboard. The SysRq key is generally the same key as Print Screen (PrtScn) on QWERTY keyboards, and the Linux kernel uses it to perform various operations even when the computer is completely locked up. It's invoked as Alt + SysRq + another key, where that other key is how you tell Linux what you want done. For example, Alt + SysRq + B will immediately reboot the system, without unmounting any drives or syncing any data.

Andrzej's patch allowed the user to configure a different key to have those magic powers instead of the SysRq key. By default, he felt that the F10 key would be appropriate.

He had submitted a similar patch in June 2020, which got a lukewarm reception, and now Andrzej was trying it out again, rebased onto the latest Linux kernel.

Randy Dunlap asked if Andrzej had tested the patch, because Randy noticed one of the lines of C code was missing the trailing semicolon. Andrzej replied that this was an error from the conflict-resolution phase of generating the patch and that he'd fix it. He quickly posted an updated patch.

Greg Kroah-Hartman's autoresponder email bot replied to Andrzej's patch, saying it appeared to be updating an earlier patch, but without specifying in the commit log what had changed since the previous version.

Elsewhere, Pavel Machek asked if F10 was the best key to replace SysRq on keyboards that had no SysRq of their own. He suggested Alt + Shift + Esc as a better alternative.

Meanwhile, Maciej W. Rozycki pointed out that "Lenovo in their infinite wisdom have placed the <PrintScreen> key (which in a traditional PS/2-keyboard manner produces <SysRq> when combined with <Alt>) in their keyboards between the right <Alt> and <Ctrl> keys. With thumbs not being as accurate as other fingers (and the overall misdesign of the keyboard and touchpad interface) you can imagine how often I have inadvertently hit <SysRq> combined with a letter key, wreaking havoc to my system (and of course I want to keep the key enabled for times when I do need it)."

Greg, however, caught up with the discussion and said, "The fact that this patch adds a 'new' sysrq key no matter what is a non-starter, please think through the consequences of such a change… So no, as-is, this change is not acceptable at all, and I would be amazed if anyone would ship such a thing." To which Andrzej replied that the patch didn't add a new SysRq key "no matter what." He added that, instead, "It does so only when the input device (keyboard) does _not_ have SysRq key at all. So I would say that this patch adds a replacement SysRq key if the SysRq key proper is _physically_ absent. Which seems not such a bad thing to me. The problem I'm trying to solve is exactly this: what to use as SysRq if there's no SysRq?" In a subsequent email he added, "Is 'connect an external keyboard' the _only_ choice Linux wants to offer to its users in case of devices such as e.g. Chromebooks?"

Greg was in no mood. He said, "it's an RFC, which can't be applied to the tree so I'll wait to review it 'for real' when you feel comfortable enough to submit it for inclusion."

And that was the end of the discussion.

It's unusual for an RFC to get stomped quite that hard by anyone other than Linus Torvalds, especially when there's an actual group of users for whom a given important feature is missing, such as the physical existence of the SysRq key.

On the other hand, this does seem to be a feature that Andrzej had already proposed in years gone by, so it's possible that with no significant changes to his proposal, the higher-ups such as Greg have less patience than they might have had earlier.

Author

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

Either way, as features go, it doesn't seem particularly stomp-worthy. Although maybe there's a lot more hiding behind Greg's cryptic "think about the consequences" statement. Maybe the top kernel people feel that it's crucially important to protect users from accidentally triggering unexpected SysRq actions by hitting key combinations that didn't seem to involve that key.

It seems like something that will continue to be debated, especially if Chromebooks have no SysRq key and Lenovo has placed it between Ctrl and Alt. Clearly the hardware manufacturers are not up to speed on the importance of SysRq. Either way, Linux does like to support everyone, so it's likely that Linus would want Chromebooks and Lenovo laptops to have reasonable SysRq options, if possible.

The Seventh Circle of Bug-Tracking Hell

Thorsten Leemhuis wanted the Linux kernel documentation to stop sending users to *bugzilla.kernel.org* to file bug reports. He said Bugzilla wasn't working very well and wasn't the preference of most kernel developers. He also summarized "the good, the bad, and the ugly" aspects of Bugzilla, by way of explaining his suggestion.

Among the good items, he said that roughly 15 people listed as kernel maintainers actually did use Bugzilla, both to track and fix bugs reported by others and to report bugs themselves. Various other subsystems, said Thorsten, also use Bugzilla. He named Greg Kroah-Hartman as one of the active Bugzilla users, though he said Greg's replies on Bugzilla were often simply to counsel the person reporting the bug to send the report to a particular distribution or mailing list instead.

Thorsten also affirmed that sometimes if multiple people reported the same bug in Bugzilla, the tool would help them find and merge their reports and then identify the right maintainer to bring the bug report to. So to that extent, he said, it was useful.

And as the final "good part" of Bugzilla, Thorsten said it offered a file-sharing capability for bugs that needed memory dumps or other files to go with them.

Among the bad items, Thorsten said that Bugzilla had a very out-of-date list of kernel components and maintainers, causing bug reports to alert the wrong people or get included in the wrong sections of the tool. Thorsten went into detail about exactly how many different components and maintainers were listed wrong or left out completely in the tool.

And among the ugly items, Thorsten pointed out that Bugzilla itself was never the official place to report kernel bugs. Nor does Bugzilla's front web page make that clear – or point to the `Documentation/admin-guide/reporting-bugs.rst` file in the kernel documentation, which does give guidance on how to report kernel bugs.

Combined with "the bad," Thorsten said, "that's the reasons why quite a few (a lot?) reports never get a reply from someone. During a randomly selected 2 week window at the end of November 2020 there were 60 public bugs and a bit more than half of them by the end of the year never got a single comment by anyone except maybe the reporter."

He added, "it's irrelevant at all who's to blame for the state of bugzilla.kernel.org; it for sure was set up with good intentions, it just didn't work out very well in the end. The situations just needs to be improved, ideally quickly; blaming it on someone isn't helping at all."

In terms of specifically excluding people from blame, Thorsten said, "But there is one aspect that should be noted here: The situation can't be blamed on the kernel.org admins. They are doing a good job at keeping the bugzilla.kernel.org up and the bugzilla codebase up2date. But as admins it's not their job to maintain the list of products and components."

He then considered the question of why he proposed ditching Bugzilla rather than fixing it. He said, "It's well known for years now that bugzilla.kernel.org is not working that well, but nobody ever stepped up to improve the situation. Maybe this commit gets something rolling. If that's the case this change can be reverted. For now the change is an improvement that was agreed on during the maintainers summit 2017 in a session discussion [on] regression tracking."

Randy Dunlap had a nit to pick in Thorsten's analysis. Thorsten had said that a lot of the email addresses used by Bugzilla actually pointed to domains

that don't exist, causing bug reports sent to those addresses to simply fail. Randy pointed out that these were actually "virtual" email addresses that were handled specifically by Bugzilla. So bug reports sent to those addresses wouldn't fail at all but instead would go to whoever they were configured to go to. So this was not a problem, although Randy acknowledged that the virtual addresses probably should be updated so more bug reports would go to more of the right people.

Thorsten bopped himself on the head for missing that, but he also asked who Randy believed would update that list of virtual addresses. Randy had said "we" should do it, so Thorsten asked:

"Who is 'we'? We as in 'the kernel community'? Or is there actually a smaller group of people you are referring to which is actively maintaining the list of products and components on bugzilla.kernel.org?

"Just trying to understand things better here, as there are other things that look strange to me and were mentioned in the patch description. For example: Why are there only 200 products and components on bugzilla.kernel.org (some of them for historic things like the ac-kernels) while the MAINTAINERS file has more than 2200 entries?"

By way of background, Thorsten added, "FWIW: I don't care too much about this whole thing, the whole idea for the approach I'm currently driving forward started when I did regression tracking in 2017. Back then I noticed quite a lot of bug reports on bugzilla.kernel.org never got a single reply, even if they were good and looked valid. That's why I brought this forward on the maintainers summit (*https://lwn.net/Articles/738216/*) and there it was discussed to basically go the route I'm taking currently. But I'm totally [fine] to adjust that route if there are good reasons, especially as that discussion happened some time ago."

Konstantin Ryabitsev from the Linux Foundation, and one of the kernel.org system administrators, remarked, "My general comment on this is that bug triage sucks and nobody really wants to do it for any extended period of time. :) There were times in the past when this or that person did step up and kept an eye on all incoming new bugs, properly routing them to the proper product/component, but they quickly burned out or found a less thankless occupation. Understandably."

He added:

"I want to encourage you and the rest of the developers to complain about this to the TAB [Linux Foundation's Technical Advisory Board]. It is entirely in their power to come to the Linux Foundation with the suggestion that perhaps bug triage should be a paid position. It's not a given that such a position would then be created and funded, but this for sure won't happen if these complaints don't reach People In Charge Of Funds at the LF.

"(FYI, this person shouldn't be me – every time I've come to the Foundation, I was asked that the proper way to go about it is through the TAB.)

"TBH, bug triage sounds like a great kernel developer semi-retirement gig. :)"

And he suggested:

"I'm not sure there's any single solution that will solve the problem. If we properly organize products/components, many people will just get lost in them and create all bug reports in 'other' (or 'helpdesk', as is the case lately).

"The sanest approach would be to have a simple web gateway to bug reporting:

– which distribution are you using?

– if they choose a distribution, show them where to report bugs for that distribution, because most bugs should start there, really

– on that page, also give a link: 'I'm a distribution maintainer and I want to report this bug upstream'

– if they click that link, let them fill out a freeform bug report that will create a new bug entry on bugzilla.kernel.org in 'Other/Other'

– creating a bug there will email the designated person in charge of initial bug triage

– that designated person or persons will then assign proper product/component, or simply forward the bug report to the proper maintainer if they are able to ascertain that

"This is far from perfect and still hinges on finding a person willing to do bug triage. However, it should hopefully improve the workflow without making it too complicated."

Close by, Thorsten remarked, "Sure, the LF or someone else could hire

someone […]; but I wonder if we have more pressing issues where the money would better be [spent] better. And even if not: getting that money and hiring someone would take some time…."

And in terms of finding someone among the kernel developers who'd be willing to do bug triage just out of love, Thorsten said, "Not me. ;-) That bugzilla.kernel.org is not working to well is known for years now, without anyone stepping up to improve the situation for real. Maybe my work/this discussion gets something rolling. But I guess until I see that happen I continue working towards discouraging people from using bugzilla.kernel.org, as otherwise things will just stay as they are."

At that point the discussion ended with no particular conclusion or decision coming out of it … until 10 months later, when Artem S. Tashkinov decided the moment was right to "express an utter dissatisfaction and even contempt for this proposal" – that is, the proposal to deprecate or remove Bugzilla from kernel development.

He said, "most open source projects are managed via bug trackers, I see no reason the Linux kernel should be exempt." He added that in spite of many developers and maintainers saying that they preferred to handle bugs via direct email, "it's an utterly preposterous, non-working and broken idea." Specifically, said Artem, "Tons of messages in various kernel related mailing lists have zero replies and are not acted upon in any way or shape." And in such an email-based system, there was "no way to see what are the current issues, what's resolved or not." He added that "Users have an extremely hard time looking for bug reports which are spread along God knows where," and "It's impossible to follow up on such messages except when you were subscribed to the original mailing list."

He listed several things he felt could be done to resuscitate Bugzilla, including updating the various components listed in the interface and having someone go through all the bug reports and sort them into the right groups. He concluded, "I really hate how this bugzilla is treated as an afterthought which no one really cares about. If no one cares, why does it exist in the first place? Let's shut it down then. Let's move to mailing lists and create a total mayhem of lack of accountability. 'Bugs? Sorry, I've had no time to read emails'. […] AFAIK kernel.org is a The Linux Foundation project. The organization is heavily sponsored by tons of companies. It would be great if we had a person actually invested in bugzilla.kernel.org."

Matthew Wilcox, in accordance with tradition, responded to Artem by saying, "Thank you for volunteering to take over administration of bugzilla.kernel.org. Can you lay out your plans for making kernel developers care about it?"

To which Artem answered, "Last time I checked the Linux foundation is not exactly living from hand to mouth and has enough financial backing from major corporations. Is this really about funding?"

The discussion died out shortly thereafter.

The thing I find fascinating about the discussion, and about the problem itself, is that bug tracking is a much-prized, highly desired aspect of pretty much any software project. And here is Linux, with tens of thousands of developers working to improve it all the time – most of whom do it for love, while many also do it as their full-time job. There is the well-funded Linux Foundation which supports Linux and open source in a not-for-profit capacity.

And yet even for Linux, the problem of bug tracking remains unsolved.

It's slightly reminiscent of the time when revision control was not solved for Linux, so Linus Torvalds took a couple weeks off kernel development and solved it for the entire world by creating Git. Maybe this is another such "if you build it they will come" situation. ■■■

Implementing Zero Trust Security

Aiming for Zero

Some old-school admins are still philosophizing about secure internal networks, but the experts have already moved on: Zero trust architectures use a reliable but complex strategy to protect the network from all threats – inside and outside.

By Martin Loschwitz and Joe Casad

In the third year of the coronavirus pandemic, it has now long been clear that many companies are likely to remember the virus as beneficial to their own business. VPN solution vendors definitely fall into this category: When home office and teleworking mutated from the exception to the rule in many companies, existing VPN solutions became substantially more in-demand. Hardly anyone expected the load on the VPN gateways to explode overnight. The large network manufacturers were happy to help out their customers as many admins purchased more powerful systems for OpenVPN (Figure 1).

But do VPNs really solve the problem of network security? Many experts are not so sure. Implicitly, all parties involved with VPNs start from the following premise: There is a difference between the internal and the external network, and it is safe to treat internal clients differently from external clients. VPNs are regularly used specifically because admins do not want certain services to be accessible from the Internet at all.

In many companies, VPNs form part of a security architecture that has grown organically over many years. Because security requirements have increased continuously over the past two decades, companies have invested more and more money in private networks and cut off more and more services from the outside world.

But cutting off external users only solves part of the problem. The classic division into an insecure external and a secure internal network implicitly assumes several things. First of all, it assumes that you can reasonably make assumptions about expected usage behavior based on location. This misguided narrative invites the belief that the company's own employees couldn't possibly mean any harm, unlike sinister hackers who hack into other people's environments from the Internet. Another faulty assumption is that you can safely make inferences about who the client is and what permissions they should have based on location. Anyone who makes it onto the internal network is automatically considered trustworthy and enjoys

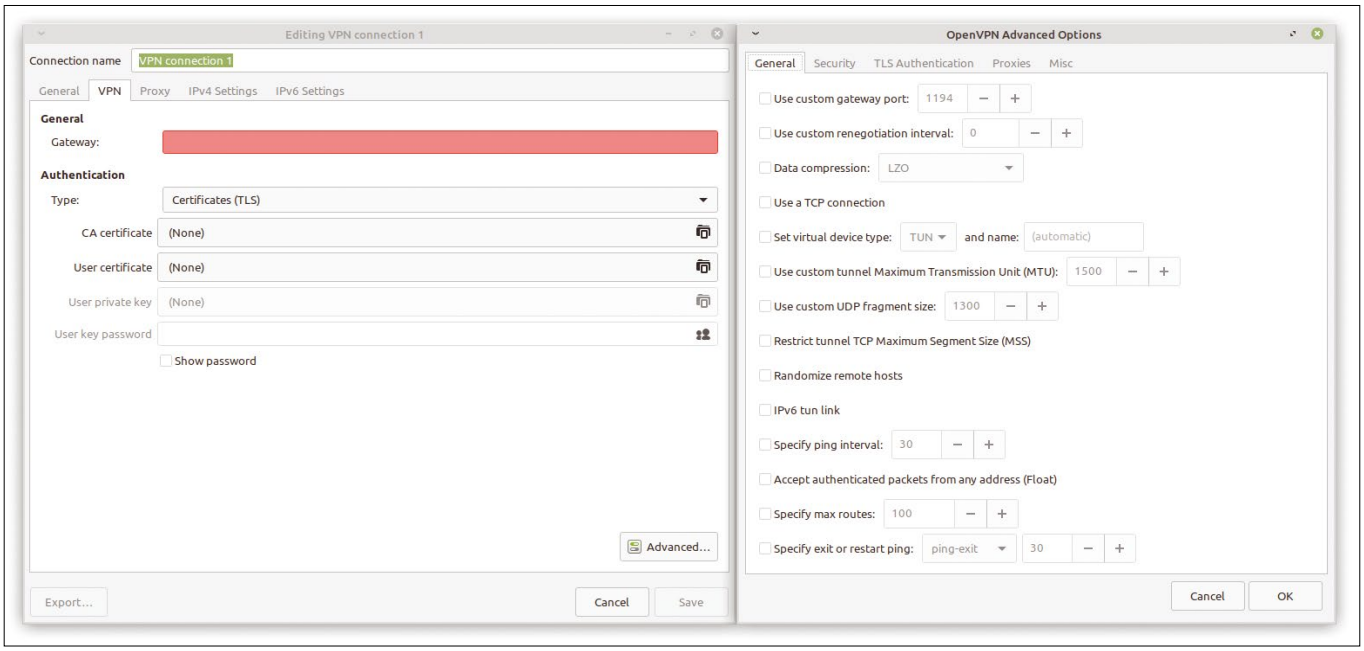


Figure 1: Homegrown VPN solutions, such as this one based on OpenVPN, are no longer suitable for state-of-art security. © KIT

expanded privileges, including access to infrastructure components that remain closed to external clients.

Today's knowledge of information security shows that such approaches are questionable. Clients on the internal network also pose a security risk. The risk could come from a user who opens a malware-infected attachment, a disgruntled employee, or a former employee who still has some form of access. Another problem is the constant presence of visitors coming and going with sophisticated (and highly networkable) mobile devices.

Zero trust is a set of principles designed to establish rules for eliminating the location bias in networking. In the zero trust model, *every* user is considered untrustworthy until vigorously proven otherwise. The zero trust rules also codify other best practices for network security, creating a state-of-the-art environment that corrects many of the out-of-date assumptions that put networks at risk.

Long History

The term "zero trust" dates back to a doctoral dissertation by Stephen Paul Marsh at the University of Stirling in Scotland [1]. Marsh's work was based on the concept of "trust" as something that can be defined mathematically, apart from the concept of morality and the complexity of human interaction. An international group called the Jericho Forum began to meet around 2003 to study the problem that they defined as "*de-perimiterisation*." The Jericho Forum raised awareness to the need for a new approach to network security, with the emphasis on eliminating the archaic idea of the internal network as a safe and protected space. The Jericho Forum's Commandments [2] were a forerunner to many of the principles now associated with zero trust.

Of course, it took some time for the real world to catch up with the theorists. It is worth remembering that the first local area networks as we know them today were isolated and typically didn't even use protocols that were routable on the Internet. When the push for Internet connectivity began in the 1990s, the concept of the LAN as a "safe" space was already firmly entrenched, and the effort to stop intruders from gaining entry focused on the gateway device. Large companies, in particular, have had a hard time letting go of the security strategies that worked for them in the 1990s.

In 2009, Google implemented the BeyondCorp security model, which is now considered an early implementation of a zero trust architecture. Meanwhile, academics and security experts continued to develop and explore the principles of zero trust, and the components that are now the building blocks of zero trust, such as encryption and identity management, continued to evolve in parallel.

The first government policy documents defining zero trust appeared a few years ago, with the US National Institute of Standards and Technology (NIST) SP 800-207 "Zero Trust Architecture" [3] in 2018 and the UK's National Cyber Security Centre (NCSC) "Network Architectures" [4] in 2019.

Here in 2022, the division of the world into "good" (internal) and "bad" (external) is becoming increasingly irrelevant, and the zero trust model is quickly gaining ground as the better approach.

Assumptions

Several formulations of the zero trust principles exist, all of which have similar goals. NIST SP 800-207 states these tenets as follows:

1. All data sources and computing services are considered resources.
2. All communication is secured regardless of network location.
3. Access to individual enterprise resources is granted on a per-session basis.
4. Access to resources is determined by dynamic policy – including the observable state of client identity, application/service, and the requesting asset – and may include other behavioral and environmental attributes.
5. The enterprise monitors and measures the integrity and security posture of all owned and associated assets.
6. All resource authentication and authorization are dynamic and strictly enforced before access is allowed.
7. The enterprise collects as much information as possible about the current state of assets, network infrastructure, and communications and uses it to improve its security posture.

The idea is that the organization will design a zero trust architecture based on these principles and then will implement that architecture to build a real-world zero trust network. The

principles ensure that no important details are left to chance. Close examination of these tenets reveals that many of the components of zero trust are things that modern networks worry about anyway, such as logging, monitoring, access management, and secure communication. The zero trust model creates an overarching structure and ensures that the principles are applied systematically.

In zero trust, a service always assumes that a client has sinister intentions until it proves otherwise, for example, by logging in with unique credentials. But even the login does not result in unrestricted trust; a sophisticated authorization strategy is an integral component of a zero trust environment. A client must have explicit authorization for a specific task in order to proceed.

Another central factor in zero trust is that all connections must be encrypted. Of course, the question arises as to what kind of encryption and what encryption tools – these questions are best answered at the design stage. One could ask whether something like a VPN is even necessary, if encryption is used on the local network anyway, and zero trust principles eliminate the concept of the internal network. Would it not be possible to just access anything from anywhere using SSH? This argument, however, reveals an incomplete understanding of zero trust.

Google's mail servers, for instance, are not directly accessible from the network via SSH. However, this is not because Google has given up on its own zero trust. It is far more the case that administrative access to infrastructure is usually reserved to a small, fairly static group of people. There is simply no reason for the average user of Google's services to access the servers via SSH. On the other hand, quite a few central Google services – those for the masses – can actually be accessed without any special connection such as a VPN. The zero trust model does not exclude the use of VPNs for special purposes. It merely contradicts the assumption that a client with VPN access should enjoy special rights.

There is much work to be done before admins can switch off the VPN in an existing environment or restrict its use, because zero trust cannot be achieved by installing a specific program. It is far more a matter of creating a strategy that integrates all the necessary components of an environment in the best possible way.

Users and Rights

As you can see from the preceding tenets, the zero trust system is heavily dependent on the need for a centralized authority for assigning granular rights to specific users. For any zero trust infrastructure, a centralized user and rights management system is a must-have. Two systems have established themselves in the market for centralized user management: LDAP and Active Directory (AD). Especially in corporate IT environments, you will often find a reasonably up-to-date instance of Microsoft's AD (Figure 2), and, with a bit of luck, LDAP compatibility is enabled. Most management software today offers support for at least one of these access methods.

If there is no ready-made user directory available and you want to create one for a Linux environment, you have several choices. One option is LDAP in the classic implementation of OpenLDAP. OpenLDAP is available for Suse Linux Enterprise Server, Ubuntu, and several other distros. Red Hat, however, does its own thing and delivers Red Hat Identification Management instead. Red Hat Identity Management is based on FreeIPA, a competitor implementation to OpenLDAP with some additional features. For example, FreeIPA comes with integrated management for SSH and SSL keys, systems and users out of the box, and a variety of CLI tools.

Roles

No matter which option you choose, what is almost more important than the existence of user names and passwords is a roles and authorization strategy that you map to the central user directory. This is where things get tricky. Opinions differ on how to map permissions in LDAP and other identity management tools.

One method that is frequently used is based on LDAP groups. In terms of the logic, you map the access permissions to a resource as a group membership. Access to the service is granted only to users who belong to the corresponding LDAP group. However, it is not possible to fine-tune this group assignment, which is why workarounds have developed. Often there are different LDAP groups for users and administrators of services. The catch is that the service that is then coupled to LDAP must also be able to evaluate these groups. There are also other hurdles. After all, LDAP also support roles and additional hierarchy levels. These factors are often a central obstacle.

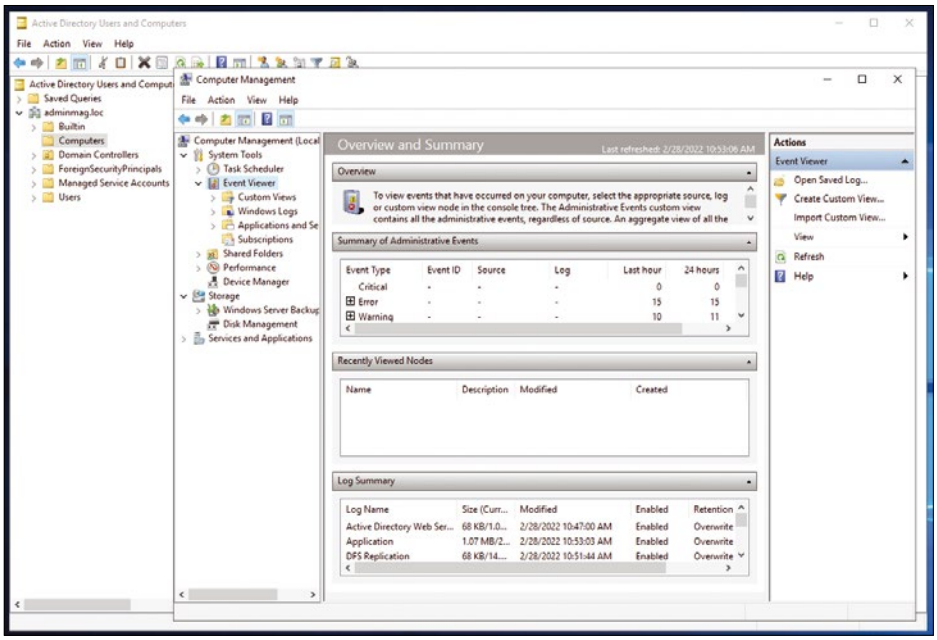


Figure 2: Active Directory can act as a central component in a zero trust architecture. © Microsoft

The complexity in assigning permissions underscores the fundamental importance of up-front planning in deploying zero trust models. Before system administrators even think about rolling out OpenLDAP or FreeIPA, they need to have a workable design for users and roles based on a RASCI matrix [5] that maps as many contingencies as possible in advance.

As usual, once the strategy is in place, far-reaching changes are difficult to implement and usually come at the cost of user resistance. On the other hand, if it is already clear in advance which authorizations are required for access to individual services, it is easier to implement the central user directory in a way that matches the design.

Finding Software

From the point of view of the system administrator, it is particularly problematic that zero trust has not yet been implemented as an established technical standard but instead only as a multitude of partly contradictory strategies. The definition provided with the SP 800-207 standard (described previously) is informative but a little vague. If you want your software to meet the requirements of zero trust, there is no ready-made script to guide you.

Network services and components can vary greatly in their support for zero trust. In most cases, central services such as existing groupware or mail servers offer the flexibility you need. Standard solutions such as Dovecot or Postfix, for example, can handle the connection to LDAP with many buttons for fine tuning, making it easy to implement a mail setup that supports zero trust.

The situation becomes more confusing when you are using proprietary tools that do not connect to LDAP at all or do not implement features such as two-factor authentication. In that case, you need to turn to workarounds: Libpam, for example, implicitly offers two-factor authentication and now has modules that integrate Google's Authenticator for one-time passwords. This even makes it possible to additionally secure SSH logins on remote systems when an SSH key is no longer sufficient by itself. However, implementing Authenticator via PAM in particular has massively affected performance in the past, so you need to consider your options carefully.

Several projects are intentionally designed to support the administrator in implementing zero trust. One well known candidate is Teleport (Figure 3), which is a broad-based replacement for OpenLDAP that promises "identity-aware authentication." In the background, Teleport relies on established standards such as X.509 or OpenID and exposes them to the user, while acting as a client for classic services such as SSH.

In practice, Teleport acts as a proxy that greatly facilitates the migration to zero trust. This approach offers an advantage, especially with regard to proprietary or legacy software. These applications can only be integrated into zero trust architectures with services such as Teleport. Anyone who has ever tried to reinstall legacy in-house software knows how difficult this can be several years after the program was created.

It is no coincidence that the Teleport website puts banks at the top of its list of high-relevance customer groups. Banks often run legacy software that you would hardly dare to think about integrating into modern security architectures without a proxy or some form of compatibility layer.

Mobile Devices

Smartphones and tablets have long since mutated into fairly powerful computers that can be used to handle simple everyday tasks in a convenient way. Special rules already apply to mobile devices independently of zero trust. As with laptops, the risk of loss means that encryption of the data on the device must have high priority. If mobile devices are maintained under a zero-trust umbrella, the company has a vested interest in maintaining control over a device at all times, even if it has been stolen or lost. In that case, it should at least be possible to wipe the device remotely and prevent further use by means of an activation lock.

In environments based on the zero trust standard, mobile devices often play a significant role. Because authentication in a zero trust environment must be secured via multiple factors, a mobile device might act as a security token via a service such

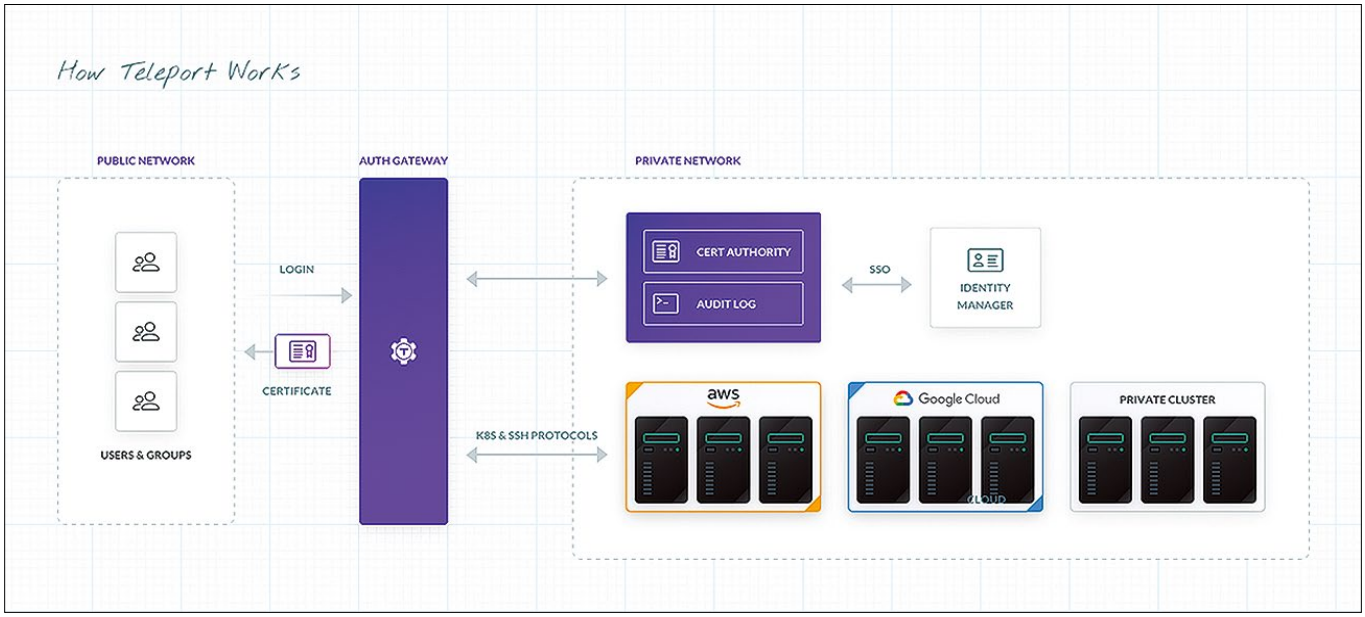


Figure 3: Teleport acts as a proxy between a zero trust architecture and other applications. © Teleport

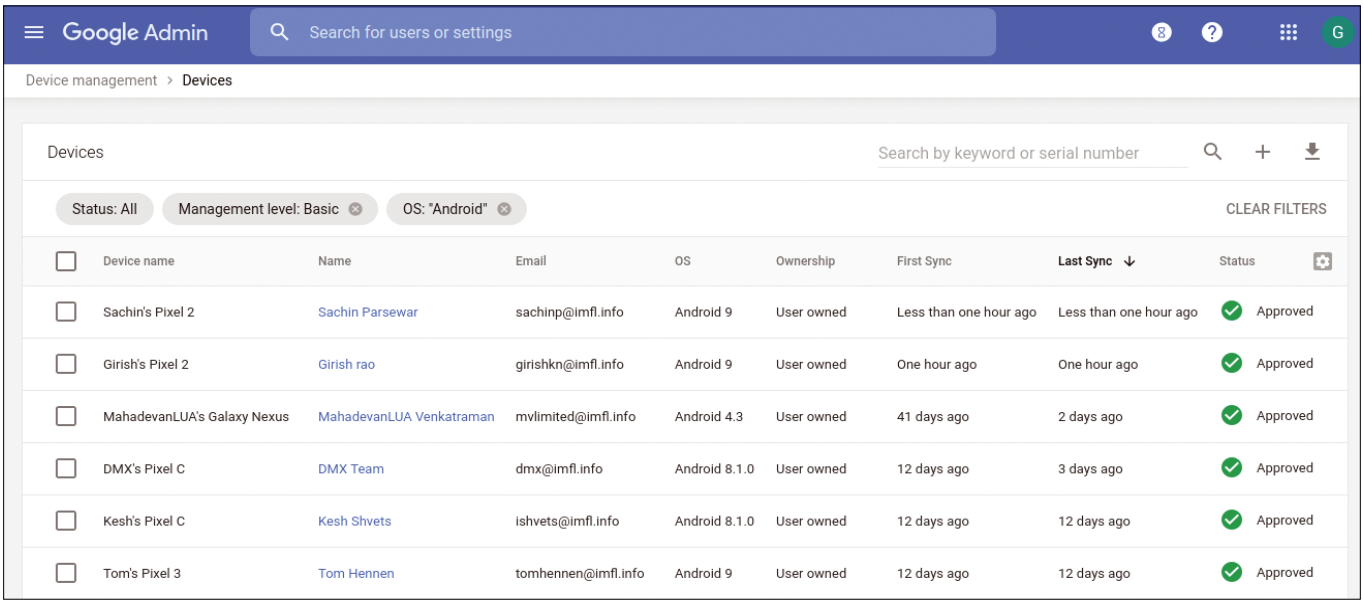


Figure 4: Mobile devices play a central role in zero trust systems. Admins find it hard to avoid using a device manager such as Google's setup for Android. © Google

as Google Authenticator. Of course, this means that the security measures we have looked at thus far have to be observed even more strictly (think unlock mechanisms). If a device can be easily unlocked, the Google Authenticator installed on it as a second factor is rendered useless. A secure and suitable unlock configuration is therefore necessary.

As central as the role of mobile devices in zero trust environments is, there are hardly any sensible options for managing the devices centrally with Linux on-board tools. At least there is nothing at the software level that could even begin to compete with the central tools from Google (Figure 4) or Apple (Figure 5), which offer features such as the option to remotely wipe a lost smartphone. If you issue cell phones to employees, take the security of smartphones into account in your planning for zero trust. It is hard to avoid biting the bullet and hiring the services of the two major manufacturers to help with your zero trust strategy.

Build It Yourself or Buy It?

At one level, zero trust is a methodology – a means for organizing the network. In theory, you could build a zero trust implementation yourself using components available within the Linux environment. However, there are other ways to implement zero trust. Several companies on the market specialize in zero trust implementations. For instance, Google is considered a pioneer with its BeyondCorp principle and has long since shaped zero trust into a product complete with bells and whistles.

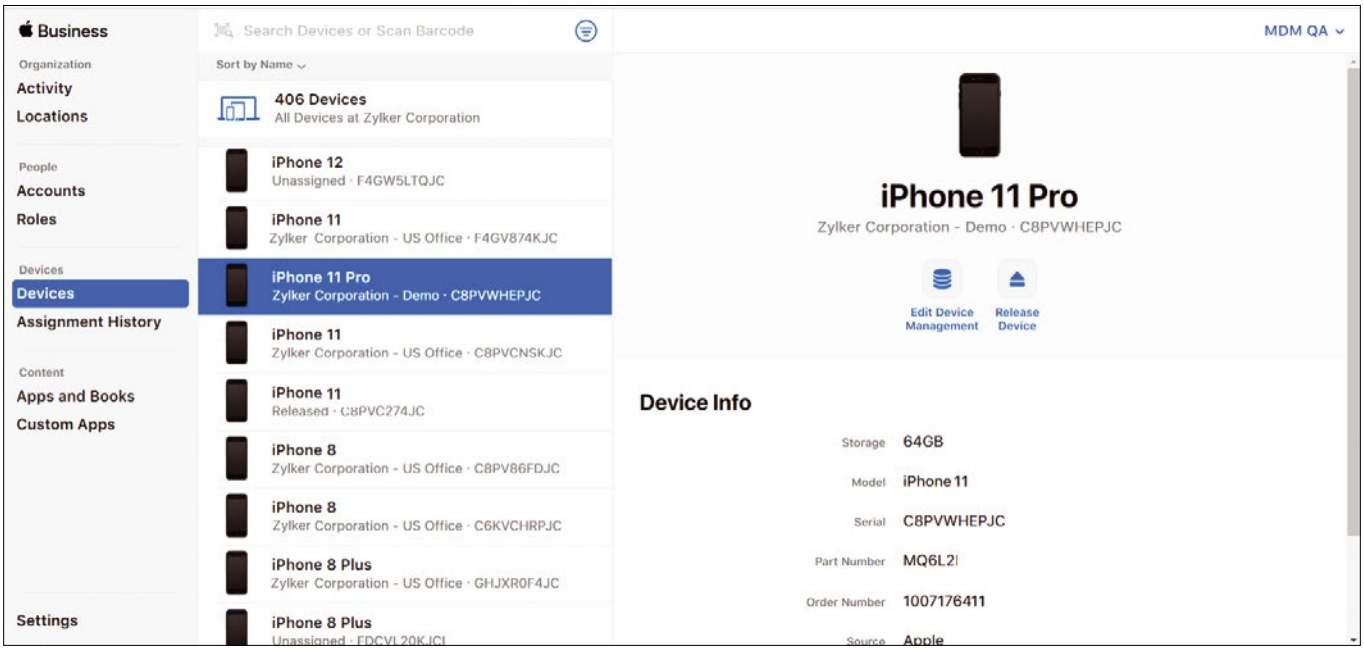


Figure 5: Apple also offers the option of remotely enforcing policy compliance with rules for iOS devices. © Apple

Anyone who wants to introduce zero trust quickly and comprehensively can commission Google to implement it. But there is a catch, of course: If you order everyday services such as email or office applications from Google, your data will inevitably end up in the Google cloud. However, the cloud is perfectly prepared for zero trust because it supports a connection to Active Directory and other authentication mechanisms and implements consistent rights management across the Googleverse.

Other service providers are also helping companies migrate to zero trust. Their offerings range from consultancy-only to ready-made cloud-based suites. From a European point of view, you need to keep in mind in all dealings with US-based providers that the US CLOUD Act and the GDPR cannot be reconciled. For European companies, the switch to zero trust is by no means a push-of-the-button experience but requires long-term planning in advance to ensure compliance with GDPR privacy requirements.

Complex but Necessary

Companies would do well to address zero trust as soon as possible. Overloaded VPN gateways and a collection of legacy firewall rules that no one understands anymore (created by employees who left the company years ago) are no match for the security threats of today. It is better to take the plunge soon rather than continuing to operate forever with 1990s-era security. ■■■

Author

Freelance journalist **Martin Gerhard Loschwitz** focuses primarily on topics such as OpenStack, Kubernetes, and Ceph.

Info

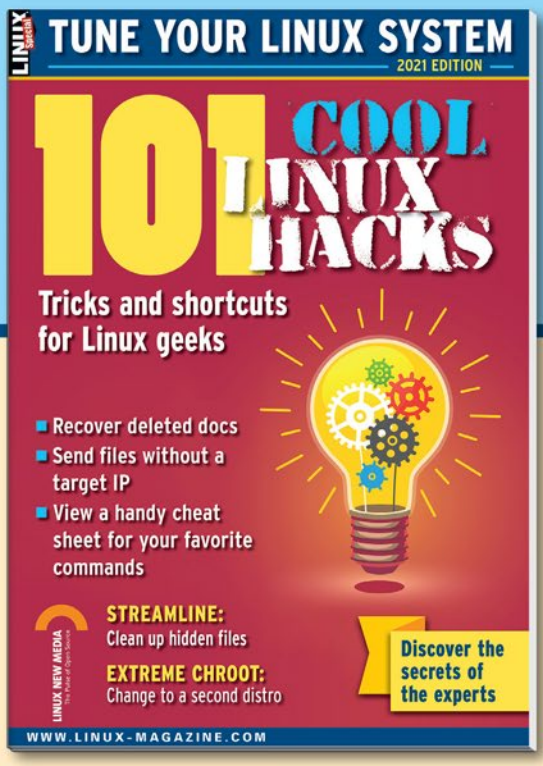
[1] Zero Trust Security Model: *https://en.wikipedia.org/wiki/Zero_trust_security_model*

[2] Jericho Forum Commandments: *https://collaboration.opengroup.org/jericho/commandments_v1.2.pdf*

[3] NIST SP 800-207 Zero Trust Architectures: *https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf*

[4] NCSC Network Architectures: *https://www.ncsc.gov.uk/collection/device-security-guidance/infrastructure/network-architectures*

[5] RASCI Responsibility Matrix: *https://en.wikipedia.org/wiki/Responsibility_assignment_matrix*



SHOP THE SHOP
shop.linuxnewmedia.com

GET PRODUCTIVE WITH
101 LINUX HACKS

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

- Undelete lost files
- Cure the caps lock disease
- Run C one-liners in the shell
- Disable your webcam and mic
- And more!

ORDER ONLINE: shop.linuxnewmedia.com/specials

Current desktop innovation

Pop!_OS

Pop!_OS, known for its innovation, customization, and user-friendliness, features one of the easiest tiling desktop options available. *By Bruce Byfield*

The years 2008-2012 marked an era of innovation for the Linux desktop: Gnome and KDE introduced radically new desktops, and Ubuntu developed Unity. However, the innovations were too much for many users. By the time development of Unity stopped altogether in October 2017, desktop developers had long since become more cautious. The age of bold experimentation seemed to be over. However, Ubuntu's switch from Unity back to Gnome immediately inspired the creation of Pop!_OS by System76, a company best-known for its Linux laptops and workstations [1]. Now, five years later, Pop!_OS has gained a reputation for innovation, particularly for the small enhancements in its installer, the modifications of Gnome and Ubuntu, and, most importantly, an option for the easiest tiling desktop yet.

Jeremy Soller, Pop!_OS's principal engineer, remembers: "When Ubuntu transitioned from Unity to Gnome Shell, the result was something we felt could be improved upon. Creating a new distribution was required due to the large number of changes we wanted to apply to Ubuntu." Over the past five years, this intent has evolved into the COSMIC desktop (which stands for Computer Operating System Main Interface Components, although the full name is rarely used these days), a graphic environment influenced by Unity, elementary OS, and tiling desktops such as awesome and XMonad, with an emphasis on user-friendliness that consists of a variety of small, thoughtful touches (Figure 1).

Sophie Coffey, System76's marketing director, writes that, "Pop!_OS is an operating system for STEM and creative professionals who use their computer as a tool to discover and create. Developers,

Photo by Jason Leung on Unsplash

Figure 1: Pop!_OS's COSMIC desktop showcases its emphasis on user-friendliness.

engineers, and AI/Machine Learning professionals can benefit from Pop!_OS as readily as gaming enthusiasts."

Unlike most modern distributions, Pop!_OS is not a community-driven distribution. Because the code was posted to GitHub, Soller says, "we have community members who do from time to time contribute changes, but this is a minority of the changes that make it into Pop!_OS. Planning is done in our Mattermost instance. Most of the time, this is in a private chat between System76 employees." However, their development system notwithstanding, you can download Pop!_OS for free, and it is mostly free software, aside from some proprietary drivers. Software is available in both DEB and Flatpak formats.

Installing Pop!_OS

The Pop!_OS installer is built around eye-candy graphics reminiscent of pulp science fiction magazines, a theme used throughout the desktop giving it an informal feel that users unaccustomed to installing operating systems might appreciate (Figure 2). More practically, help is hard-coded into the window – for instance, entering a short password produces the message "Not 8 letters," although you are not prevented from using the unrecommended password. In addition, when a choice is made, the titles on navigation buttons change, and, wherever possible, small shortcuts are offered, such as an option to use the same password for the user account and to unencrypt. All these are simple changes, but they should go a long way towards helping users.

When Pop!_OS boots for the first time, it starts with the Welcome app, a configuration wizard (Figure 3). Some of its screens, such as the location and time zone, could easily be part of the installer. Others, however, are a useful guide to configuration, such as whether to use a dock, where to place buttons on the top panel, which themes to use, and which social media apps to set up. Given that customization can take much longer than the bare installation, the Welcome app is a guide that I'd like to see more distributions use.

Ubuntu and Gnome Tweaks

Like any derivative, Pop!_OS remains closely tied to its parent distribution. However, according to Soller, Pop!_OS differs from Ubuntu in several ways:

- Mainline – not modified – kernels are used.
- Mesa updates are backported regularly.
- Snap is replaced by Flathub for universal package support.
- The installer has NVIDIA drivers preloaded and usable in a live session.

Even more extensive are the differences between the COSMIC desktop environment and Gnome:

- Virtual workspaces that are arranged horizontally, rather than vertically
- A default dock, with several choices for positioning
- Pop!_Shop, a fork of elementary OS's AppCenter (Figure 4)

Another major difference is the software selection. Pop!_OS includes standard applications such as Firefox and LibreOffice, but some of its software, such as Geary, is less common. Many utilities are written especially for Pop!_OS, such as the Notifications tool, which is combined with a calendar and has a *Do Not Disturb* option (Figure 5). The result is a much more unified look than a theme alone can provide – although Pop!_OS

Figure 2: Eye-candy graphics give the installer an informal, friendly look.

Figure 3: At first boot, Pop!_OS runs a configuration wizard.

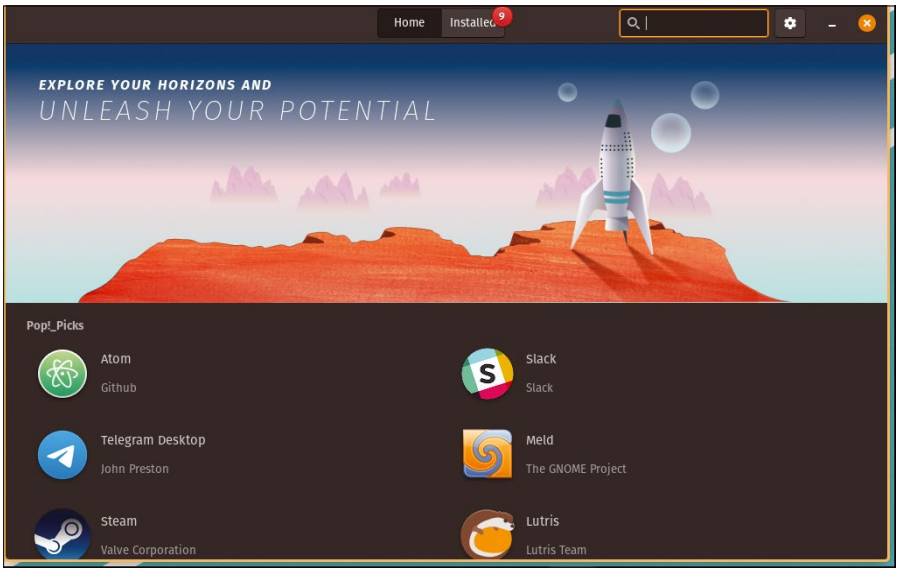


Figure 4: Pop!_Shop offers a sweeping collection of packages.

also includes its own themes and icon sets. More importantly, Pop!_Shop contains numerous items tailored to the intended audience of engineering professionals. At the top of the entries is a list of services, ranging from general services such as Steam and Slack to more specific services such as Atom and Mattermost for developers. In Pop!_Shop's categories, the selection is equally original. The Graphics category is atypical in its diversity, including such apps as Aeskulap, a medical imaging tool; Birdfont, a font editor; and Cura for 3D printing. You will also find more common apps such as Krita and Gimp and Plasma apps such as Gwenview and Okular, which are not usually found on Gnome installations. Even if you think you know Linux applications, this carefully curated selection is certain to contain choices you have never heard of before.

Probably the greatest difference is that Pop!_OS adds considerably to Gnome's settings (Figure 6). While many of the section headings are the same on both desktops, Pop!_OS has general settings for purposes such as setting the super key, enabling hotspots, and positioning buttons on the top panel. Similarly, the contents and placement of both the dock and workspaces each have several options. As well, under Privacy, Ubuntu's single setting for Connectivity Checking is further enhanced in Pop!_OS with Location Services, Thunderbolt support, and File History & Trash. File History & Trash is especially welcome, with File History acting much like the command history in a terminal and Trash Settings offering the deletion of temporary files and an expiration date. Differences in content make comparisons difficult, but not since Plasma have I seen a desktop with so many practical customization choices. Individual windows can even be customized separately.

Tiling Made Modern

By far the greatest buzz around Pop!_OS is its implementation of tiling. Unlike most desktops, which stack new windows on top of each other, tiling desktops arrange windows in a grid and are designed to be navigated by keyboard. They are almost as old as Linux itself and often used by developers, so at first, Linux veterans might wonder – as I did – what the fuss is about.

So what makes Pop!_OS's implementation so noteworthy? Levi Portenier, System76's QA team lead, comments that "Pop!_OS strives to streamline your workflow, so it only made sense that a user benefit driven feature like auto-tiling would eventually make its way into our desktop design. Pop Shell is built into Gnome, so it doesn't feel quite so foreign for those already accustomed to Gnome (which most Pop users likely are). Tiling in Pop Shell can be toggled on and off very quickly, so experimentation does not have to impact workflow. Plus, users that prefer normal floating mode can just leave tiling turned off."

Without tiling, Pop!_OS, like Unity, encourages users to open one window at a time. Most apps open full-screen, or nearly so, and although windows can be resized with the mouse, doing so can be awkward. This arrangement, I suspect, means that most users will eventually want to try tiling. Tiling is turned on by a slider on the upper right (Figure 7). When tiling is turned on, opening windows are automatically placed on a grid and can be dragged manually to other positions as needed. Practically speaking, any more than four open windows makes each too small to use, but other

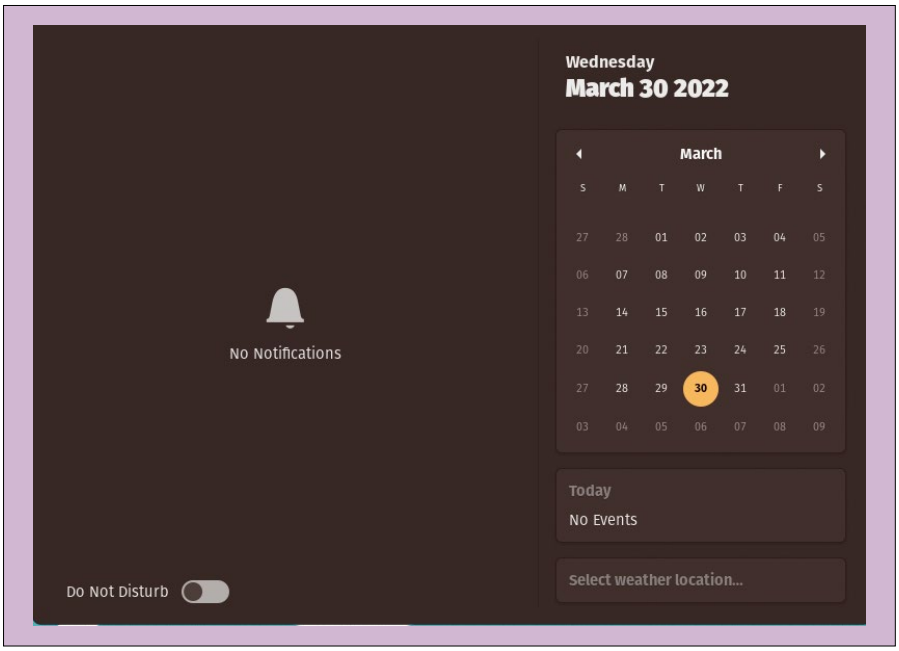


Figure 5: The Notification window is a good example of how Pop!_OS rethinks standard features.

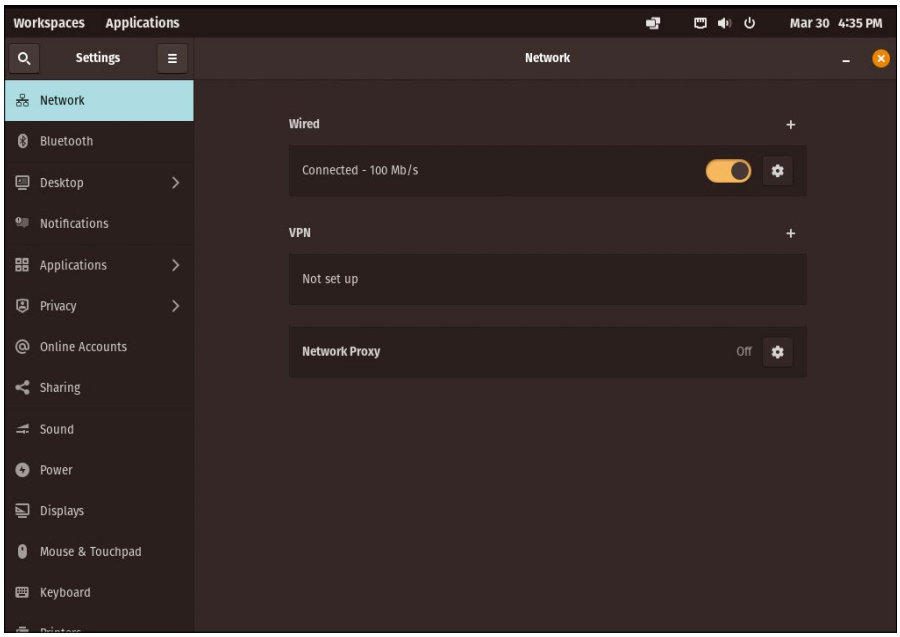


Figure 6: Pop!_OS makes extensive changes to Gnome's settings.

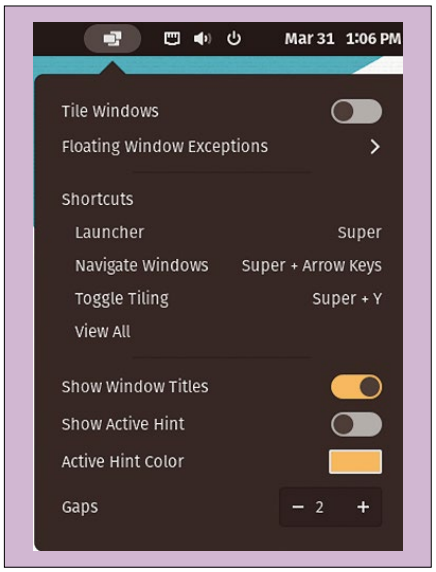


Figure 7: With a careful choice of controls, the tiled desktop is accessible even to new users.

windows can be added to a virtual workspace or else dragged by the titlebar to be free of the grid and stacked on top of the others.

To make tiling easier, Pop!_OS has several options. Should an application fail to tile properly, it can be set not to tile. Titlebars can display to make navigation easier, and visibility can be improved by setting the size of the gap between windows on the grid. Most useful of all, a hint window displays the keyboard controls for changing the position of the current window on the grid and for changing the size of windows (Figure 8). Any regular user will probably want to learn at least a few of the keyboard controls, just as with other tiling desktops. All the same, Pop!_OS has generally succeeded in making tiling accessible to the average user.

Is Innovation Worth It?

Pop!_OS borrows features from other distributions and then reshapes them with its own thoughtfulness and thoroughness. By doing so, it has produced the most innovative desktop environment in the past decade. However, I wonder if System76 has made the same mistakes as earlier desktop innovations such as Gnome 3 and Unity. In its attempt to improve the desktop, has it simply replaced one form of clutter with another? More importantly, does its design assume that all users work the same way and that there is only one way to be efficient?

I do not pretend to have the answers. Anyway, it would be wrong to judge an effort that is still maturing. For now, all I can say is that Pop!_OS is a desktop environment worth watching. I suspect that its automatic tiling in particular will soon start to appear on other desktops. ■■■

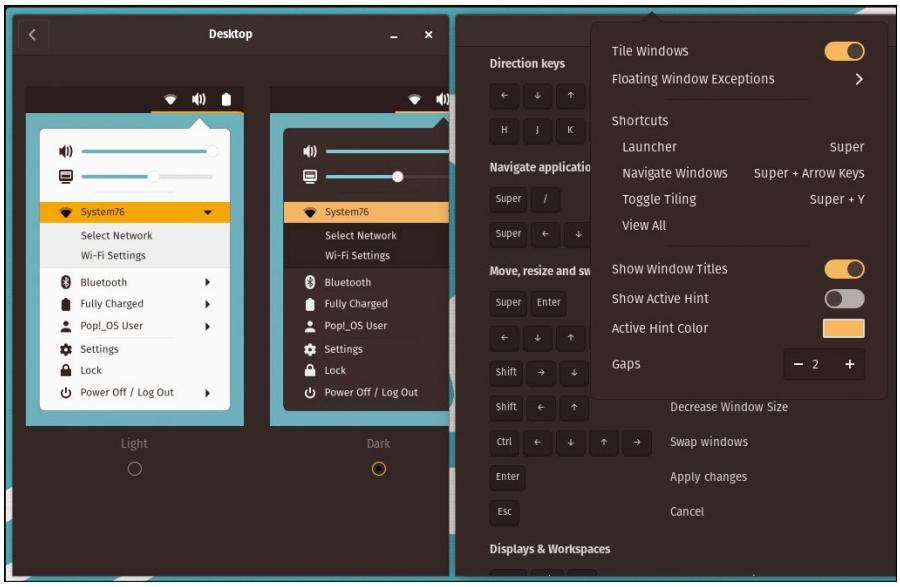


Figure 8: A list of hints makes learning to tile by keystrokes easier.

Info

[1] Pop!_OS: *https://pop.system76.com/*

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/*.

An updated Xfce desktop with Twister UI

Spruced Up

Twister UI modernizes the Xfce desktop, making it ideal for both new users and old hardware. *By Erik Bärwaldt*

At 25 years of age, the Xfce desktop is a veritable dinosaur among Linux work environments. While this lean and fast desktop is still great when it comes to saving resources, visually, the Xfce interface is starting to show it age despite a couple of overhauls. In addition, Xfce has fallen behind KDE Plasma and Gnome in terms of configuration options. Twister UI [1] changes all that. Specially designed for Xfce, Twister UI visually enhances the Xfce desktop while showcasing the potential of this old-timer interface.

Installation

Twister UI is available for download from the Pi Labs project page [2]. The developers offer three binary packages: one for 32-bit hardware (Xubuntu/Linux Mint), and two for 64-bit environments (Xubuntu/Linux Mint and Manjaro Linux).

After downloading a package, you first need make sure that the operating system is up to date by calling the respective update manager. Then change to the Twister UI download directory and grant the downloaded file execute permission by typing:

```
chmod +x TwisterUI<Version>Install.run
```

Then begin setup with:

Figure 1: With Twister UI's bright visuals, the Xfce desktop is virtually unrecognizable.

Photo by Towfiqu barbhuiya on Unsplash

```
./TwisterUI<Version>Install.run
```

Because the routine downloads some elements off the web, you will need Internet access during installation. After confirming a security prompt and authenticating to escalate to admin privileges, the setup configures the software, adding new themes, fonts, and additional programs. The system then prompts you to reboot, which you do by pressing *OK*. The result of this prep work is a visually impressive, bright, and colorful desktop (Figure 1) with a dock at the bottom of the screen and a horizontal panel bar at the top.

Growth

When you first look at the menus, which you can access by pressing the *Menu* button top left, you will notice a significantly larger number of applications than offered by Xfce. In particular, the Accessories, Settings, and System submenus show how Twister UI has grown the number of new applications. These new applications are not limited to Xfce programs. Gnome programs and applications developed independently of a desktop environment have also found their way into Twister UI.

In the Multimedia submenu, you can call up the fully preconfigured Kodi to turn your workstation into a media center at the push of a button. The Games submenu has also seen some additions in the form of the Lutris game browser and the Steam client. Other programs such as the internationally usable Hypnotix IPTV streaming application (Figure 2) developed for Linux Mint and the Discord client round off the entertainment segment.

Figure 2: Linux Mint comes with the Hypnotix free IPTV streaming app preinstalled.

Figure 3: ThemeTwister lets you bring the look and feel of Windows and macOS to the Linux desktop at the push of a button.

By adding Wine, Twister UI additionally integrates a runtime environment for Windows programs. The corresponding configuration tools can be found in the System submenu. Under the Accessories submenu, you will find the Winetricks tool for customizing Wine. Under the Settings submenu, maintenance programs, such as the Restore Twister Theme Config and Restore Twister UI Splash Screen routines, will help you reconstruct your system if necessary.

Unnecessary

In addition to the Firefox web browser already available on Ubuntu or Linux Mint, Twister UI also installs the Google Chrome browser, which comes with Google Docs Offline as an add-on. This add-on lets users edit various document formats in the browser.

Privacy-conscious users may want to remove Google's web browser because it sends data to Google without notice and without giving the user complete control. Because the LibreOffice suite is usually already available on the system and Firefox is far more friendly in terms of user data use than Chrome, the Google browser offers no additional benefits.

Convertible

The ThemeTwister tool, located in the Settings submenu, is one of the highlights in Twister UI. ThemeTwister lets you quickly customize the entire desktop environment with just a few mouse clicks. The developers have based the appearance of the available environments on other operating systems' graphical interfaces, such as Apple and Microsoft Windows. In particular, ThemeTwister allows newcomers to overcome any inhibitions in using Linux by imitating the look and feel of a familiar interface (Figure 3). Under the hood, however, the Xfce desktop remains active.

The visually emulated interfaces' original operating concepts are also largely retained in Twister UI. For example, in iTwister, a clone of Apple's macOS interface, you will also find the macOS controls in the window titlebars.

Matter of Opinion

Twister UI starts the Xfce desktop with the default selected theme, but you can use the LightPad application to display

the tile view familiar from the Gnome desktop environment. To this end, the developers added the LightPad launcher to the system. You will find LightPad in the modern desktop themes section of the desktop dock. The tool superimposes the tile view over the conventional desktop when called (Figure 4). To exit it again, just press Esc.

Frugal

Despite many visual effects, Twister UI uses minimal resources. For example, the system only needs about 500MB RAM when idle with the system monitor open. The CPU load is also kept within reasonable limits, which means that Twister UI can be used without problems on older computer systems with less powerful, dual-core processors, and little RAM. A minimum memory size of 2GB is recommended for smooth operation to allow even heavyweight applications such as LibreOffice or Firefox to run without noticeable latencies.

Conclusions

Twister UI impressively demonstrates the possibilities offered by modular desktop environments such as Xfce even in their old age. The Twister UI developers have gone to great lengths to give the respective themes a coherent appearance. The fonts, icons, and the window and panel bar designs stick as closely as possible to the imitated operating system. This makes Twister UI especially suitable for users switching from other operating systems to Linux who desire a familiar working environment. The old Xfce desktop proves to be so versatile that even aficionados of other operating systems have to take a very close look to notice that it is a Linux system. As an added benefit, frugal resource requirements make Twister UI especially useful on older hardware. ■■■

Info

[1] Twister UI: *https://twisteros.com*

[2] Download Twister UI: *https://twisteros.com/twisterui.html*

Author

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

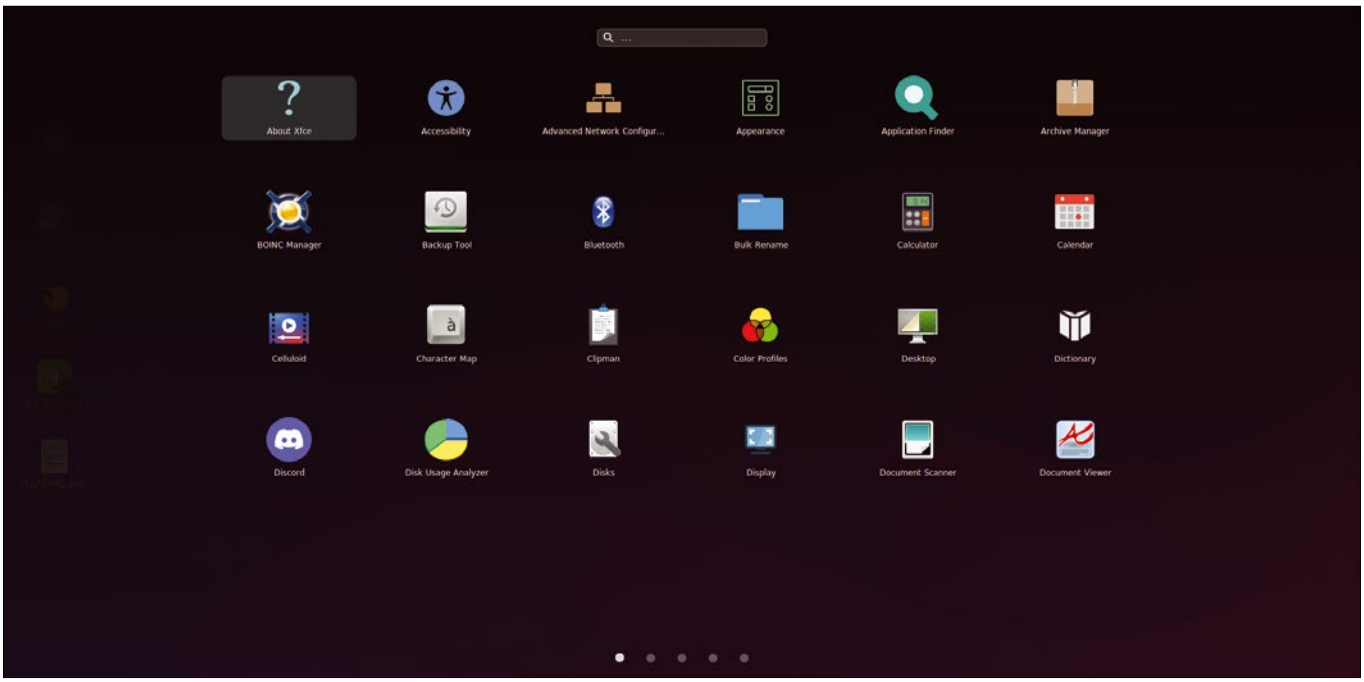
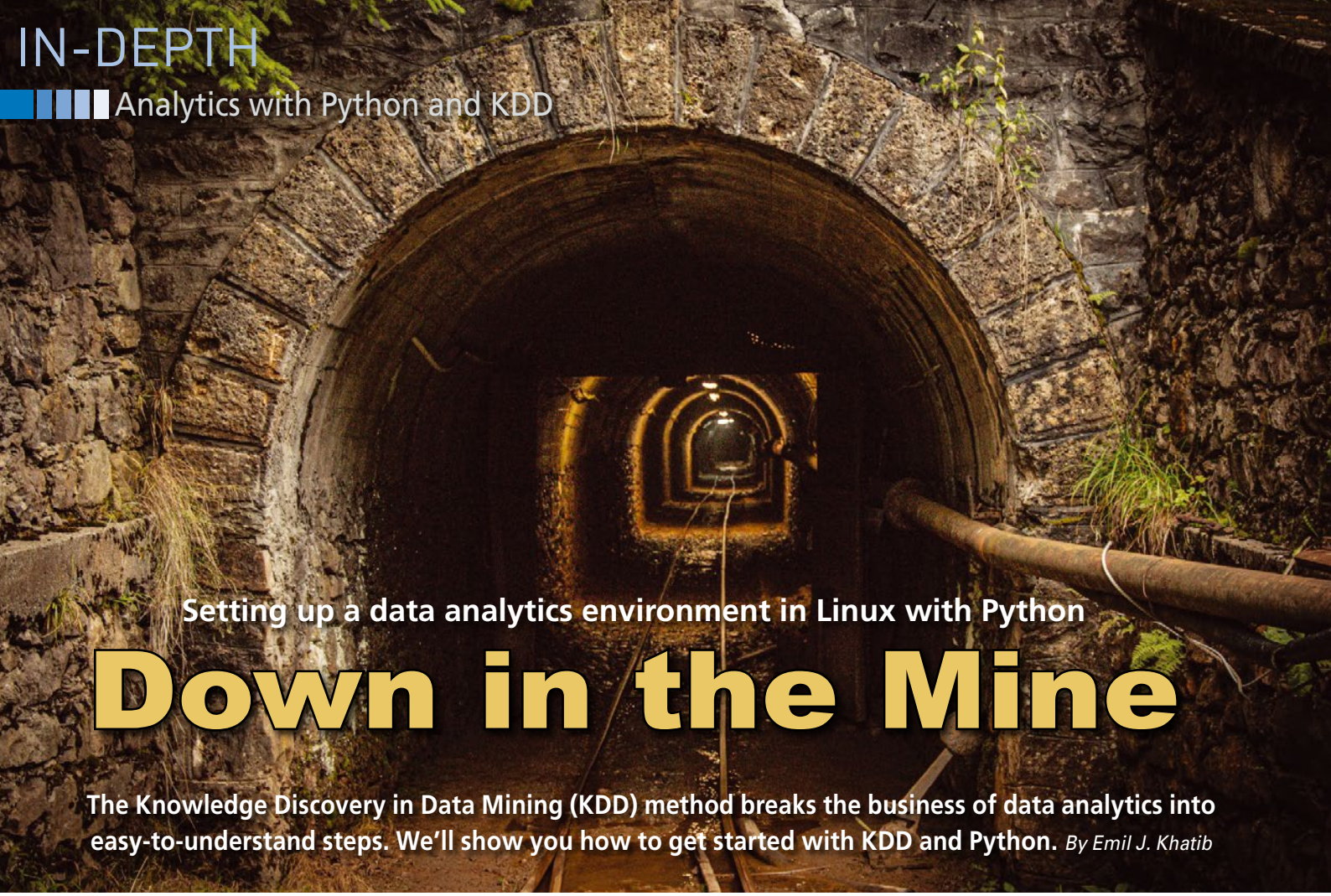


Figure 4: LightPad brings a Gnome-like tile view to the desktop.

SUSECON digital 22

Enable tomorrow's innovation today.

Register now at www.susecon.com



Setting up a data analytics environment in Linux with Python

Down in the Mine

The Knowledge Discovery in Data Mining (KDD) method breaks the business of data analytics into easy-to-understand steps. We'll show you how to get started with KDD and Python. *By Emil J. Khatib*

Data analytics is a major force in the current zeitgeist. Analytics are the eyes and ears on a very wide variety of domains (society, climate, health, etc.) to perform an even wider variety of tasks (such as understanding commercial trends, the spread of COVID-19, and finding exoplanets). In this article, I will discuss some fundamentals of data analytics and show how to get started with analytics in Python. Finally, I will show the whole process at work on a simple data analytics problem.

A Primer on Data Analytics

Data analytics uses tools from statistics and computer science (CS), such as artificial intelligence (AI) and machine learning (ML), to extract information from collected data. The collected data is usually very complex and voluminous, and it cannot be interpreted easily (or at all) by humans. Therefore, the data on its own is useless. Information lies hidden within the data, and it takes many forms: repeating patterns, trends, classifications, or even predictive models. You can use this data to uncover insights and build knowledge of the problem you are studying. For example, suppose you wish to measure the traffic in a parking lot that is monitored by a network of IoT sensors covering the whole city. Reading a single occupancy sensor doesn't say anything about the traffic on its own. Neither do the readings of all the parking sensors of the city without any more context. But the timestamped percentage of occupied places within the monitored parking lot does tell us something, and we use this information to derive insights, such as the times of day with maximum traffic.

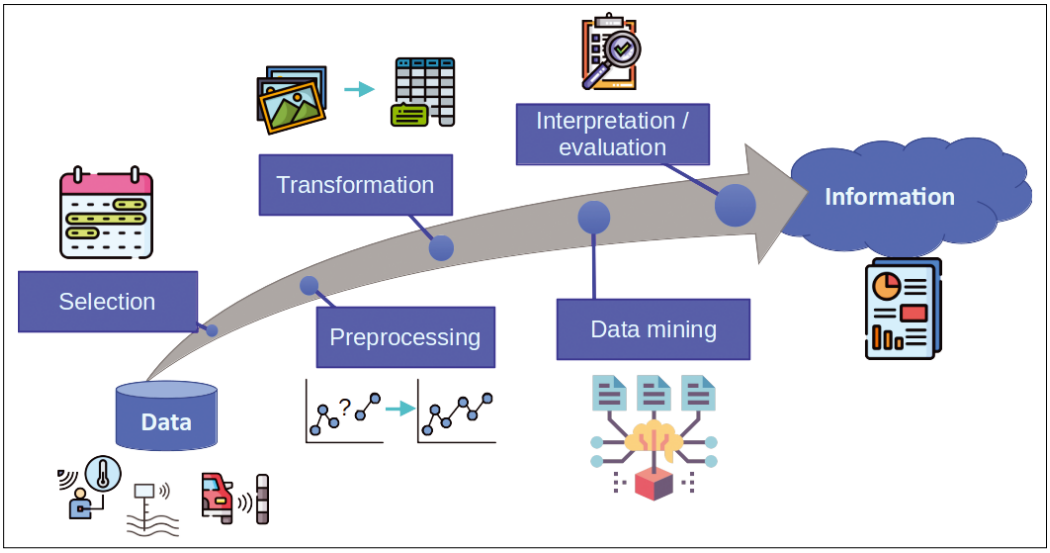


Figure 1: KDD is a systematic process for analyzing data.

Learning the mathematical background and analytics tools is only half the journey. Field expertise (experience on the problem that is being studied) is equally important. Some data scientists come from a statistics

Photo by Luca Maffeis on Unsplash

background, others are computer scientists who pick up the statistics as they go, and many are people starting from a field of expertise who need to learn both the statistics and the computing tools.

One important approach to data analytics is to use the Knowledge Discovery and Data Mining (KDD) model [1] – also known as Knowledge Discovery in Databases. The KDD process (see Figure 1) takes as input raw data from diverse sources (sensors, databases, logs, polls, etc.) and outputs information in the form of graphics, reports, and tables. The process has 5 steps:

- Selection – normally data sources are way more comprehensive than needed. Sensors might collect data from time periods or spatial locations that are out of the interest range or variables that are of no interest to the problem. This step narrows down the data that we know contains the information of interest. In the case of the parking example, we may only want to select data referring to the parking we want to monitor, as opposed to other parking spots throughout the city.
- Preprocessing – data is dirty; in other words, it may contain wrong or missing values caused by measurement errors or system failures. These errors can cause problems down the line, such as failures (in the best case) or hidden biases in the extracted information (in the worst

Table 1: Should You Use Anaconda?

Anaconda	Vanilla Python
Pro: All data science packages in one package.	**Con:** Need to install packages one by one.
Pro: Includes Anaconda Navigator, a GUI for managing the environment.	**Con:** Need to manually manage everything (not a con for many people).
Con: Uses the Conda package manager, which is not as complete as pip and interferes with it.	**Pro:** Has fewer "moving parts."
Con: Anaconda is not that well integrated into Linux package managers.	**Pro:** Python is very well integrated into most Linux distributions, unlike in Windows or macOS.

Python Elements

The Python environment consists of several important elements, including the Python interpreter, the package manager, the shell, and the virtual environment manager.

The interpreter is the virtual machine that reads and executes the code. A Python interpreter is usually present in most Linux distributions by default. Although most users rely on the vanilla interpreter (CPython), there are several alternative, specialized interpreters, such as Jython (integrated with the Java VM), PyPy (more performant than vanilla Python), or MicroPython (geared towards microcontrollers). Due to compatibility with libraries, vanilla Python is recommended for most of the tasks.

The package manager downloads and installs libraries that can be used to extend the basic functionality of Python. The default package manager is called pip, and it has an online repository of more than 300,000 libraries called the Python Package Index (PyPI) [2]. An alternative to pip is Conda, which is related to the Anaconda Python Distribution. Anaconda packages the basic data science tools of Python, and it is especially useful for Windows and macOS users, where Python is not that well integrated into the system. Anaconda is also available for Linux, but it may add one layer of complexity in exchange of providing a sane collection of preinstalled packages. See Table 1 for a comparison between Anaconda Python and vanilla Python in Linux.

Both Conda and pip can coexist in an install, but it is better to not mix them up if possible. Some package developers also distribute their libraries without integrating them into any repository. In that case, there are several common installation methods (`easy_install` or the `setup.py` script). Although the repositories see some kind of curation (albeit far from secure), downloading packages from the Internet without checking the code is always a bad idea.

Another important element is the shell, which is the interactive interface to the interpreter, not unlike a terminal like Bash or Zsh. The basic shell is offered by the interactive mode of the vanilla Python interpreter. It can be invoked by calling `python` in a terminal (see Figure 2), and it can be used for small tasks and testing out simple code. IPython offers an improved interactive experience, with functions such as syntax highlighting and code completion. You can also use IDEs, such as PyCharm (which has a FOSS community edition) or Spyder (which will be familiar to users coming from MATLAB).

A final important component is the virtual environment manager, which is a utility that creates isolated setups of the Python interpreter and packages for different projects. There is a base environment, associated with the system Python interpreter that is rooted within the basic OS filesystem. The virtual environments inherit the packages from the base environment and are rooted in a specific directory (normally within the home directory of the user). All the previously described elements can "live" either in the base environment or within a virtual environment. The basic manager included with Python is `venv`. While there are many other compatible alternatives (mainly to support older versions of Python), it is good practice to use `venv`. Another alternative available to Anaconda users is Conda, which also has the capability of creating virtual environments. Conda cannot be mixed with `venv`.

Figure 2: The vanilla Python interpreter.

case). Detecting wrong and missing data and filling it or dropping samples is part of the preprocessing stage. In the parking example, your might find null values when many parking sensors fail to send a reading or inconsistent values such as negative numbers.

- Transformation – once a clean dataset is in place, you might need to change its format to fit the requirements of the next stage. Tasks such as binning, converting from strings to numbers, obtaining parameters from images, etc. are just some of the thousands of possible actions in this stage. For the parking lot example, you might wish to do some binning, transforming the timestamp into a label representing an hour of a specific day.
- Data mining – this is the core of the whole KDD process. Data mining uses algorithms that extract the information from the clean, appropriately formatted data. Mining could consist of simple statistical computations (averages, standard deviations, and percentiles) or complex ML/AI processes (such as deep learning or unsupervised classification). In the parking lot example, I might wish to extract a time profile that represents the occupancy of the parking lot per hour. The process might be something like calculating the percent of occupied places per hour and day, and then averaging for several days at the same hour.
- Interpretation/evaluation – after you extract the necessary information, you still need one more step in which the results are validated before using the data to generate insights. Especially for complex outputs, such as predictive models, you need to evaluate the accuracy (normally with a separate validation dataset that was not used for the data mining process). Finally, you can use the information to generate insights or predictions. In the parking example, the resulting model could be used in a report that highlights the need for expanding the lot due to saturation at peak hours.

All of these tasks that form part of the KDD process must be supported by a computing platform. You'll need appropriate network connections to the sources, data storage, and a rich toolset to preprocess, transform, and mine the data. Linux is the ideal platform that provides all of these tools, thanks to its great network capabilities, the availability of databases (both small like SQLite and large, unstructured databases like MongoDB) and the great variety of FOSS tools for data processing and representation (such as LaTeX, gnuplot, or web servers for interactive and real-time reports). Among data scientists, Python stands out as an easy and capable programming language with a very comprehensive set of libraries for processing data from very diverse fields. And all of this comes with the advantages of FOSS.

The Python Programming Language

The principal benefits of Python are its ease of use, code clarity, and extensibility. Another important advantage of Python is the very wide ecosystem of libraries for many different fields. See the box entitled "Python Elements" for more on the basic components of the Python environment. No matter the topic (astronomy, macroeconomics, personal accounting, computer vision …), you will find a library in Python tailored to it. Coincidentally, data mining is also applicable to many different fields. The availability of both field-specific libraries and data analytics libraries makes Python quite appealing to data scientists. The box entitled "Python Data Science Libraries" highlights some of the important libraries used with data analytics applications.

Most Linux variants already have Python installed by default, or as a dependency of another package. But normally, only the interpreter is installed by default, so you need to manually install the rest of the elements. Table 2 shows the names of the Python interpreter, the pip package manager, and

Python Data Science Libraries

One supreme advantage of Python over other platforms is the rich ecosystem of libraries for data analytics, along with a myriad of smaller, field-specific libraries. Important libraries include:

- NumPy – the NumPy library provides a set of tools that make Python an efficient language for numerical computation. NumPy consists of the following elements: the `ndarray` object (which implements n-dimensional vectors and arrays), the operators and functions needed to perform mathematical calculations with `ndarray` efficiently, functions to read and write data from disk and memory, and various mathematical algorithms (e.g., generation of series, random numbers, transforms, etc.). NumPy is in the core of most mathematical libraries in Python.
- Pandas – like NumPy, Pandas provides new data types (mainly, the `Series` and the `DataFrame` objects), and a whole ecosystem of functions around them.
- Scikit-learn – provides a set of algorithms that can be used with medium-sized datasets to train classifiers and regressors with machine learning, as well as predictive models. In other words, the Scikit-learn library is mainly for the data mining stage of the analytics process. It offers algorithms such as random forests, neural networks, and ensemble methods. Scikit-learn also provides functions for preprocessing the data before the data mining stage.
- Keras/TensorFlow – for projects with very large datasets, TensorFlow provides a deep learning platform, which trains multi-layer neural networks for classification and regression. Keras is a high-level interface for TensorFlow, which simplifies its usage.
- Matplotlib – provides a very rich set of functions for graphical representations. From simple line plots to animated 3D graphs, Matplotlib allows almost any kind of graphic representation. It also includes functions for annotating graphs, and manipulating axes. Matplotlib is the equivalent of a Swiss army knife for graphic representations, although the learning curve for complex graphs might be steep. There are many other libraries for graphic representations in Python, tailored to specific uses and based on Matplotlib. The Matplotlib library is especially useful for the final stage of the data analytics process, where it is useful for representing information in an understandable manner.
- Seaborn – based on Matplotlib, Seaborn provides quick data visualizations that include empiric probability density functions, linear regression plots, grid views, and more. The Seaborn library is very useful for exploratory investigation of the data, before deciding, for instance, which algorithms will be applied in the data mining stage.

Table 2: Package Names for Python Components

Component	Ubuntu/Debian	Fedora	Arch	openSUSE
Basic environment	*python3, python3-pip, python3-venv, python-is-python3*	*python3, python3-pip*	*python, python-pip*	*python3, python3-pip*
Jupyter	*python3-jupyter* (no JupyterLab)	*python3-notebook* (no JupyterLab)	*jupyterlab*	*python3-jupyterlab*
NumPy	*python3-numpy*	*python3-numpy*	*python-numpy*	*python3-numpy*
Pandas	*python3-pandas*	*python3-pandas*	*python-pandas*	*python3-pandas*
Scikit-learn	*python3-sklearn*	*python3-scikit-learn*	*python-scikit-learn*	*python3-sklearn*
Matplotlib	*python3-matplotlib*	*python3-matplotlib*	*python-matplotlib*	*python3-matplotlib*
Seaborn	*python3-seaborn*	*python3-seaborn*	*python-seaborn*	*python3-seaborn*
Keras	*python3-keras*	Not in default repositories	*python-keras*	*python3-keras*

the `venv` library for some of the most popular distributions. Note that in most distributions, you must explicitly indicate that you are installing Python 3, to avoid confusion with Python 2 (which was deprecated in 2020, but is still a dependency of some software packages). When calling the interpreter, you must make sure that Python 3 is invoked, not Python 2. For instance, in Ubuntu, Debian, and openSUSE, you need to explicitly use the `python3` command when invoking the interpreter. You can fix this in Ubuntu and Debian by installing the package *python-is-python3*. Another way to fix this problem is to use a use a virtual environment, where the default interpreter of the system is overridden.

Once the basic packages are installed, you can proceed to create a virtual environment. Although this step is optional, it is highly recommended when working with many different projects in parallel, which is normal in the life of a data scientist. A virtual environment will have a root folder with its own Python interpreter and its own package selection. The new Python interpreter will have access to the packages of the system interpreter. Note that, if Python 2 and Python 3 coexist in the system, a virtual environment created with Python 3 will only have access to the packages of the Python 3 system interpreter. Any package you install in the virtual environment will only be accessible within it. Also, you do not need to call `python3` or `pip3` explicitly, because within the virtual environment, only Python 3 is available.

To create a new virtual environment, open a terminal, `cd` into the directory where you want to create it, and run the following command:

```
# python3 -m venv
new_environment_root
```

where `new_environment_root` can be any name. This command will only create the environment; you will not be able to use it until you activate it. For that, run the following command without changing the directory:

```
# source
new_environment_root/bin/activate
```

This command will modify the terminal session to use the virtual environment's interpreter, along with the packages it can access. It will also change the behavior of the `pip` package manager, so packages are installed in the virtual environment. If you install a package that is also installed in the system, it will be overriden only within the virtual environment. This is ideal for when you need a specific version of a package. When you are done working with the environment, change the terminal session back with:

```
# deactivate
```

Setting Up Jupyter

JupyterLab and Jupyter Notebooks are very important components in the Python

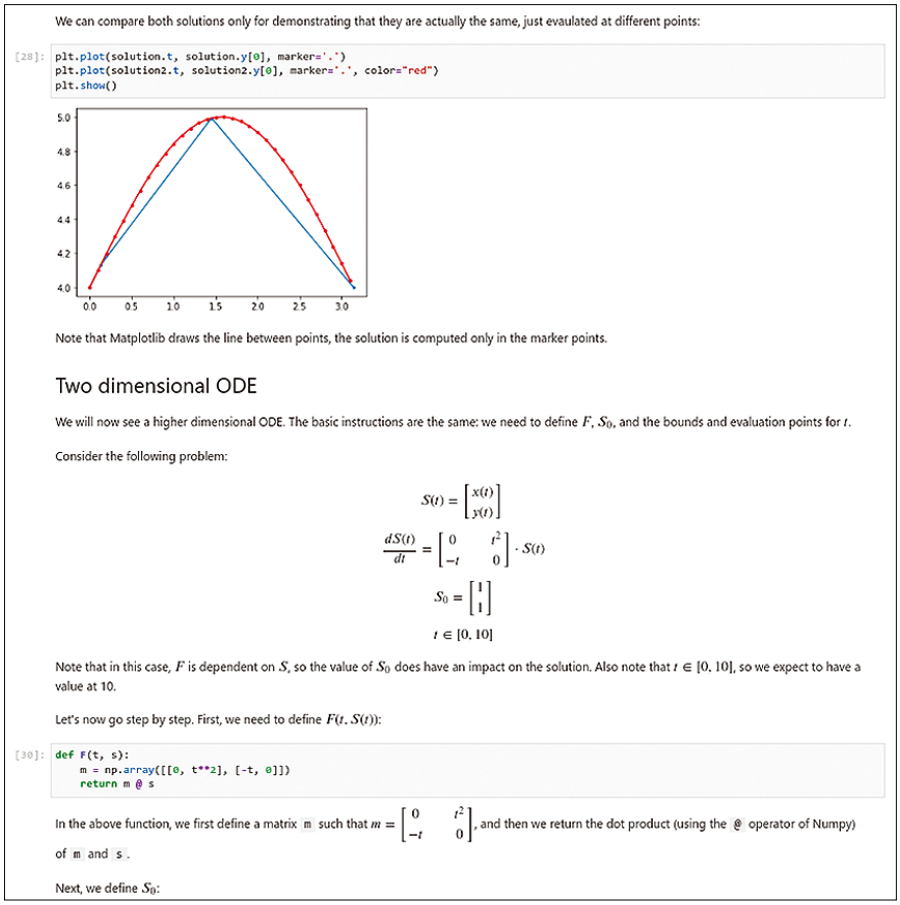


Figure 3: Example of a Jupyter notebook: Yes, this is a program, not a textbook!

Updating with Pip

It may seem redundant to have a package manager for Python when there are so many wonderful distro-specific package managers in Linux. But pip has some unique functionality that make it specially useful. Pip has a browsable repository [2], where the latest versions of libraries are promptly available. You can download and upgrade packages throughout the life cycle of a project, to integrate new functions or fixes. The first package you must upgrade is pip itself: `pip install pip --upgrade`. This should be done normally right after creating a new virtual environment. In general, to upgrade installed packages, enter `pip install package --upgrade`.

data analytics environment. Jupyter proposes a completely different way of using Python, by providing an experimentation + coding + documentation workflow. In JupyterLab, you can create notebooks (Figure 3), which contain both code cells (that can be run in an interactive way and in no particular order) and documentation cells (which can contain Markdown, HTML, and LaTeX code). Because Jupyter lets you run different cells in a nonlinear fashion, it is a particularly useful environment for experimenting and doing exploratory data analytics. The output of code cells (both in the form of text or graphics) is also embedded in the document. We can therefore produce comprehensive documents with interactive code and even graphics, which we can use to document the process of data analytics.

Once you have Python installed in the system and a virtual environment to work in, you can start an interactive session with the interpreter or run scripts right away; but in order to use a powerful data science environment, you must install some tools and packages. The first one is JupyterLab. You have two installation options: using `pip` or using the OS package manager. Each method has its own advantages and disadvantages.

To install Jupyter with `pip`, run:

```
# pip install jupyterlab
```

This command will make JupyterLab available to the current virtual environment (if none is activated, it will install it on the system's Python installation) and will download the latest version. Once Jupyter is installed, you will have to upgrade manually from time to time (see the "Upgrading with Pip" box).

The other way of installing JupyterLab is using the OS package manager, which will make it available to all the virtual environments and will subject it to the update cycle of the OS, but will not install the latest version. Especially for distributions that use oldish packages (Debian Stable, Ubuntu LTS …), the installed version might be significantly outdated. Table 2 shows the name of the package for Jupyter in the main distributions.

Regardless of how you install it, to run JupyterLab, execute the following in a terminal while the virtual environment is active:

```
# jupyter-lab
```

This command will launch a server in port 8888 by default (Figure 4). The server shows a URL that can be opened in a browser to access the JupyterLab interface. The server will attempt to launch the default browser. Note that the server window must not be closed while working with Jupyter.

The web interface (Figure 5) will allow you to create new notebooks (which is the default document type for Jupyter) in the directory you select in the file manager. Figure 5 shows the JupyterLab

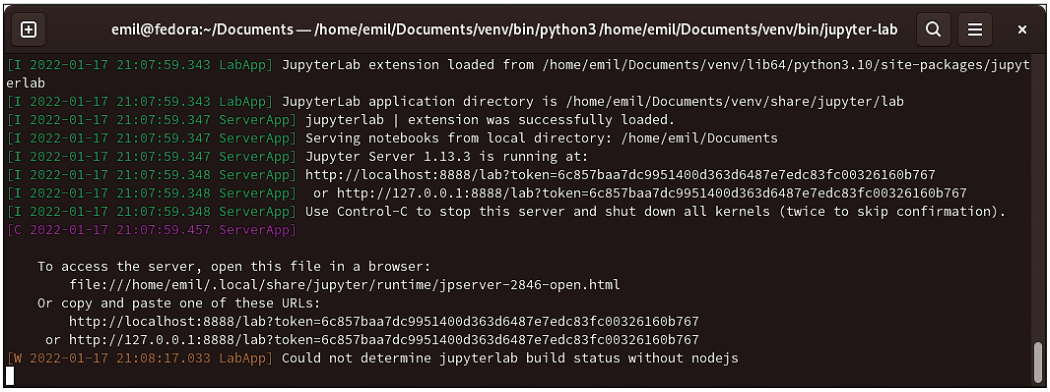


Figure 4: The Jupyter server terminal.

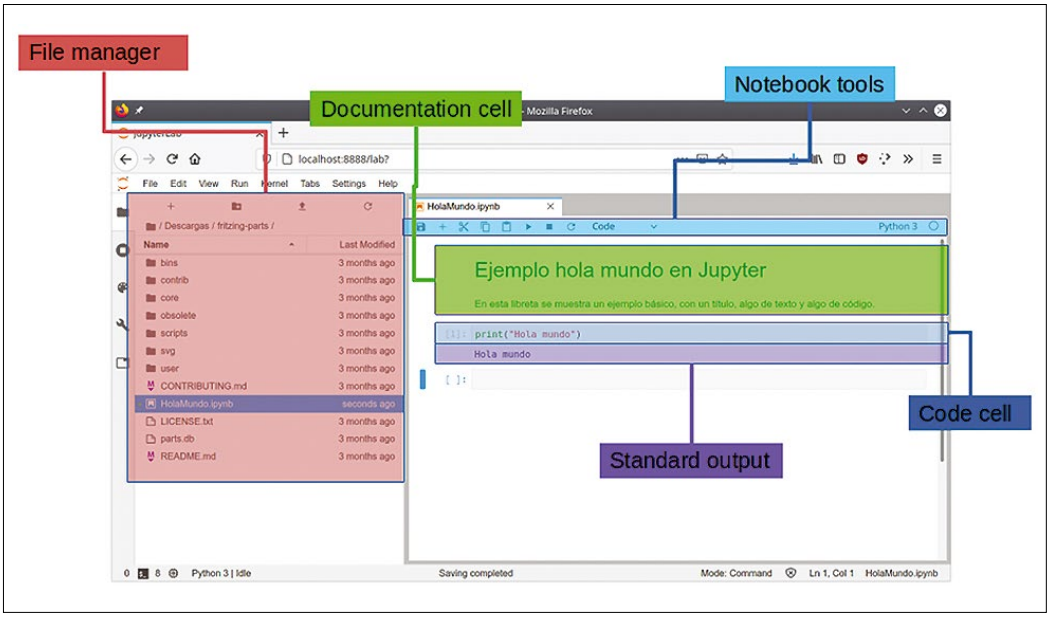


Figure 5: JupyterLab user interface.

screen, with the different areas marked. At the left side, you will find the file browser, where you can manage directories. On the right side is the main working area, where you will find a tabbed interface for the different notebooks. To create a new notebook, press the + button above the file explorer, which will open a new tab that offers the possibility of creating several new objects. Within the notebook, you will see documentation cells, where we can write Markdown, LaTeX, or HTML expressions, and code cells, where you write Python code. The code produces output right below the code cell, with text or graphic output.

Setting Up the Data Science Libraries

The final step is to install the main libraries. Again, just like with JupyterLab, you have the option of installing them with `pip`:

```
# pip install numpy ➥
pandas matplotlib sklearn
```

or with the OS package manager (Table 2).

This setup leaves you an environment that is ready both for exploratory data analytics, using JupyterLab, and for large batch processing, using the Python interpreter in script mode. Note that JupyterLab allows you to export a notebook to a Python script. You can also distribute results and documentation using Jupyter notebooks, to report data analytics work to clients. There is one more step that some users might want to take, depending on the specific data analytics project, and that is to install additional Python libraries. PyPI [2] lists all the libraries available in pip. It is good practice to explore the package index before a big project and assess the available field-specific libraries, as well as their maturity and compliance with project requirements.

Example

Suppose I want to understand the behavior of the traffic in a parking lot. I will obtain a profile that shows the hourly average occupancy of the parking lot based on data collected in several measurement campaigns, at different days, in different points of the city. First, I need to retrieve the raw data. For this example, I will use the Birmingham Parking dataset [3], which was used in research work on Smart Cities [4]. Download the full dataset using `wget`:

```
wget ➥
https://archive.ics.uci.edu/ml/➥
machine-learning-databases/00482/➥
dataset.zip
```

You can enter this command in a terminal window, or you can use the special character `!` within Jupyter to run a command in an embedded terminal. Next, unzip the data with `unzip`.

Given the great variety of formats, processes, and policies of data collection, dataset retrieval will look different each time; sometimes you need to download a ZIP file, sometimes you might just go to a database, or other times you might need to retrieve an SD card from an embedded system. That's the beauty of data science: Each project starts and develops in a different way.

For this example, I will include the Pandas data analytics library. Pandas completely changes the data workflow in Python, making it much more intuitive and easy. Internally, Pandas uses the mechanisms provided by NumPy, thus inheriting its efficiency. One common scenario is to load the data into a Pandas object, on which to perform preliminary data analysis tasks (especially the selection, preprocessing, and transformation stages).

The first step is to read the contents of the file into Pandas `DataFrame`, using the function `read_csv()` (Figure 6), to which you pass the mandatory filename parameter and an optional `parse_dates` parameter to force it to interpret one column as a datetime field. You can then visualize the contents loaded from the file with `display()`.

As you can see in Figure 6, the data appears in columns. The first column is `SystemCodeNumber`, which is an identifier of the parking lot. The second column (`Capacity`) shows the total capacity of the lot, and the third one (`Occupancy`) shows the current number of occupied parking spaces. Finally, `LastUpdated` shows the time and date of the last sensor reading.

The next step is to apply a selection process to only take the samples of the `NIA North` parking lot. For this step, use the `.loc` property of the Pandas `DataFrame` object, which allows you to filter the rows. The code shown in Figure 7 filters all the entries in `df`, where the parking lot name is `'NIA North'`.

The `.loc` property is very powerful, allowing filtering with a great variety of conditions. More information can be found in the Pandas documentation [5].

You now have the data of interest in `df`. Nevertheless, data in the real world

```
[3]: import pandas as pd
     df = pd.read_csv('dataset.csv', parse_dates=['LastUpdated'])
     display(df)
```

	SystemCodeNumber	Capacity	Occupancy	LastUpdated
0	BHMBCCMKT01	577	61	2016-10-04 07:59:42
1	BHMBCCMKT01	577	64	2016-10-04 08:25:42
2	BHMBCCMKT01	577	80	2016-10-04 08:59:42
3	BHMBCCMKT01	577	107	2016-10-04 09:32:46
4	BHMBCCMKT01	577	150	2016-10-04 09:59:48
...	...	...	...	...
35712	Shopping	1920	1517	2016-12-19 14:30:33
35713	Shopping	1920	1487	2016-12-19 15:03:34
35714	Shopping	1920	1432	2016-12-19 15:29:33
35715	Shopping	1920	1321	2016-12-19 16:03:35
35716	Shopping	1920	1180	2016-12-19 16:30:35

35717 rows × 4 columns

Figure 6: Loading the dataset into a `DataFrame`.

```
[4]: df = df.loc[df['SystemCodeNumber'] == 'NIA North']
```

Figure 7: Narrow the dataset to the parking lot of interest.

```
[5]: import matplotlib.pyplot as plt

     plt.stem(df['LastUpdated'], df['Occupancy'], markerfmt='.')
     plt.xticks(rotation=80)
     plt.show()
```

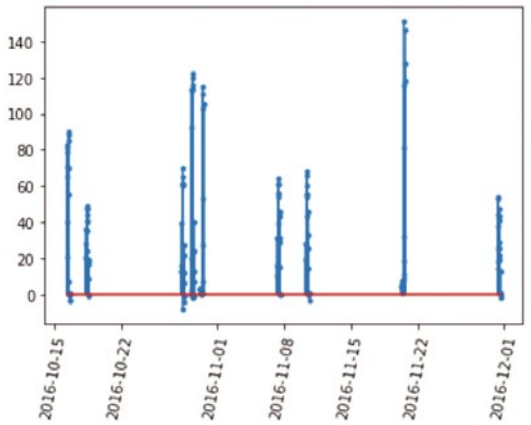


Figure 8: A visual representation of the data shows some inconsistencies.

```
[6]: df.loc[df['Occupancy'] < 0, 'Occupancy'] = None
     df = df.fillna(method='ffill')
```

Figure 9: Data cleaning step.

normally comes with errors and/or outliers. This dataset is not an exception, as you can see in the Matplotlib plot shown in Figure 8.

In Figure 8, the readings only come from isolated days where measurements were taken. Also, some values of occupancy are lower than 0 (which is impossible), so I need to remove these wrong values. These errors will be different in each project, so normally you will have to spend some time in this phase thinking of possible errors and chasing them. It takes some experience to do this quickly, and normally you might miss some errors and detect them further down the road. When you do so, you need to come back to this part of the study and add the appropriate mechanisms to detect them. Thanks to Jupyter's nonlinear workflow, you can do this easily by adding or editing cells in the appropriate places. Again, the `.loc` method will come in handy. In this case, I will replace the wrong values with `None`. If I knew a method to directly correct them, I could have used that method instead. Next, I will fill in the missing values with some generic value. Pandas offers the `.fillna()` method for filling missing data. You can fill in a constant value (for instance, `0`), or use the last known value. I will use the last known value in this case, because a good estimation for occupancy of a parking lot is the occupancy that it had previously. The code in Figure 9 shows the command for cleanup, and Figure 10 shows the corrected data.

Next is the transformation step. Start by thinking about what the modeling process (the next step) requires. Because you want to do an hourly average of the occupancy expressed as a proportion, you'll need two transformations. First, you need to extract the hour from the date-time field, as shown in Figure 11. With this, you can create a new column that only contains the hour. Next, you need to compute a new column that expresses the occupancy as a proportion, instead of an absolute value (Figure 12). Figure 12 also shows the dataset with the new columns.

To build the model in the data mining step, you actually only need the last two columns. Start by taking all the samples for each hour, and then calculate the average of the occupancy. In other words, group by the `Hour` column and calculate the mean. Grouping is such a common task that Pandas offers the `groupby` shorthand (Figure 13).

`groupby` will result in a new data frame, `model`, indexed with the unique values of `Hour`, and that new data frame contains the average value of all the other numerical fields grouped by `Hour`.

In this simple example, the data mining process was intentionally trivial. In some cases, the grouping and averaging operation can even be a part of the transformation step. Data mining can be very complex, including ML/AI processes, different kinds of numerical methods, and other advanced techniques. But there is one secret that all

```
[7]: plt.stem(df['LastUpdated'], df['Occupancy'], markerfmt='.')
     plt.xticks(rotation=80)
     plt.show()
```

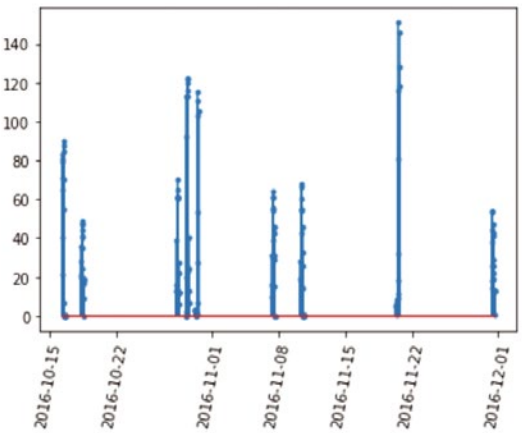


Figure 10: After cleaning, you do not see the inconsistencies.

```
[8]: df['Hour'] = df['LastUpdated'].apply(lambda x: x.hour)
```

Figure 11: Add a new column for the hour.

data analysts learn sooner or later: Most of the hard work of the data analytics process is already done before the data mining step. You can now use the model to represent a chart with the occupancy of the parking lot as a percentage for different hours of the day (Figure 14). More complex projects might involve live charts or detailed reports that are sent automatically by email to interested parties.

Conclusions

This article has been a primer on data science. I described how to take the KDD model as the outline for a typical workflow in a data analytics project. You also learned about the main Python libraries used with data science projects. Finally, I reviewed how to get the environment up and running, and I presented a simple example showing how to use it. This brief introduction is just the beginning. I'll leave it to you to discover how to apply the rich Python data analytics ecosystem to the problems you encounter in your own field of expertise. ■■■

Info

[1] Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth, "The KDD process for extracting useful knowledge from volumes of data," *Communications of the ACM*, 39(11), 1996, pp. 27-34

[2] PyPI: *https://pypi.org/*

[3] UCI Machine Learning Repository: *https://archive.ics.uci.edu/ml/datasets/Parking+Birmingham*

[4] Stolfi, Daniel H., Enrique Alba, and Xin Yao. "Predicting Car Park Occupancy Rates in Smart Cities." In: Smart Cities: Second International Conference, Smart-CT 2017, Málaga, Spain, June 14-16, 2017, pp. 107-117

[5] Pandas DataFrame.loc property: *https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.loc.html*

Author

Dr. Emil J. Khatib is a researcher at the University of Málaga in the field of cellular networks and industrial IoT. He also loves programming hardware and web and mobile apps. *www.emilkhatib.com*

```
[9]: df['Occupancy_percentage'] = 100 * df['Occupancy'] / df['Capacity']
     display(df)
```

	SystemCodeNumber	Capacity	Occupancy	LastUpdated	Hour	Occupancy_percentage
23873	NIA North	480	21.0	2016-10-16 08:01:13	8	4.375000
23874	NIA North	480	40.0	2016-10-16 08:27:13	8	8.333333
23875	NIA North	480	65.0	2016-10-16 09:01:15	9	13.541667
23876	NIA North	480	71.0	2016-10-16 09:27:13	9	14.791667
23877	NIA North	480	81.0	2016-10-16 10:04:13	10	16.875000
...	...	...	...	...	...	...
24030	NIA North	480	19.0	2016-11-30 14:28:40	14	3.958333
24031	NIA North	480	13.0	2016-11-30 15:01:39	15	2.708333
24032	NIA North	480	13.0	2016-11-30 15:28:40	15	2.708333
24033	NIA North	480	13.0	2016-11-30 16:01:39	16	2.708333
24034	NIA North	480	1.0	2016-11-30 16:28:40	16	0.208333

162 rows × 6 columns

Figure 12: Adding another column with the percentage of occupancy.

```
[10]: model = df.groupby('Hour').mean()
      display(model)
```

Hour	Capacity	Occupancy	Occupancy_percentage
7	480.0	2.750000	0.572917
8	480.0	14.700000	3.062500
9	480.0	36.235294	7.549020
10	480.0	55.400000	11.541667
11	480.0	47.800000	9.958333
12	480.0	58.500000	12.187500
13	480.0	36.900000	7.687500
14	480.0	35.733333	7.444444
15	480.0	33.750000	7.031250
16	480.0	22.461538	4.679487

Figure 13: Grouping and averaging are the focus of this data mining process.

```
[11]: model['Occupancy_percentage'].plot(kind='bar')
      plt.title('Occupancy model per hour')
      plt.show()
```

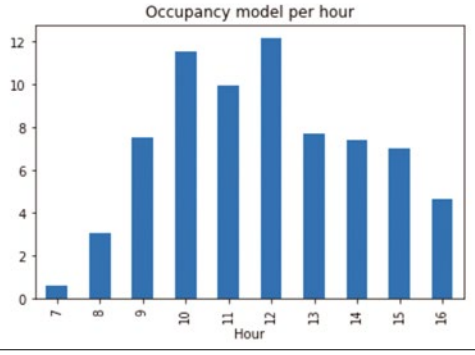


Figure 14: Final product of the data analytics project.


Hone your skills with special editions!
Get to know Shell, LibreOffice, Linux, and more from our Special Edition library.
The *Linux Magazine* team has created a series of single volumes that give you a deep-dive into the topics you want.
Available in print or digital format
Check out the full library!
shop.linuxnewmedia.com


FREE DVD!
openSUSE Leap 15.3
JOIN THE LINUX REVOLUTION!
ALL THE SOFTWARE YOU NEED!
GETTING STARTED WITH
LINUX
• MORE POWERFUL • MORE SECURE •
LEARN HOW TO SET UP A LINUX SYSTEM
• Listen to Music • Play Games • Process P
• Surf the Web • and Much More!
START
LINUX NEW MEDIA
WWW.LINUX-M
LINUX Special
301 BEST BASH COMMANDS
LINUX SHELL
HANDBOOK 2022 Edition
LINUX Special
SUPERCHARGE
YOUR LINUX SKILLS
Power at Your Fingertips
• Pipe and redirect output
• Monitor processes
• Create custom scripts
LibreOffice Expert
FREE DVD!
LibreOffice Full Version
Dive deep into the world's greatest free office suite
2022 Edition
LibreOffice
Expert
Edit and Save MS Office Files
Write Your Own LO Macros
Save time and automate
LINUX NEW MEDIA USA
LINUX Special
TUNE YOUR LINUX SYSTEM
2021 EDITION
101 COOL LINUX HACKS
Tricks and shortcuts for Linux geeks
Recover deleted docs
Send files without a target IP
View a handy cheat sheet for your favorite commands
STREAM
Clean up hi
EXTREM
Change to
LINUX NEW MEDIA
WWW.LINUX-MAGA
ARCHIVE DVD
RASPBERRY PI GEEK
THE COMPLETE ARCHIVE
2,000 pages of maker projects and more!
FREE DVD
$39.90 VALUE!
MakerSpace
HANDS-ON PROJECTS FOR MAKERS
BUILD A RASP PI RADIO!
MakerSpace #02
HANDS-ON PROJECTS FOR MAKERS
Cool Tricks!
Charge up with a saltwater battery
Painting with Light
Sure you need a programmable light stick
PiMiga 2.0
Get your game on with this Amiga emulator
MORE FUN FOR FPGA GEEKS!
LINUX NEW MEDIA

A command-line network intrusion detection system

Sniffing Out Intruders

Snort lets you protect your network from intruders with a customizable ruleset. *By Bruce Byfield*

Snort [1] is one of the oldest and most reliable network intrusion detection systems. Founded in 1998 by Martin Roesch, then the CTO of Sourcefire, Snort quickly became so popular that in 2009 *InfoWorld* declared it one of the top 36 pieces of free software [2]. Like the definitions in a virus detector, Snort relies on a series of rules to detect all known means of compromising a system. It is not difficult to install, but it requires preparation, and, the more customization, the more time-consuming your installation will be. What follows are instructions for a minimal installation for Debian-like distributions, which should be good enough for many users, especially on standalone machines.

To get the very latest protection, install Snort from source [3], using the usual `./configure`, `make`, and `install` commands. Most distributions also offer a package, although the package is often older than the latest version. However, in a mature application such as Snort, the differences between versions are apt to be minimal, and the rules you install are probably more important. Whichever version you use, you might want to create a Snort group and user solely for running the app, just for added protection. In addition, before installation, gather the necessary information (Figure 1) by running:

```
ip a
```

```
root@debian:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp5s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 4c:cc:6a:25:08:51 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.3/24 brd 192.168.1.255 scope global dynamic noprefixroute enp5s0
       valid_lft 83155sec preferred_lft 83155sec
    inet6 fe80::4ecc:6aff:fe25:851/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Figure 1: Installing Snort requires the network interface's name and IP address.

You will need the name of the network interface (the second item on the list) and the IP address (on the line starting with `inet`). Keep the terminal open so you can copy and paste as needed. If you use the Snort package from the Debian repository, the installation will offer you one or more choices, depending on the distribution and its version (Figure 2). As a beginner, you can simply select *OK* to continue and edit the configuration later if necessary.

Configuring Snort

When installation is complete, you need to edit `/etc/snort/snort.conf` as root (Figure 3). At a minimum, you need to find the lines that begin with `ipvar` and replace the placeholder `HOME_NET` with your network address. The `snort.conf` file is heavily commented; eventually, you should go through its dozens of options with the Snort documentation and `read.me` files open, uncommenting and adding entries as needed for your circumstances – a process that can take hours to do completely. For now, however, all you should note is that `snort.conf` is divided into nine steps, each containing dozens of fields:

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/*.

Photo by Jack Seeds on Unsplash

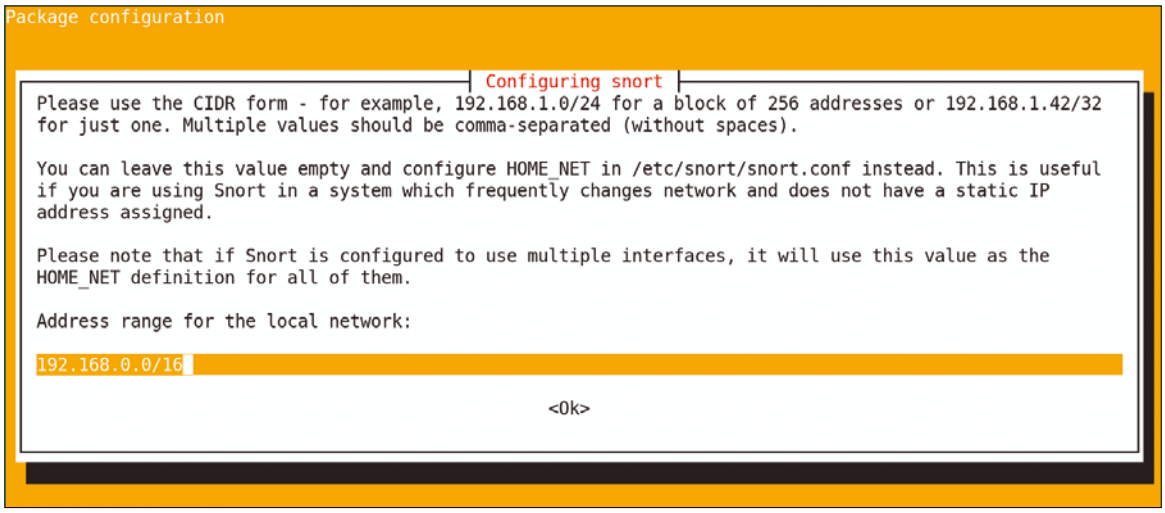


Figure 2: As part of the installation, Debian systems ask one or two questions about package configuration. Usually, users can accept the installer's defaults.

1. Set the network variables/addresses.
2. Configure the decoder.
3. Configure the basic detection engine.
4. Configure dynamic loaded libraries.
5. Configure preprocessors.
6. Configure output plugins.
7. Customize your rulesets.
8. Customize your preprocessor and decoder alerts.
9. Finally, customize your Shared Object Snort Rules.

Debian also has a Step 0 for a Debian-specific configuration as a separate configuration. Beginners, though, can ignore the Debian-specific choices.

Probably the most important place to begin is with the steps that involve rules. Intrusion detection is only as good as the rules it uses, and some rules may be out of date by the time you install.

Downloading Rules

Snort has three types of rules (Figure 4) for detection:

- Community rules are rules written by the community and available for free (Figure 5). These can be useful but may become obsolete, or may be developed some time after a new means of intrusion is known. Their quality depends on how conscientious the developers are.
- Registered rules are free rules, but they are only available to registered users. In order to download rules, you will need to first enter your personal code.
- Subscription rules are registered rules that are available only to paid subscribers. Subscribers can download them before a software release, which offers the very latest protection. The price differs for personal and business users.

Other rules are available on developer sites such as GitHub. If none of the available rules meet your needs, you can try writing your own. Online diagrams that parse the components of rules are widely available online [4].

For all rules, create the directory `/usr/local/etc/rules`, and add a symbolic link to `/usr/sbin/snort`. If you are using a Snort user or group, transfer ownership of these directories to them. Rules should be downloaded and uncompressed as root to `/usr/local/etc/rules` from *https://www.snort.org/downloads/#snort-3.0*.

Final Touches

Usually, you want Snort to listen to all traffic, which is known as promiscuous mode. To configure promiscuous mode, run the command:

```
ip link set NETWORK-INTERFACE promisc on
```

```
###################################################
# Step #1: Set the network variables.  For more information, see README.variables
###################################################

# Setup the network addresses you are protecting
#
# Note to Debian users: this value is overriden when starting
# up the Snort daemon through the init.d script by the
# value of DEBIAN_SNORT_HOME_NET s defined in the
# /etc/snort/snort.debian.conf configuration file
#
ipvar HOME_NET any

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any
# If HOME_NET is defined as something other than "any", alternative, you can
# use this definition if you do not want to detect attacks from your internal
# IP addresses:
#ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET
```

Figure 3: The `snort.conf` file is well-structured and heavily commented, making it easy to read.

```
# Copyright 2019-2022 Sourcefire, Inc. All Rights Reserved.
#
# This file contains rules that were created by Sourcefire, Inc. and other third parties
# (the "GPL Rules") that are distributed under the GNU General Public License (GPL),
# v2. The GPL Rules created by Sourcefire are owned by Sourcefire, Inc., and the GPL
# Rules not created by Sourcefire are owned by their respective owners. Please see
# the AUTHORS file included in the community package for a list of third party owners and their
# respective copyrights.
#
# This file does not contain any Sourcefire VRT Certified Rules; the VRT Certified
# Rules are distributed by Sourcefire separately under the VRT Certified Rules License
# Agreement (v 2.0)
#
#-----------------
# COMMUNITY RULES
#-----------------

# alert tcp $EXTERNAL_NET any -> $HOME_NET any ( msg:"APP-DETECT VNC server response"; flow:established; conten
t:"RFB 0",depth 5; content:".0",depth 2,offset 7; metadata:ruleset community; classtype:misc-activity; sid:560;
 rev:9; )
# alert udp $EXTERNAL_NET any -> $HOME_NET 5632 ( msg:"APP-DETECT PCAnywhere server response"; content:"ST",dep
th 2; metadata:ruleset community; classtype:misc-activity; sid:566; rev:10; )
# alert tcp $HOME_NET any -> $EXTERNAL_NET any ( msg:"APP-DETECT psyBNC access"; flow:to_client,established; co
ntent:"Welcome!psyBNC@lam3rz.de",fast_pattern,nocase; metadata:ruleset community; classtype:bad-unknown; sid:49
3; rev:11; )
```

Figure 4: Snort rules follow a concise structure of well-defined fields.

```
root@debian:~# wget https://www.snort.org/downloads/community/snort3-community-rules.tar.gz
--2022-03-26 13:53:49--  https://www.snort.org/downloads/community/snort3-community-rules.tar.gz
Resolving www.snort.org (www.snort.org)... 104.18.139.9, 104.18.138.9, 2606:4700::6812:8b09, ...
Connecting to www.snort.org (www.snort.org)|104.18.139.9|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://snort-org-site.s3.amazonaws.com/production/release_files/files/000/024/219/original/snort3-co
mmunity-rules.tar.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAU7AK5ITMGOEV4EFM%2F20220326%2Fus-eas
t-1%2Fs3%2Faws4_request&X-Amz-Date=20220326T205349Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature
=9d4078f968d36612b5c5e4e2b91df9d926ab7b1621b6ee64dbb0c6b9338e5e6c [following]
--2022-03-26 13:53:49--  https://snort-org-site.s3.amazonaws.com/production/release_files/files/000/024/219/ori
ginal/snort3-community-rules.tar.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAU7AK5ITMGOEV4EFM%2F20
220326%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20220326T205349Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&
X-Amz-Signature=9d4078f968d36612b5c5e4e2b91df9d926ab7b1621b6ee64dbb0c6b9338e5e6c
Resolving snort-org-site.s3.amazonaws.com (snort-org-site.s3.amazonaws.com)... 52.217.135.41
Connecting to snort-org-site.s3.amazonaws.com (snort-org-site.s3.amazonaws.com)|52.217.135.41|:443... connected
.
HTTP request sent, awaiting response... 200 OK
Length: 313713 (306K) [application/gzip]
Saving to: 'snort3-community-rules.tar.gz'

snort3-community-rules.tar. 100%[=========================================>] 306.36K   911KB/s    in 0.3s

2022-03-26 13:53:50 (911 KB/s) - 'snort3-community-rules.tar.gz' saved [313713/313713]

root@debian:~#
```

Figure 5: Community rules are freely available for the download. Other rulesets require registration and/or subscription.

```
          Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.1  <Build 1>
          Preprocessor Object: SF_POP  Version 1.0  <Build 1>
          Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
          Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
          Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
          Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
          Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
          Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
          Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
          Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
          Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
          Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
          Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
          Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
          Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
          Preprocessor Object: appid  Version 1.1  <Build 5>

Snort successfully validated the configuration!
Snort exiting
```

Figure 6: Snort confirms that the configuration is valid.

Listing 1: Editing /lib/systemd/system/snort.service

```
[Unit]
Description=Snort NIDS Daemon
After=syslog.target network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/snort -q -u
          snort -g snort -c
          /etc/snort/snort.conf -i eth0

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

```
Commencing packet processing (pid=5134)
03/28-14:07:43.282759  [**] [1:2657:8] WEB-MISC SSLv2 Client_Hello with pad Challenge Length overflow attempt [
**] [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.1.3:38860 -> 34.117.23
9.71:443
03/28-14:08:53.643216  [**] [1:1384:8] MISC UPnP malformed advertisement [**] [Classification: Misc Attack] [Pr
iority: 2] {UDP} 192.168.1.1:1900 -> 239.255.255.250:1900
```

Figure 7: A typical log entry: You may see false positives because Snort logs all packet activity.

At this point, you can run Snort in test mode to check that it is ready. As root, run:

```
snort -T -c /etc/snort/snort.conf
```

If configuration is successful, a copyright notice displays, followed by build notices and a message that the installation has been validated (Figure 6). If validation fails, the messages will most likely involve directories you need to add or rules that can be commented out before trying to validate again. As a final test, you can add a rule and then run Snort to see the results [5].

Running Snort

When Snort is configured and validated, you can run it with a single-use command:

```
snort -d -l /var/log/snort/ ➥
  -h IP-ADDRESS -A console ➥
  -c /etc/snort/snort.conf
```

Note that this command is a general purpose command for sniffing packets at the designated IP address, writing the log to standard output (`-A console`), as well as a file (`-l`), and using the listed configuration file (`-c`). The output starts with a copyright notice, followed by a description of the rules used and a listing of any rules that are obsolete or missing, and, finally, an ongoing list of events on the IP address.

Individual log entries begin with the date and the time, followed by the activity, its classification, and the source of the activity. An activity is given a priority number, generally on a scale of 1 (severe) to 4 (mild), although you can assign priorities as high as you wish when you write or modify a rule. If you are new to packet sniffing, you may be alarmed at the frequency and persistency of activities on a modern computer, so remember that Snort logs all packet activity, not just potentially suspicious activity, and is only as good as the installed rules. False positives will be common (Figure 7). Log entries are also written to `/var/log/snort`.

Most likely, though, you will want Snort to be running all the time, especially on a network. If your system uses systemd, create and edit as root the file `/lib/systemd/system/snort.service` and include the code provided in Listing 1 to the file.

Save the file, and then reload `systemctl` with:

```
systemctl daemon-reload
```

Snort will then be started at login or with the command:

```
systemctl start snort
```

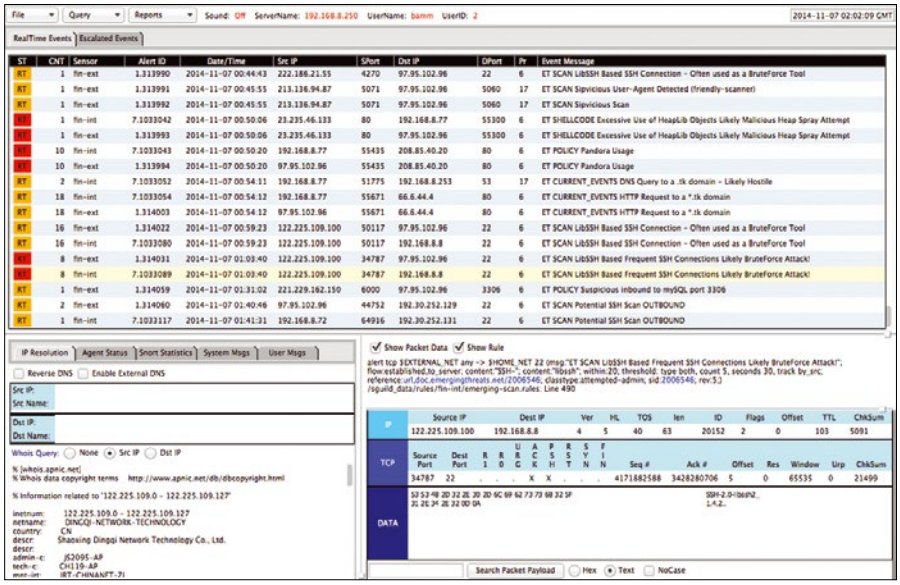


Figure 8: Sguil is a graphical app that relies heavily on Snort, as well as related tools.

Further Information

I have provided a bare bones outline for getting Snort ready to use. The Snort man page includes dozens of options. Trying to detail all the options here would be unrealistic. For further information, consult the latest Snort documentation [6] – which, despite the title displayed, is intended for version 3.0, not version 2.9.16. Snort is a mature piece of software that addresses complex issues, so be prepared to invest considerable time and experimentation if you want to get the most from it.

If you want to learn more about intrusion detection but prefer a desktop environment, spend some time with Sguil (Figure 8). Sguil relies heavily on Snort but also includes a number of related tools, including Barnyard2, `tcpdump`, and Wireshark. However, because Snort heavily comments its configuration files, it may be all you need to protect your system to your satisfaction. Where you go from here is up to you. ■■■

Info

[1] Snort: *https://www.snort.org/*

[2] "The greatest open source software of all time" by Doug Dineley, *InfoWorld*, August 17, 2009: *https://www.infoworld.com/article/2631146/the-greatest-open-source-software-of-all-time.html*

[3] Snort download: *https://www.snort.org/downloads*

[4] Rule structure: *https://cyvatar.ai/write-configure-snort-rules/*

[5] Test rule: *https://upcloud.com/community/tutorials/installing-snort-on-debian/*

[6] Documentation: *http://manual-snort-org.s3-website-us-east-1.amazonaws.com*

[7] Sguil: *http://bammv.github.io/sguil/index.html*

Solve Wordle puzzles with regular expressions

KING OF THE WORDLE

Five letters, one word, six tries – that's Wordle. You can solve any Wordle in just a few steps and gain practical experience using grep and regular expressions. *By Christoph Langner and Ian Travis*

You've probably come across strange posts on various social media platforms recently where users have shared images of what at first glance appears to be a very simple word game. Typically, you'll see a grid of five by six boxes colored either gray, yellow, or green populated by five-letter words that don't seem to have anything in common.

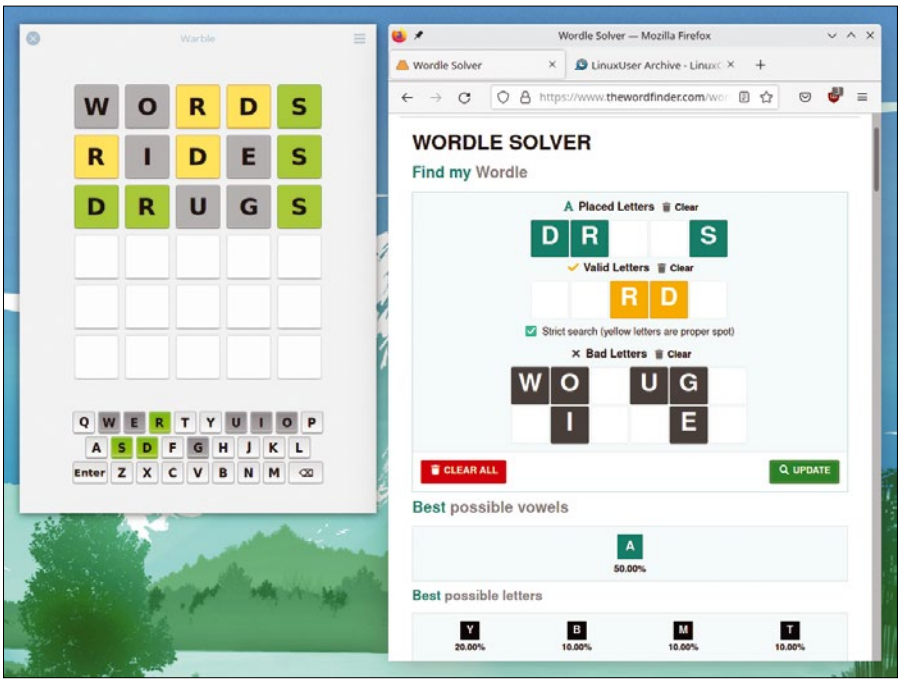


Figure 1: Warble imitates Wordle on Linux (left). Sites like The Word Finder help to solve the ubiquitous language puzzle (right).

If you've steered clear of the hype so far, this phenomenon goes by the name of Wordle [1]. Launched in October 2021, the free and currently ad-free Wordle was quickly acquired by The New York Times Company from US software developer Josh Wardle for a "low seven-figure sum" – rumors on the web claim the actual sum was $6 million [2].

To solve Wordle puzzles, you need an extensive vocabulary. If you want to make things a little easier, you can use a dictionary file and some regular expressions and create your own Wordle solver. From an IT point of view, Wordle offers an ideal practical example for getting started with `grep` and regular expressions.

Numerous Wordle Clones

All the hubbub about Wordle and the game's simple structure have already prompted numerous developers to program clones. In addition to various web-based imitations, there are also apps for mobile operating systems such as Android and iOS. There are even native Wordle imitators for Linux, such as

Lead Image © lightwise, 123RF.com

Listing 1: Wordle in the Terminal

```
01 $ wget https://github.com/ivanjermakov/wordle/releases/latest/download/wordle
02 $ chmod +x wordle
03 $ ./wordle
```

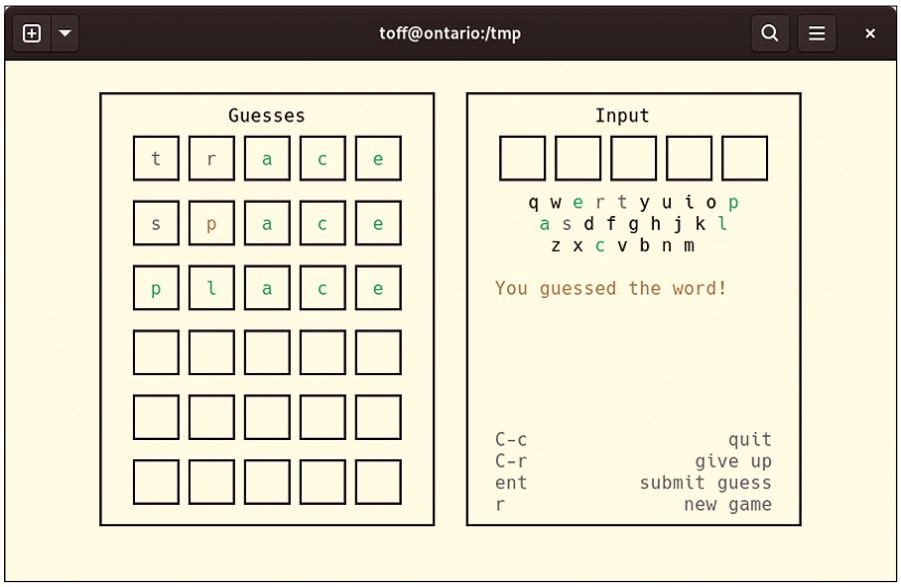


Figure 2: Wordle does not require complex graphics or animations, which explains why there are now Wordle games for the Linux terminal.

Warble [3]. Like the original, all of these applications use an English-language dictionary (Figure 1).

The easiest approach to implementing a Wordle game is probably via the command line because the game does not need complex graphics. For Linux, an open source Wordle [4] (Figure 2) can be downloaded with a few simple commands (Listing 1).

In addition to clones, sites such as The Word Finder [5] or Wordle Solver [6] can help you find a solution (the source code for Wordle Solver is available on GitHub [7]).

Basics of the Game

To solve a Wordle, you don't have to learn complicated rules. You have six attempts at each Wordle. In the first step, just enter an arbitrary five-letter word in the top line. On your first attempt, there are no clues; your first guess is completely random.

After pressing *Enter*, Wordle checks the input. If a field lights up green, the letter you entered is correct and is in the right place. If the field shows up yellow, the letter is part of the word, but currently in the wrong place. A letter with a dark gray background, on the other hand, is not part of the target word. With the help of these hints, you can narrow down the target word in the following lines until all of the letter fields light up green and you have solved the Wordle. You have a total of six attempts (Figure 3); there is always only one Wordle a day (see the "Second Chance" box).

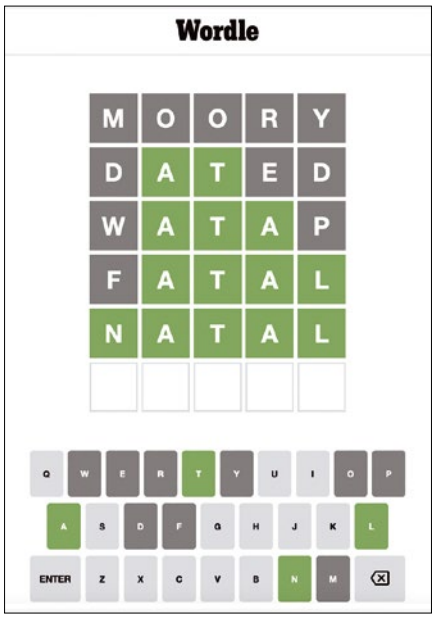


Figure 3: A first attempt: All the fields are still empty in the beginning. Enter a random word as the first guess.

Using Regexes

You can solve a Wordle using the Linux `grep` tool and a couple of simple regular expressions (regexes), which can be used to automatically process and filter texts and strings. If you have not worked with these tools previously, they are indispensable in the everyday life of software developers. Using these tools to solve a Wordle is a wonderful introduction to this topic with huge practical benefits. To solve a Wordle, we will work our way through the hints step by step, and at the end the computer will output the solution.

For this to work, however, you first need a dictionary. A dictionary may not exist on your Linux system, but check out `/usr/share/dict/` first – you might find a file that fits the bill. For a word list, you will use `words_alpha.txt`, which you can get from GitHub [8]. This lists contains over 370,000 words, including some pretty quirky ones, as you'll see later.

Preparing the Dictionary

Change to your home directory and surf to the following GitHub page to open the file: *https://github.com/dwyl/english-words/blob/master/words_alpha.txt*. Press *Download*, and you will see the start of a list of words. Now right click and select *Save Page As* to save the file in your home directory. If you're allergic to the GUI, you can use `wget` instead.

To make the work a little easier, you should convert the list to uppercase

Second Chance

Wordle usually only lets you play once a day. Instead of implementing this function via an account system, the providers simply set a cookie in the browser that identifies the user. If you are not worried about your statistics, you can simply delete the corresponding cookie and have another go at discovering the word of the day. To manage the cookies stored by the current website (e.g., in Chrome), click on the lock icon to the left of the address bar and select *Cookies* from the menu that then opens. Then select the cookies for the page and press *Remove*. Reload the page by pressing F5 to restart the game. However, this still only leaves you with one search term a day.

(all Wordle entries are uppercase) with `tr` (translate or transliterate) and store the results in a file named `wordle-caps.txt`:

```
$ tr '[:lower:]' '[:upper:]' < ➥
  words_alpha.txt > wordle-caps.txt
```

Your new dictionary file named `wordle-caps.txt` should have just over 370,000 words. Use `wc -l` (short for word count) to count the lines in the file:

```
wc -l wordle-caps.txt
370102 wordle-caps.txt
```

For Wordle, you only need the words with exactly five characters from this list. Again, you need the help of `grep`. Because all of the words are already in uppercase, you only need to output the five-letter words to a text file. The following `grep` command simply stores the five-letter words in a file named `wordle-complete.txt`:

```
grep -o -w "\w\{5\}" wordle-caps.txt > ➥
  wordle-complete.txt
```

The `-o` option tells `grep` to print the matching words, while `-w` tells `grep` that the search term is a regex. The regex string itself, `\w\{5\}` is equivalent to five continuous characters. Now run another line count as follows:

```
$ wc -l wordle-complete.txt
15918 wordle-complete.txt
```

Listing 2: Game 1, Round 1

```
$ shuf -n 5 wordle-complete.txt
FANGA
FRASS
SIAFU
MOORY
HALDU
```

This leaves you with nearly 16,000 words, which is more than enough to solve the Wordle of the day. Let's find out.

Grep the Wordle

While you only have to do the preliminary work once, keep the `wordle-complete.txt` file safe for later. To solve the wordle shown in Figure 3, you need to start with a completely random word from your Wordle dictionary. Initially, the game grid shown in Figure 3 is empty. You can run `shuf` to pick five random five-letter words from the file (Listing 2). If you are not happy with the selection, simply repeat the command.

Wow! Listing 2 resulted in an amazing collection of weird and wonderful words. In our example, we went for the word *MOORY*. When we entered it in Wordle, all the letter fields were gray – so at first glance, this wasn't a good guess. But now we know that the word we are looking for does not contain any of the letters from *MOORY*. This knowledge is actually helpful in our search for the solution.

The first command from Listing 3 filters out all words from our word list that contain the characters *M*, *O*, *R*, and *Y*. The `-v` switch (`---invert-match`) tells *grep* to invert the regex rule that follows. The command saves the results to the file `wordle1`, which "only" contains 5,362 words. From this list, you can output another five arbitrary words.

From the selection offered, we liked *DATED* best – well, it was the only word we understood, so hey ho. I wonder if Wordle will agree with us. Transferred to Wordle, the *A* in the second position and the *T* in the third position both light up green, so a pretty good guess. We now know that the second letter in the solution we are looking for is an *A* and the third letter is a *T*. The *D* and the *E* in *DATED* are shown in gray, so the letters do not appear in the solution.

Armed with this information, we can now narrow down the word list even further. The `grep` command from line 1 of Listing 4 combines all the conditions into a single call. The circumflex (`^`) means that the single statement should be inverted, similar to the `-v` switch. So the full regular expression `[^ED][A][T][^ED][^ED]` searches for a string of five letters. The first must not be *E* or *D*, the second must be an *A*, the third must be a *T*, and so on.

Our `wordle2` file now contains only 55 potential solutions. From this, we again output five random words (line 4). The dictionary defines a watap as a thread made of the string roots of various coniferous trees and used by Native Americans, so let's go with it. Again, Wordle isn't entirely happy with our guess. But we have a matching trio of *ATA* in the middle of our word, which results in more fodder for `grep`:

```
grep '[^WP][A][T][A][^WP]' wordle2 > ➥
  wordle3
```

Another call to `wc -l` tells us that `wordle3` only contains 10 words, so let's just `cat` the file and see what we get:

```
$ cat wordle3
BATAK
BATAN
CATAN
FATAL
KATAT
LATAH
LATAX
NATAL
SATAI
SATAN
```

Time for some guesswork: *FATAL* looks like a good choice, but, fatally (ouch),

Listing 3: Example 1, Attempt 2

```
$ grep -v '[MOORY]' wordle-complete.txt > wordle1
$ wc -l wordle1
5362 wordle1
$ shuf -n 5 wordle1
TUDEL
DATED
CEILE
ENCUP
DEFET
```

Listing 4: Example 1, Attempt 3

```
01 $ grep '[^ED][A][T][^ED][^ED]' wordle1 > wordle2
02 $ wc -l wordle2
03 55 wordle2
04 $ shuf -n 5 wordle2
05 HATCH
06 BATAN
07 PATTA
08 BATTS
09 WATAP
```

Listing 5: Game 2, Round 2

```
$ grep -P '[C][^LR][^LR][^LR][^LR]' wordle-complete.txt | grep A | grep O >
wordle1
$ wc -l wordle1
71 wordle1
$ shuf -n 5 wordle1
COCOA
CANOE
CHOCA
COMMA
COTTA
```

Wordle doesn't see things our way. Not to worry, though: The *L* in fifth position is marked in green, and the only remaining candidate is *NATAL*. Lo and behold, we finished the game in four steps, but only due to bit of bad luck at the end.

New Day, New Game

Using the same logic, we can tackle the next game (Figure 4). The hard work has already been done (i.e., we already have retrieved a dictionary and created an uppercase word list). Again, we need to start with an arbitrary word, extracting it from the complete Wordle dictionary:

```
$ shuf -n 5 wordle-complete.txt
CRAPS
DAMON
TAREQ
GEYAN
CLARO
```

This time we went for *CLARO*. Not a bad start: It looks like the *C* is in the right place already. The *A* can occur in the second, fourth, or fifth position, and the *O* can occur in the second, third, or fourth position. *L* and *R* do not occur at any position in the target word. The regex for this is `[C][^LR][^LR][^LR][^LR]`, but we also need to pipe the output through two further *greps*: After all, the word needs to contain an *A* and an *O*, too (Listing 5).

Now, I don't drink cocoa and prefer boats that are bigger than canoes, and I'm pretty sure that *CHOCA* isn't actually a word, so I'll go for the next word on the list *COMMA* – after all, I probably type hundreds of them a day. Success! We solved the Wordle in only two guesses.

Conclusions

Often the simplest ideas pay off. For the Wordle inventor, his idea translated to millions of dollars in hard cash. For Linux users, Wordle also offers a great opportunity to learn how to use `grep` and regular expressions: The easiest way to learn is with a concrete example. Plus, you can easily impress other Wordle users: Wordle? Easy as pie – you just have to know the right words.

Savvy programmers are unlikely to think the examples shown here are the most elegant approaches. There are definitely better regular expressions to tackle Wordle puzzles in a far more compact and efficient way, but this was not our objective in this article. We wanted to use expressions that were as understandable and comprehensible as possible. We'd love to hear about your approach to solving Wordle. Do you know a better word list for Wordle? Have you programmed the killer Wordle regex or a Wordle solver for the terminal? Let us know. ■■■

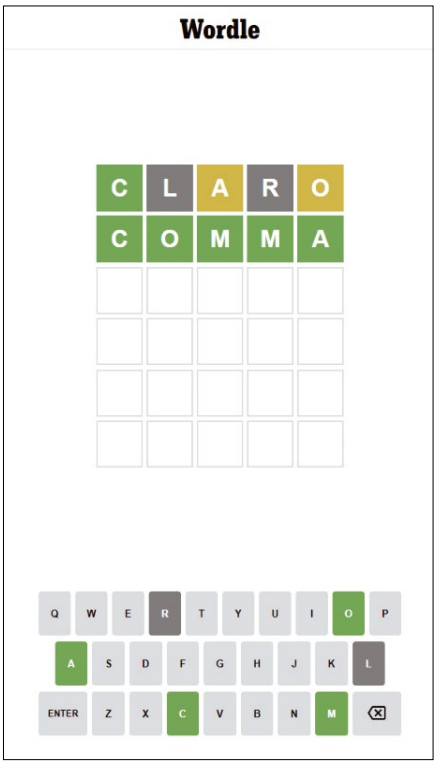


Figure 4: We solved the second example in just two steps with a bit of luck this time.

Info

[1] Wordle: *https://www.nytimes.com/games/wordle/index.html*

[2] "The New York Times Buys Wordle" by Marc Tracy, *The New York Times*, January 31, 2022: *https://www.nytimes.com/2022/01/31/business/media/new-york-times-wordle.html*

[3] Warble: *https://github.com/avojak/warble*

[4] Wordle for the Linux terminal: *https://github.com/ivanjermakov/wordle*

[5] The Word Finder: *https://www.thewordfinder.com/wordle-solver*

[6] Wordle Solver: *https://solvewordle.games*

[7] Source code for Wordle Solver: *https://github.com/jason-chao/wordle-solver*

[8] Word list: *https://github.com/dwyl/english-words/blob/master/words_alpha.txt*

Enterprise Resource Planning with BlueSeer

Desktop ERP

An open source ERP solution can save you thousands of dollars – in licensing fees as well as customization expenses. BlueSeer is an open source ERP solution that runs on the Linux desktop.

By Terry Vaughn

Most businesses in the manufacturing sector adopt some form of centralized software that records and retrieves data from various departments within the organization for purposes of metrics and financial reporting. This software is collectively called "Enterprise Resource Planning" (ERP). ERP software is ubiquitous throughout modern manufacturing industries and constitutes a total enterprise solution that governs most, if not all, aspects of daily business activity – from inventory control to financial accounting.

Today's ERPs have a much larger scope of application compared to their predecessors (see the box entitled "A Brief History of ERP"). Beyond accounting and materials management, current ERPs offer solutions for other departmental operations within an organization, such as sales and marketing, supply chain management, human resources management (HRM), customer relations management (CRM), asset management, and many other business operations. The last two decades have seen an explosion of new ERP software applications with new vendors marketing creative solutions to manage business operations and fill feature gaps of competitor offerings.

Most of the innovations today target options that go beyond on-premise installations and engage cloud-hosted platforms in the form of software as a service (SaaS), platform as a service (PaaS), or hybrids of cloud-based technology. As with other markets, the drive to offer more enhancements and fill feature gaps has led to consolidations of packages by acquisitions and mergers. This consolidation has created a best-of-breed class and fully established SAP and Oracle as the flagships of today's ERP vendors. ERPs are now marketed as off-the-shelf total solution packages that attempt to encompass every aspect of business operation. The biggest advantage is the implementation of a single-access application portal that removes departmental "silos" of operations.

In systems of the past, various departments in an organization would use the "swivel-chair" approach, entering and retrieving data between independent software applications. This approach can lead to inconsistencies in data, data duplication, and poor interdepartmental coordination. Job functions such as sales order entry and shipping are highly dependent on inventory availability – at both the manufacturing floor level and raw materials level – and the availability of cross-departmental data improves customer delivery efficiencies and decreases interdepartmental communication errors. Another distinct advantage is the reduction in redundant data entry processes between disparate systems. The storage of master data within a single back-end repository means other functional applications within the package can interact with the master tables without replication.

The use of ERPs is not without difficulties. The cost associated with purchasing and implementing a total ERP solution can be substantial. Commercial ERP applications are becoming increasingly too costly and too complex for many smaller businesses to afford and implement. Most high-end ERPs (SAP, Oracle) can easily run in the millions of dollars when the implementation project is complete. Even

Lead Image © melpomen, 123RF.com

A Brief History of ERP

The term ERP first appeared in an article published by the Gartner Group in April of 1990 that forecasted a vision for what manufacturing software was to become. The article suggested a larger definition and scope of software applicability that would encompass operations other than the core manufacturing concerns exhibited at the time, which were primarily financial accounting and inventory control. However, the actual concepts of an ERP system predate the acronym by three decades. The application of an electronic system for managing manufacturing processes date back to the 1960s with the advent and large-scale availability of mainframe systems introduced by IBM. Several manufacturing companies at the time applied these systems to manage inventory movement of raw components by incorporating calculations of reorder points and material usage within a structure called a Bill of Materials (BOM). These systems were labeled Inventory Control (IC) systems and were, in their early stages, simplistic and cumbersome, but developmental work during this period laid the foundation for future software development.

As IC systems were being introduced into the manufacturing mainstream, a parallel effort to standardize best-practices in materials management was also being cultivated based on the pioneering work of Joseph Orlicky, Oliver Wight, and George Plossl, whose research culminated in what was to become the basis for Materials Requirements Planning (MRP). MRP was a strategic approach to managing movement of inventory materials that was gaining popularity throughout the late sixties and seventies as a standard for best practice in manufacturing. The 1980s saw further refinement of MRP business strategy to include integration with other business operations, leading to the evolution of MRP II (with a renaming of the acronym to Materials Resource Planning). MRP II became the gold standard for operational excellence in manufacturing, and many budding software companies of the time touted MRP II as a core component of their software. These MRP II systems would later evolve into today's ERP. MRP II is still considered the key signature component of any manufacturing software that can legitimately claim the label ERP. Several software companies led the charge in incorporating best practices of MRP II and accounting into their software, and some of these would become the firebrands of today's ERP systems. SAP, MAPICS, JD Edwards, Baan, and Oracle all were early products in the ERP market. SAP and Oracle originally focused on financial accounting before expanding to encompass other operational concerns. MAPICS, derived from an IBM-developed predecessor in the late 1960s called PICS, was one of the first to adopt the principles of MRP II as a core feature. All of these early commercial ERP systems eventually adopted some form of MRP II features, as well as adding layers of other functional and operational concerns.

the cost of commercial mid-range ERPs can be overwhelming to some and effectively prohibitive to others. Training and implementation costs are also challenges when considering an ERP. Open source ERP systems that are freely available for usage and customization can provide some relief to the cost of commercial systems, particularly to the small or start-up manufacturing company.

Commercial ERP systems have gotten bigger, more versatile, and more expensive. The leading ERP vendors focus on customers with multiple locations and hundreds, if not thousands, of users. But what if you just need a simple system for a small company with a single data entry point or, at most, a few nodes on a local network? BlueSeer [1] is an open source ERP system that is designed to run on the Linux desktop. If you are looking for a practical solution that is easy on the budget, consider BlueSeer for ERP.

Open Source ERPs and BlueSeer

Open Source options in the ERP market have become more prevalent in recent years. Open source ERPs represent a strategic movement in reducing the overall cost of ownership for companies investing in ERP implementations. Not all ERPs that are open source are free however. Most open source ERP software brands have the usual marketing hooks of free trial downloads with expiration limitations and registration gimmicks, and only a handful are truly free. However, regardless of whether the software application is free or provisional, the open source nature of the source code does provide advantages that reduce the total cost of ownership.

The feasibility for customization of the software is an important part of the cost savings. Customization is practically inevitable, even with commercial ERPs, and customization of the ERP to match the actual process can be quite costly. Most commercial ERPs either do not allow client customization of the software or will require you to purchase the source code. Open source ERPs offer a better alternative by insuring the source code is readily available for end users who wish to better

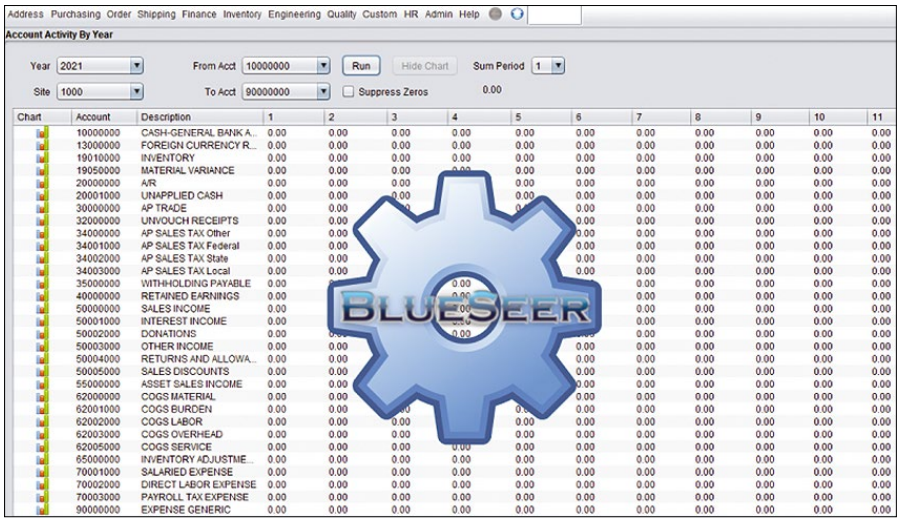


Figure 1: BlueSeer ERP is an open source business software package targeting desktop environments.

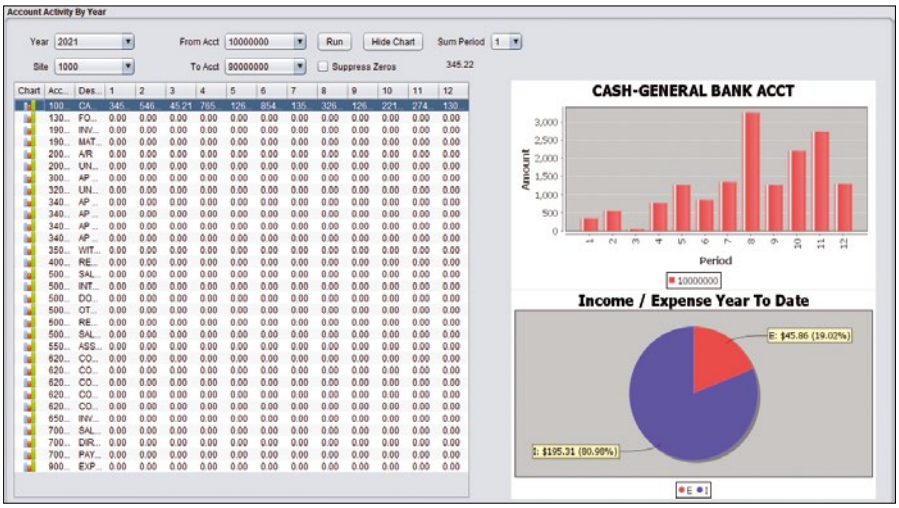


Figure 2: Financial reporting and metrics in BlueSeer.

manage the cost of customization. Furthermore, open source ERPs that primarily use free developmental toolsets, such as Java, Python, MySQL, and PostgreSQL, provide even greater savings because they support so many software libraries and are known to so many developers. Commercial ERPs, on the other hand, with their proprietary or highly specialized components, lead to substantially higher customization costs.

BlueSeer ERP (Figure 1) is specifically designed to confront the cost of customization and the overall cost of implementation and ownership. BlueSeer aims to be the first truly free desktop-based ERP package in the manufacturing community. The BlueSeer codebase is written entirely in Java, and the back-end database engines are freely available toolsets (SQLite and MySQL) that have a wealth of available documentation and software library resources. The development is consistent with the aforementioned primary pillars of ERP design and MRP II concepts, with financial accounting (Figure 2) and inventory control at the center of its core functionality.

BlueSeer has the usual functional areas that are expected from a traditional ERP, such as order entry, purchasing, scheduling, shop floor control, accounts receivable, accounts payable, and human resource management. In addition, BlueSeer offers non-traditional functionality that commercial ERP systems typically leave to bolt-on 3rd party applications. Electronic Data Interchange (EDI) mapping, label management, and time clock management are three functional areas that are often additional purchases if you work with a commercial ERP, but BlueSeer has integrated these modules within the core application.

BlueSeer's most distinguishing characteristic is arguably its deployment architecture. BlueSeer ERP is a desktop application that specifically targets the Linux (and Windows) desktop using the Java Swing GUI libraries. This approach is in contrast to most open source ERPs that deploy a web-based architecture, incurring the need for a separate web server and the installation complexities that accompany a web-based design. With BlueSeer's SQLite version (pre-bundled), the download and installation take a matter of minutes on either Linux or Windows. Implementing BlueSeer ERP on any of the free Linux distros (as either client or server) can be considered a serious contender for the lowest cost of ownership approach. The Linux OS has long been targeted as a back-end solution in web-based distributed ERP architectures (both commercial and open source). However, the Linux desktop has gone largely ignored as a viable deployment solution. BlueSeer was designed *especially* for deployment on the Linux desktop as a single-user ERP with the option to deploy multi-user client-server implementations through a MySQL/Linux server combination as necessary.

In either case, BlueSeer ERP, in combination with a Linux OS distro, is a considerable cost saver and particularly beneficial to the small or start-up money-conscious manufacturer. To demonstrate how easy it is to get started, I'll show you how to install BlueSeer ERP on the Linux desktop and provide step-by-step instructions for the simple business scenario of creating a sales order, shipping the sales order, and printing supporting documents such as packing slips and invoices.

Installation

You can install the BlueSeer ERP package on most Linux distributions. Two Linux download options available. One

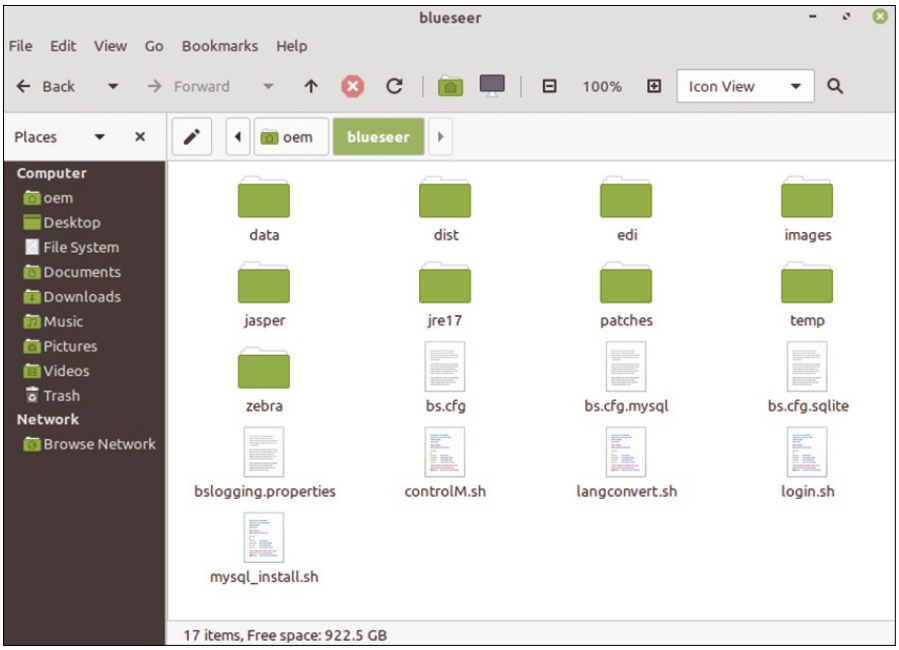


Figure 3: The newly installed BlueSeer application within the install directory.

is a `.deb` file for Debian-based distributions, and the other is a generic install (ZIP file) that is applicable to a wider variety of Linux distros. I will use the generic ZIP file installation, which is a relatively simple procedure. Both install packages come with a built-in Java Runtime Environment (JRE), so you do not have to be concerned about any specific JRE/JDK version preinstalled on your desktop. The first step is to download the software from the Github repository. Go to the following link and download the `blueseer.generic.Linux.v61.zip` file:

```
https://github.com/blueseerERP/
blueseer/releases/download/v6.1/
blueseer.generic.Linux.v61.zip
```

Open a shell prompt and type:

```
unzip blueseer.generic.
Linux.v61.zip -d
/home/user/blueseer
```

The preceding command will unzip the contents of the ZIP file and create a directory called `blueseer` in the `/home/user` directory. You can adjust `/home/user` to be whatever parent directory you desire. Once the contents are extracted, your installation of the standalone SQLite version of BlueSeer is complete. For a directory listing of the contents, see Figure 3.

You can execute the application by typing:

```
cd /home/user/blueseer
./login.sh
```

The application should start, and you will be immediately prompted for credentials. The default credentials are *admin* and *admin* for the user and password. On the initial execution, you will be prompted for a *country of origin* dropdown selection box. Choose your country of origin (which will effectively assign the default currency code for the application). Then restart the application. You now have a working instance of the BlueSeer ERP.

Business Simulation

With the application installed, you can proceed to run the business transaction simulation of creating an order and invoicing the order. You will first need to do a little configuration to get the customer and item master data created and configured.

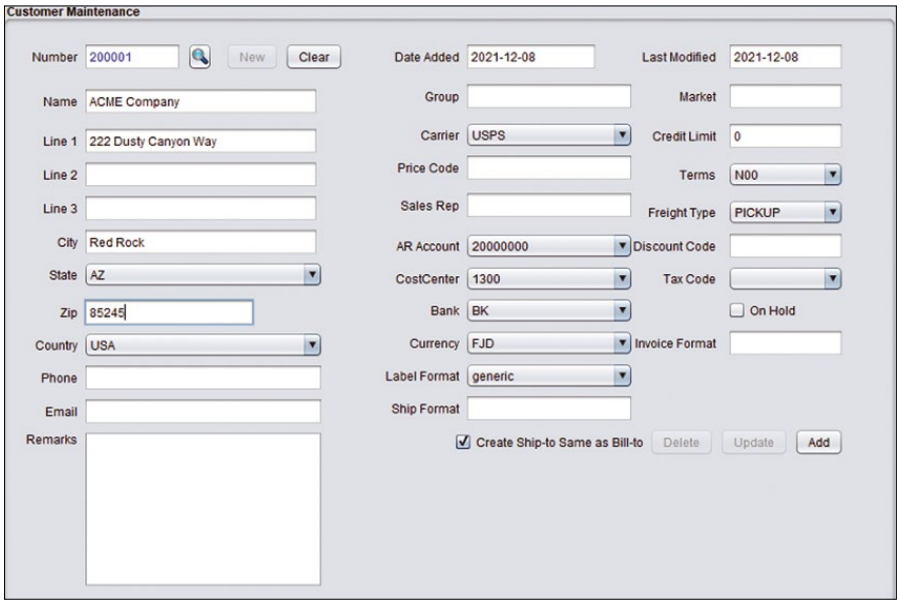


Figure 4: A screenshot of the Customer Maintenance menu, where customer address info is maintained along with associated data representing the customer.

Customer Master Record

To create a customer, click on the menu and select *Address* | *Customer Menu* | *Customer Maintenance*. You can optionally enter *cusm* in the navigation box on the main menubar. This will bring up the *Customer Maintenance* menu (Figure 4). Click *New* and enter the data for name and address. You can keep the defaults in the drop-down selection fields. The address information entered here will appear on any packing slips or invoice prints. Take note of the address code assigned to your new customer address record. Once you've entered the name and address, click the *Add* button to commit.

Item Master Record

To add an inventory item that will be shipped and invoiced in this business transaction scenario, click on *Inventory* | *Item Menu* | *Item Maintenance* or enter *item* in the navigation box. Click *New* and enter any description for the item you desire. You can keep most of the defaults in the other text fields. You will need to enter a price in the *Selling Price* text box. Once you've typed in the description and selling price, click *Add* to commit the item to the item master. Figure 5 shows the Item Master Data menu with a sample item added.

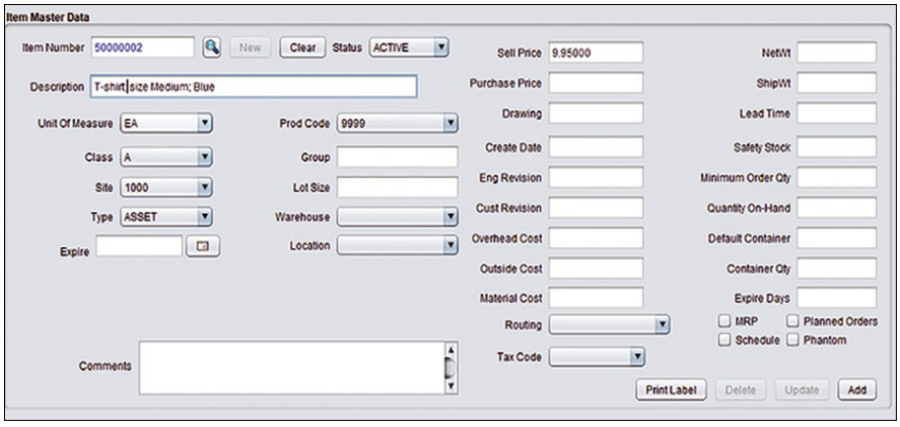


Figure 5: The Item Master Data menu contains information regarding the type and class of an item, as well as pricing, cost, and on-hand quantity per warehouse/location.

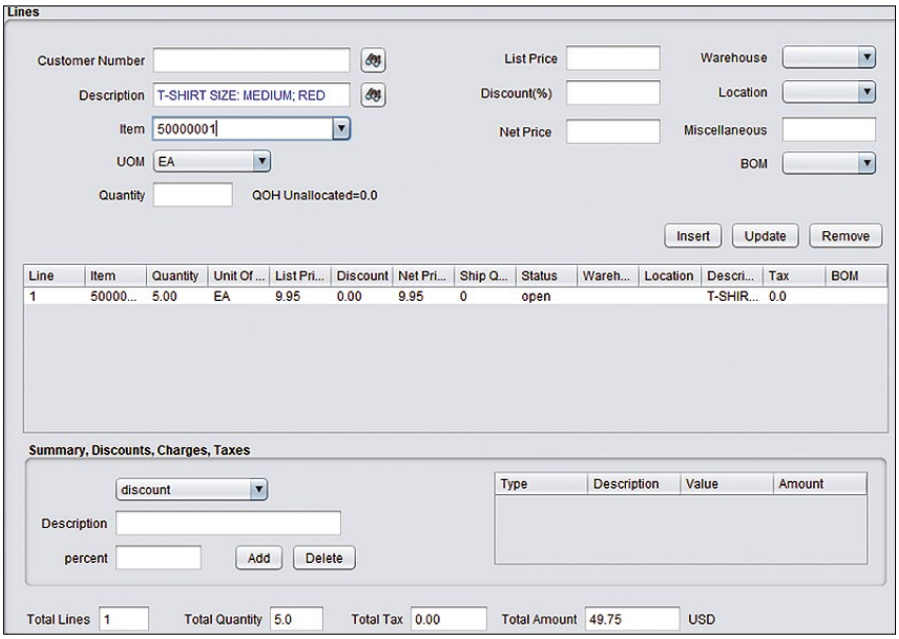


Figure 6: A record showing the quantity and price for an item to be shipped to a specific customer on a unique sales order. The total sales order price is shown at the bottom of the *Lines* tab.

Sales Order, Shipment, and Invoicing

Now that you have a customer and an item record, you can create the order, ship the product, and invoice the shipment. Typically, this is done with separation of duties by one or more responsible parties within the business (order entry department, shipping department, etc.), but BlueSeer provides a short-circuited menu option to perform all three functions in one menu, and I will use this option to demonstrate the order-to-invoice business scenario. Click on *Order* | *Order Maintenance* or enter *ordm* in the navigation text box. You will notice three tabs on this menu (*Main*, *Lines*, *Schedules*). Click *New* on the *Main* tab and choose the newly added customer code in the *Customer* drop-down selection box. You can keep all the other default values on this tab. Next, click the *Lines* tab. You should see that the item drop-down box is already selected with the item you created in the item master. You will also notice the price has been assigned from the selling price in the item master. All that is needed is to enter a quantity and the click the *add item* button. The item will be assigned to the details table in the *Lines* tab, along with a summation of the total at the bottom of this tab (Figure 6). Now, click the *Main* tab again and click *Add* to commit the order. This action will immediately insert an order record in the database and return you to the *Main* tab in the Order Maintenance menu with the newly created order retrieved.

You will notice the message *the order has not been shipped*, indicating that this order is new and has not been further processed. To ship and invoice the order as-is, simply click the *Invoice* button in the bottom left corner. Note that this invoicing step is performing several functions at once. By clicking the *Invoice* button, you are shipping the order and invoicing the shipment within the system. To print the invoice, click the *print invoice* button and a dialog frame will appear with your invoice print (Figure 7). You can either print or save as a PDF as necessary.

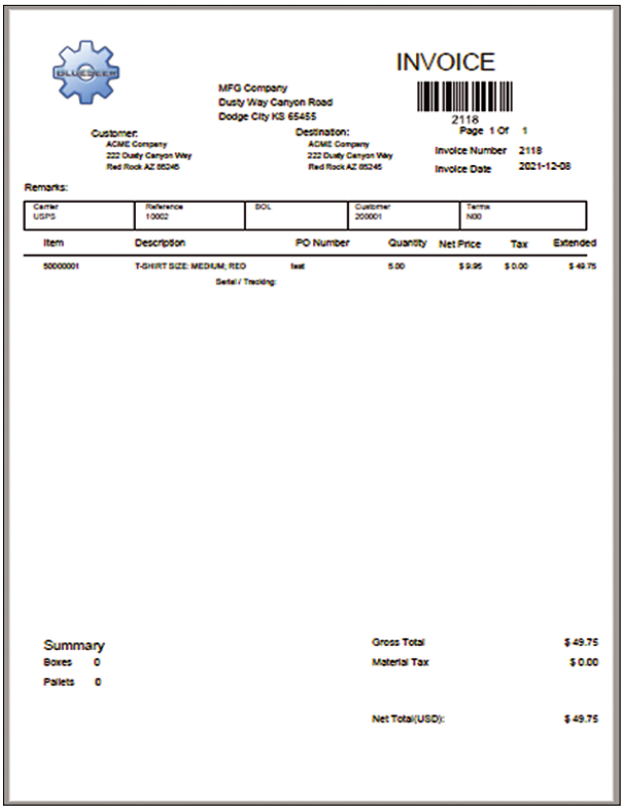


Figure 7: A screenshot of the actual invoice PDF created upon completion of the sales order. Relevant information, such as invoice number / date, bill-to party addresses, quantities, and summary prices are provided for accurate payment by the customer.

Conclusion

This article is just a sampling of the capabilities afforded an ERP instance on the Linux desktop with BlueSeer. Label printing, barcode scanning, and EDI communications are other tasks that are easily achieved on practically any Linux distribution. Given the open source nature of the BlueSeer application, you can easily extend the functionality for customization purposes. For more information, consult the BlueSeer ERP documentation available as a PDF at *www.blueseer.com*. ■■■

Info

[1] BlueSeer source code, documentation, and other install options: *https://github.com/BlueSeerERP*

Author

Terry Vaughn (*https://github.com/vaughnte*) is the creator of BlueSeer ERP. He is a strong supporter and advocate for Linux and open source solutions in the manufacturing sector. You can reach him at *terry@blueseer.com* or *https://www.linkedin.com/in/terry-vaughn-69337719a*.

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:

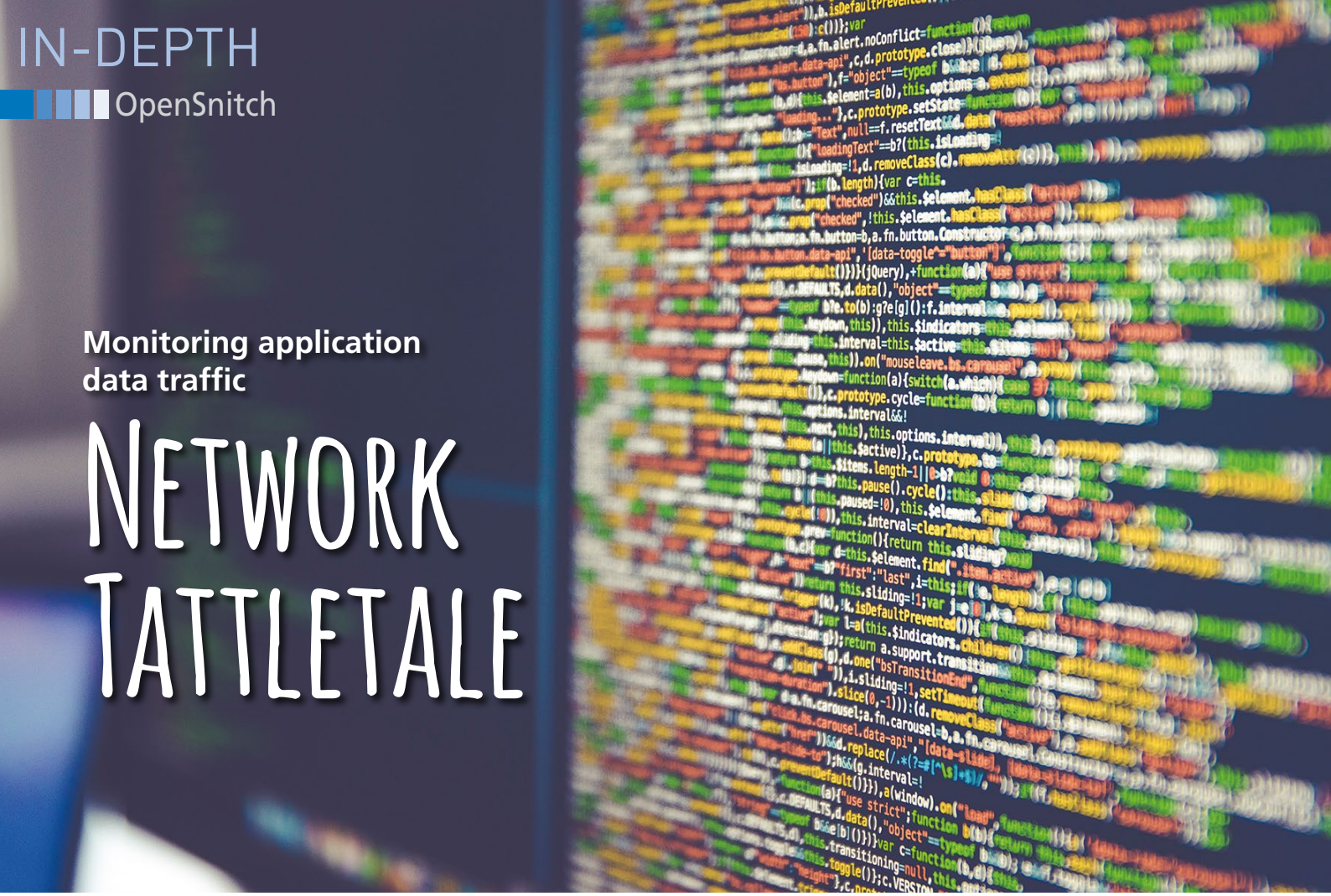
- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- Insightful news on crucial developments in the world of open source
- Cool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



Monitoring application data traffic

Network Tattletale

OpenSnitch, an application-based firewall, protects you from unwanted data leaks by letting you set customized rules for all your applications. *By Ferdinand Thommes*

An application opening a connection to the Internet is a normal procedure and typically completely legitimate, but there are programs – even open source applications – that like to phone home or track the user. On Linux, there is usually an opt-in step – you have to actively agree to the data collection. Often, the collected data relates to telemetry functions and gives the developers information about a user's interaction with their program. However, open source does not always protect you against being investigated. In Firefox, you have to actively opt out of sending telemetry stats if you do not want this to happen.

An application firewall can reveal what's going on behind the user's back. While conventional firewalls examine the data flow packets to and from the CPU, an application firewall takes an application-specific view when monitoring the outgoing data flow. (Do not confuse an application firewall with a web application firewall [1].) Examples of application firewalls include FirePrompt [2] for Linux and GlassWire [3] for Windows.

In this article, I'll take a closer look at an open source application firewall: OpenSnitch [4], a Python port of the proprietary Little Snitch [5] personal firewall for macOS. OpenSnitch development began about four years ago.

Snitch

With "snitch" in its name, you can tell much about how OpenSnitch works: Snitching is exactly what this firewall does. OpenSnitch analyzes applications' outgoing data traffic and exposes trackers and similar unpleasantries if configured accordingly, letting you intervene if necessary. In general, if an application tries to connect to the network, OpenSnitch stops it first and asks if you want to allow this to happen. You grant permission by defining a rule for the application.

Listing 1: Manually Activating OpenSnitch

```
# systemctl --now enable opensnitched
```

Figure 1: A pop-up window notifies you that Firefox is requesting a connection to update its version. You can allow, deny, or restrict access via various parameters at the bottom of the dialog box.

Photo by Markus Spiske on Unsplash

```
ft@aura:~/Downloads$ cd /etc/opensnitchd/rules
ft@aura:/etc/opensnitchd/rules$ ls
deny-until-restart-simple-opt-google-chrome-chrome.json
ft@aura:/etc/opensnitchd/rules$ cat deny-until-restart-simple-opt-google-chrome-chrome.json
{
  "created": "2022-01-01T12:31:13.642896881+01:00",
  "updated": "2022-01-01T12:31:13.642917414+01:00",
  "name": "deny-until-restart-simple-opt-google-chrome-chrome",
  "enabled": true,
  "precedence": false,
  "action": "allow",
  "duration": "always",
  "operator": {
    "type": "simple",
    "operand": "process.path",
    "sensitive": false,
    "data": "/opt/google/chrome/chrome",
    "list": []
  }
ft@aura:/etc/opensnitchd/rules$
```

Figure 2: OpenSnitch rules can be created and edited – not only in the graphical interface, but also in the form of a rules file in JSON format.

OpenSnitch is not typically found in the package archives of the popular distributions. On Arch Linux, up-to-date packages can be found in the Arch User Repository. MX Linux offers OpenSnitch, but only the outdated version 1.3.6. The current stable OpenSnitch v1.5.1 can be downloaded as a binary package from the project's GitHub page. Besides DEB and RPM packages [6] for 32- and 64-bit systems, you will also find the source code on GitHub if you want to build OpenSnitch yourself. Additional packages are available for the armhf and arm64 architectures.

Installation

I tested OpenSnitch v1.5.0-rc1, which is likely to be the stable version when this issue reaches the newsstand. I installed the packages for the daemon and the GUI on Debian Siduction ("sid") and Debian 11 ("bullseye").

On Siduction, there was a problem with some Python dependencies, but I was able to fix this by typing

```
sudo apt --fix-broken install
```

In Debian 11, the install completed without any hitches. Debian and its derivatives enable the OpenSnitch service automatically after installing the software. With other distributions you may have to do this manually (see Listing 1).

After the first launch, you'll find OpenSnitch in the system section of the control bar. Clicking on the OpenSnitch icon opens the application's main window. Right-clicking does the same thing after selecting *Statistics* but additionally lets you disable or close the firewall and gives you access to the help documentation.

Blockade

Initially, OpenSnitch blocks all connections to the outside world. If an application that does not have a rule tries to access the Internet, OpenSnitch pops up a dialog. Before you even get around to calling OpenSnitch from the system section, several successive pop-up windows will probably already be telling you that applications on your system are trying to contact hosts outside their own network (Figure 1).

If you grant permission for an application to contact the outside world in the pop-up window, this permission will be applied until the next restart by default. However, you can also make the new rule permanent or limit its validity to a specific period of time. Optionally, you can define whether the rule should apply to the running process only, to the targeted URL, or to the domain to be contacted.

OpenSnitch saves the rules you create in JSON format in the `/etc/opensnitch/rules` file, which you can also edit manually. If necessary, you can save as a CSV file the list of applications that try to contact external hosts, for example, and process the list further with external applications (Figure 2).

If you do not configure any settings, the window closes after 15 seconds and OpenSnitch blocks the connection by default. I found the time frame a bit short, so you might want to extend the grace period in the Preferences dialog, which you can access via the middle icon at the top of the application window (Figure 3). Under the *Pop-ups* tab, you can change the default action from *deny* to *allow*, set a shorter period instead of the default duration *until reboot*, or extend the duration to *always*, depending on your needs. If you missed a window (e.g., because it closed faster than you could react), you can edit the settings in the main window (Figure 4).

Firewall On or Off

In the main window's header bar, on the far right, you will find the play/pause

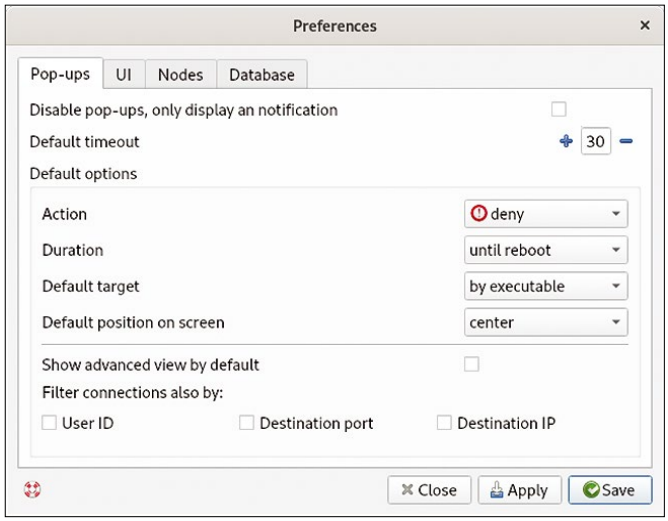


Figure 3: To extend the default timeout for pop-up windows for new rules, you can increase the value in the Preferences dialog, as well as adjust other default settings as needed.

button where you can turn the firewall on and off. This button is especially important initially because you need some time to define rules for all the applications that need to contact the outside world. You can use this button to break up the task into convenient chunks of time.

In the menubar below, you will find eight tabs. The *Events* tab lists all contacts to the outside world in real time (Figure 5). *Nodes* typically only lists one socket per device, from which the OpenSnitch GUI obtains the data for visualization. The default for this is `/tmp/osui.sock`.

The *Rules* tab, as expected, lists the application rules that have been created (Figure 6). The *Hosts* tab lists the remote sites that applications have attempted to contact and how often that occurred per host. The *Applications* tab lists the applications that tried to make contact and shows the frequency of those attempts.

The *Addresses* tab keeps track of the URLs contacted and the frequency of contact attempts. *Ports* does the same in terms of the ports on the contacted hosts, while the *Users* tab lists the users involved and records the number of contact attempts initiated by the users. From any of these tabs, you can edit entries that are released for editing by right-clicking on them.

To avoid losing your way when faced with many entries, you can also sort or filter the entries on the individual tabs. At the bottom of the window, you can see the number of connections during the current uptime and how many of them were rejected (dropped).

FAQs

OpenSnitch can manage virtually anything that connects to a host from a Linux system. For multi-user systems, the rules can also be defined individually for each user. According to the developers, however, OpenSnitch occasionally misses an app's connection attempt; the project wiki [7] on GitHub explains the possible reasons for this. However, I did not experience any such oversights in my test. An FAQ [8] answers frequently asked questions relating to the application firewall.

Once you have created all your rules, OpenSnitch runs unobtrusively in the background. A notification will only appear if you install a new app that makes an attempt to contact the outside world. If an app makes a conspicuous number of connections, you will want to harden the rule for that app by checking each process for an outgoing request or the domain contacted in each case, and then confirm or deny access.

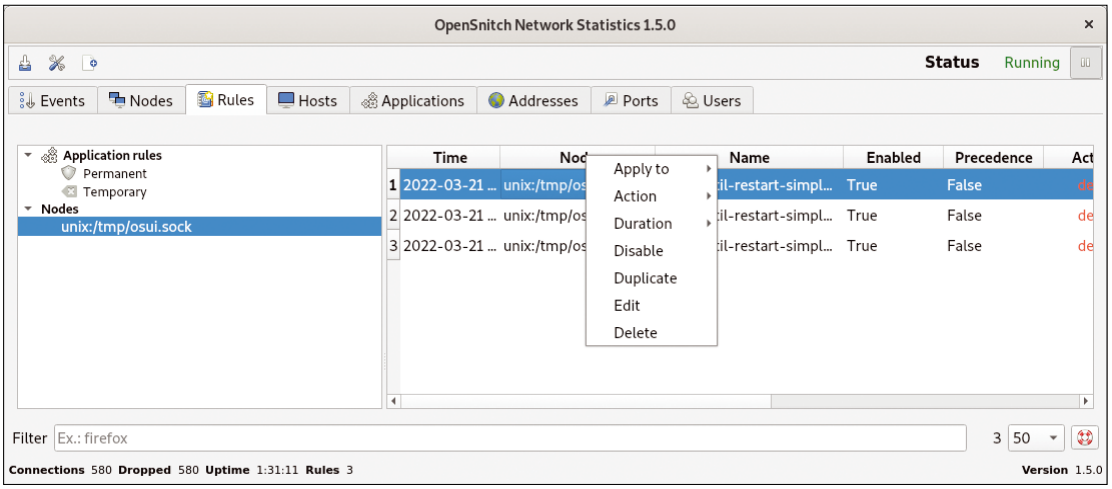


Figure 4: You can edit previously created rules via the main window by right-clicking and selecting the corresponding table entry.

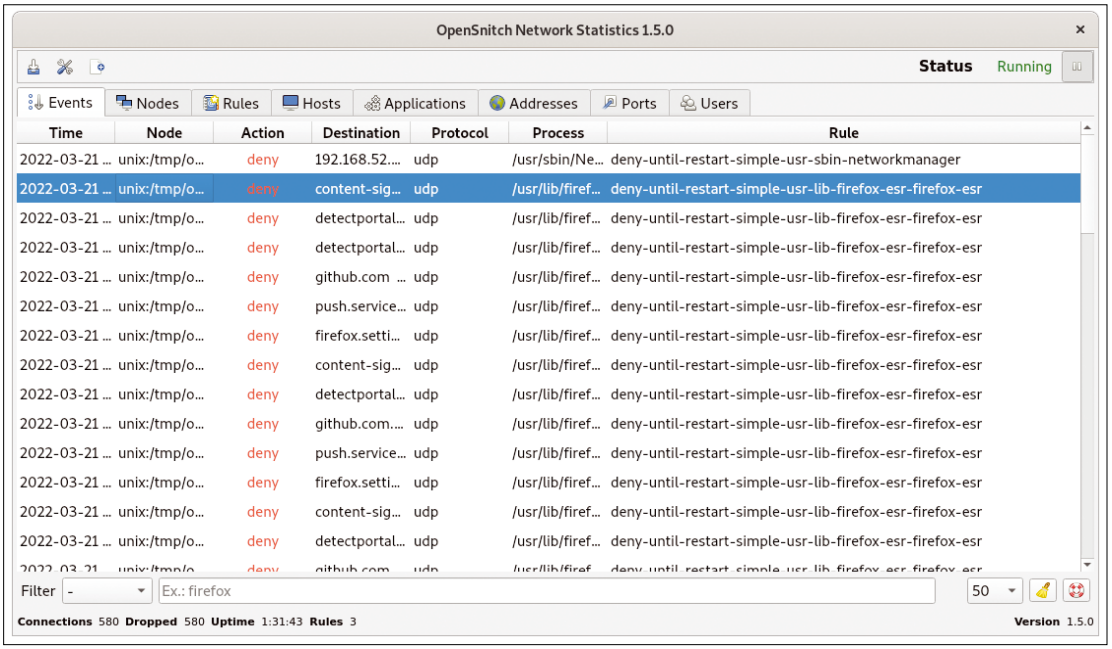


Figure 5: The *Events* tab lists all outside connections in real time. You can see connection attempts from Firefox to different IP addresses here.

Conclusions

While OpenSnitch is annoying at first, this means it is doing its job properly. You can temporarily avoid the many requests for rules by disabling the firewall and then defining more rules when it suits you. Getting started with OpenSnitch is comparatively easy thanks to the good documentation [9].

OpenSnitch is particularly

interesting for browser plugins, web apps, or third-party applications in general. It helps you keep a closer eye on these applications and make adjustments to rules as necessary. You will be surprised about what some apps try to do. In conclusion, OpenSnitch definitely improves the security of your system without asking too much of you beyond the initial setup. ■■■

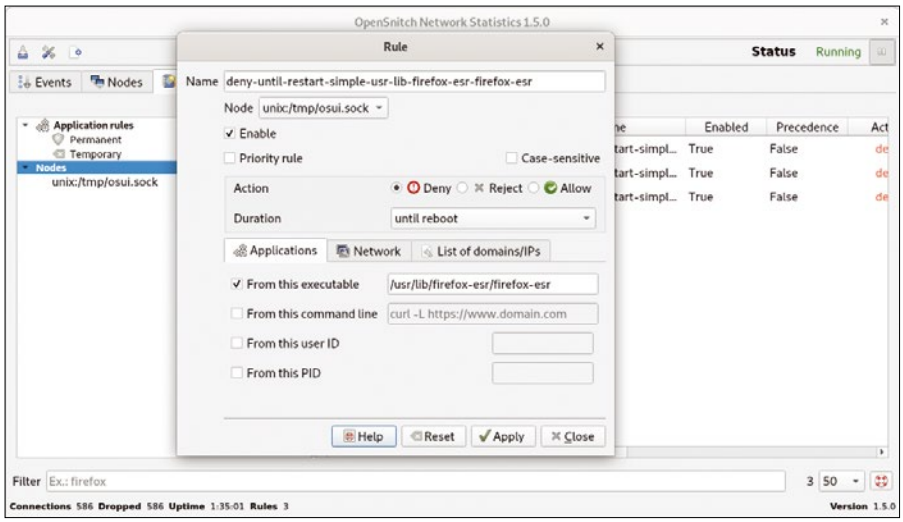


Figure 6: Because Firefox (as shown in Figure 5) wants to access many IPs, click on the *List of domains/IPs* tab in the Rule dialog to specify exactly what the program is allowed to do.

Info

[1] Web application firewall: *https://www.f5.com/services/resources/glossary/web-application-firewall*

[2] FirePrompt: *https://fireprompt.com*

[3] GlassWire: *https://www.glasswire.com*

[4] OpenSnitch: *https://github.com/evilsocket/opensnitch*

[5] Little Snitch: *https://www.obdev.at/products/littlesnitch*

[6] Download: *https://github.com/evilsocket/opensnitch/releases*

[7] Failure to intercept: *https://github.com/gustavo-iniguez-goya/opensnitch/wiki/Why-OpenSnitch-does-not-intercept-application-XXX*

[8] FAQ: *https://github.com/gustavo-iniguez-goya/opensnitch/wiki/FAQs*

[9] Documentation: *https://github.com/gustavo-iniguez-goya/opensnitch/wiki*

Author

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

The return of the tiling window manager

What's Old Is News Again

Tiling desktops have been experiencing a resurgence in popularity. Here are a few options that can help keep your desktop better organized. *By Bruce Byfield*

Tiling desktops are graphical environments in which windows open in a grid. They appeared early in Linux's history and have always had a few followers, especially among developers. For much of the past two decades, though, tiling desktops were ignored in the efforts to mimic Windows and macOS and to improve usability. However, in recent years, tiling desktops have become more popular, most likely because modern computing power means that more users are working with more windows open. Today, users can choose from a variety of tiling desktops. Some have been around for years, and others are more recent.

The idea behind tiling desktops is to reduce clutter on the desktop and make windows easier to find. By contrast, the standard or stacking desktop becomes less orderly with each open window. Most stacking desktops open windows in the upper left corner or some other default location. As users search through windows, the unwanted ones tend to be dragged aside, destroying what little order existed. In fact, the clutter is so great that many stacking desktops have a Show Desktop icon or widget. Others, such as Ubuntu's discarded Unity desktop, encourage users to open only one window at a time. Tiling desktops, on the other hand, arrange windows in a grid, making them easy to find. Should the windows become too numerous and too small for comfortable browsing, users can use virtual workspaces to add another grid. You can remove windows from the grid to increase their size and temporarily stack them on top of the grid. Another advantage of tiling desktops is that they can be easily navigated from the keyboard, although many also support a mouse.

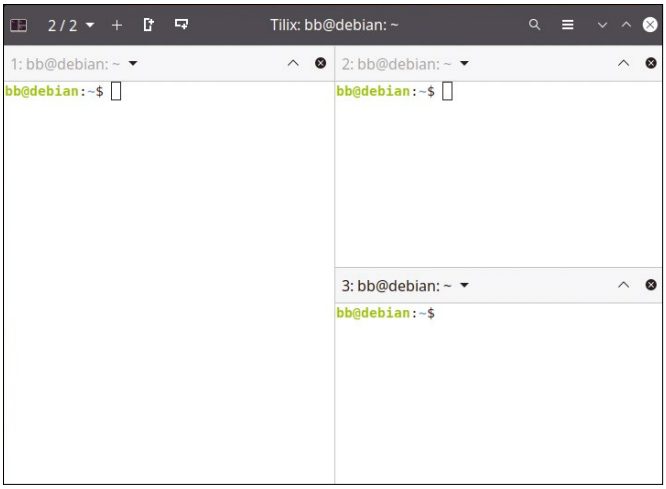


Figure 1: Terminal multiplexers such as Tilix are similar to tiling desktops.

You can get a feel for tiling desktops from terminal multiplexers such as GNU Screen, tmux, or Tilix (Figure 1), which turn a terminal into a tiling environment with multiple prompts. When you are ready to try an actual tiling desktop, here are some of the most popular choices. Many of these options can be installed as desktop environments in distributions, and a couple are also distributions in their own right.

ratpoison

Inspired by the terminal multiplexer GNU Screen, ratpoison [1] (Figure 2) is one of the earliest tiling desktops, and it still has a following today. Ratpoison is written in the C programming language (although some users may prefer StumpWM, which offers similar features and is written in Common Lisp). The name is a playful reference to the fact that ratpoison does not require a

Photo by Roman Kraft on Unsplash

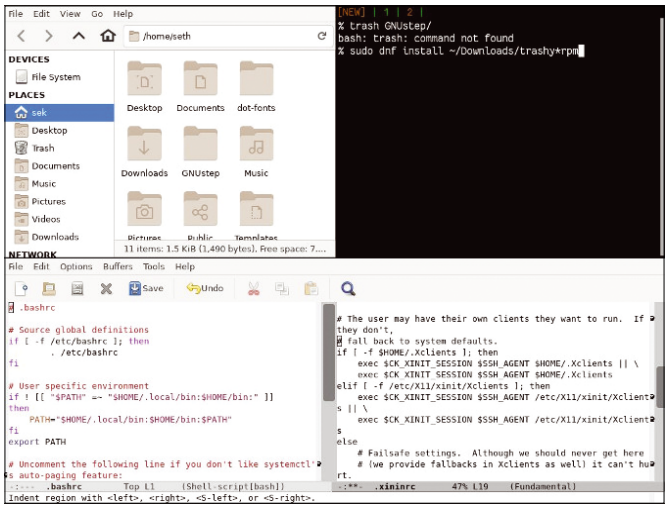


Figure 2: One of the earliest tiling desktops, ratpoison is also one of the most minimalist.

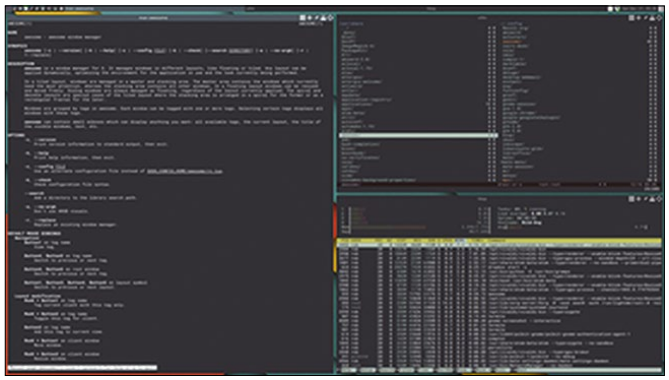


Figure 3: Awesome is known for its speed and configurability.

mouse, relying instead on keyboard commands that closely follow those of the once popular Emacs text editor. Moreover, although it can launch graphical applications, they are run from the command line. By default, ratpoison uses a single workspace, but it usually installs with a script called *rpws* to enable additional ones. If you value speed and configurability and have the patience to learn, you will find ratpoison ideal. If you value user-friendliness above all else, you should look at another tiling desktop instead.

awesome

Forked from the older window manager dwm, awesome (Figure 3) quickly went on to eclipse its original inspiration. Like many tiling window managers, awesome is designed for power users and those who enjoy tinkering. Written in C and Lua, awesome has all the expected features of tiling desktops, while enjoying a reputation for speed and configurability. In fact, an online search reveals page upon page dedicated to scripting for awesome, including the awesome project website [2] and Debian's awesome-dedicated wiki [3]. These websites should make customizing awesome much easier.

XMonad

Although written in Haskell, XMonad [4] (Figure 4) has much in common with awesome. Both were forked from dwm at roughly the same time and are highly configurable, with plenty of scripts available online that users can easily modify. However, XMonad is aimed at more general users, with clear documentation that includes an FAQ and a step-by-step guide to basic navigation. Just as important, XMonad offers many features such as separate layouts for workspaces, separate status bars for screens, and on-the-fly updating of the display when configuration files are updated. Many features introduced by XMonad later have been copied by other tiling managers.

The i3 Family

i3 (Figure 5), or i3-wm as it appears in some repositories, is one of the most widely used tiling desktops. You navigate i3 by keyboard using commands starting with the Mod key (Alt or Win) plus letters or numbers for switching tiles, resizing the current tile or making it full-screen, or opening a virtual workspace. Similarly, applications are started with Mod + d. Users can also add a new tile that is horizontal or vertical to the current one. Minor or temporary windows, such as pop-up dialog boxes, can be set to float so that they do not take up a lot of room. Should you have multiple screens, one workspace is set up on each screen by default. The result is a lot of flexibility, all of which can be customized in text-based configuration files, but the flexibility comes at the cost of a lot of learning.

Like many popular open source apps, i3 has spawned a number of forks. For instance, i3-gaps allows spacing between

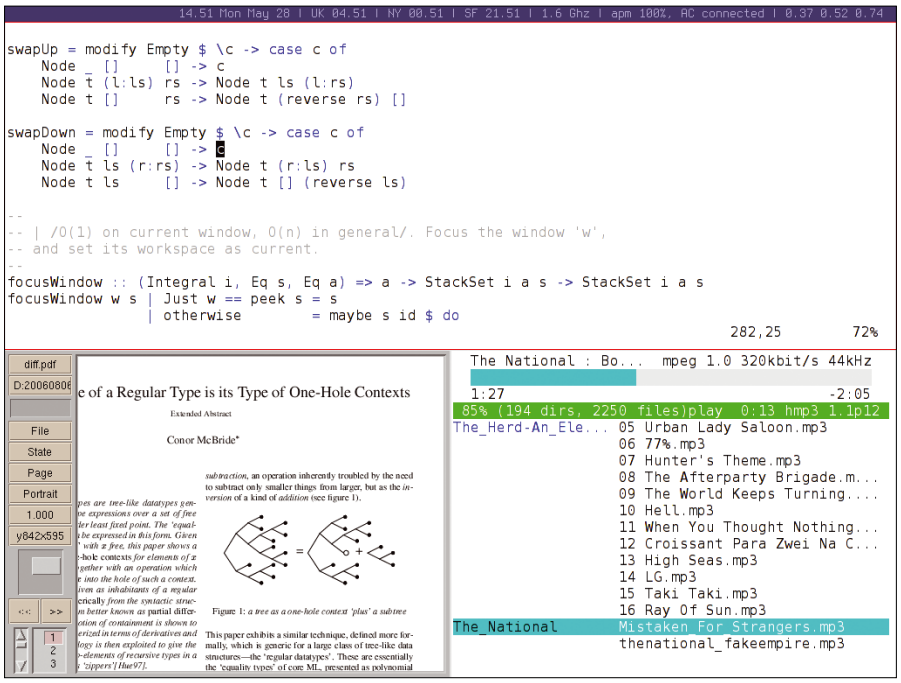


Figure 4: XMonad is known for its innovative approach to tiling as well as for its documentation.

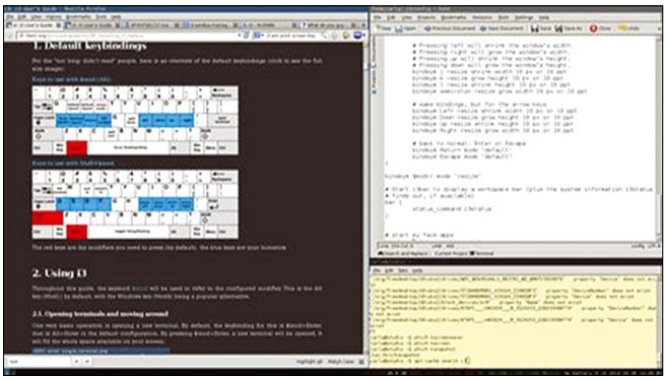


Figure 5: i3 is a popular tiling choice with numerous forks.

Figure 6: Regolith is available both as a package and an Ubuntu-derived distribution.

tiles, which makes for easier reading. More conveniently, Regolith [5] (Figure 6) places i3-gaps in both an Ubuntu-based distribution and a package for related distributions. Regolith features a handy cheat sheet of keystroke commands and can save snapshots of window layouts for future use. Yet another fork, Sway, runs on Wayland.

Pop!_OS

Designed by System76 for its workstations and laptops, Pop!_OS [6] (Figure 7) is probably the main reason for the recent renewed interest in tiling desktops. Although tiling is not enabled during installation, a button in the desktop's upper-right corner enables tiling and provides a drop-down list of features and settings (including a list of keyboard shortcuts), a setting for adjusting the gap between tiles, and can be conveniently displayed as a tile. Although tiles are arranged automatically, you can adjust each tile's size by dragging with the mouse. Best of all, Pop!_OS lets you discover how tiling works in a matter of minutes, making Pop!_OS by far the most user-friendly tiling desktop available today – so much so that tiling desktops are now less exclusive than they once were. Because the basic desktop is based on Gnome, the tiling features are already available for Ubuntu as an extension and will likely soon be ready for other distributions as well.

Hybrid Desktops

Unknown to many users, a few traditionally stacking desktops have offered their own limited form of tiling. Gnome, deepin, and KDE Plasma all provide an overview of virtual desktops and the windows open on them, although Gnome and Plasma remain primarily stacking desktops. Due to the increased interest in tiling, Gnome is currently developing PaperWM, a tiling extension, while KDE offers *kwin-tiling*, a script for its window manager.

A word should also be said about Plasma Activities, which are separate stacking desktops, each with its own desktop environment and icons. Activities can be organized by task or by some other criterion. For example, I have one Activity devoted to graphics, with icons for Gimp, Inkscape, Krita, and LibreOffice Draw, and another for the command line, which includes only icons for a terminal and screenshots. Other users, I have heard, have separate Activities for different projects, or for school or office. Such specialization minimizes clutter and also makes required resources easy to find and a single click away, providing similar benefits to tiling.

All these solutions require rethinking the way you work. On the one hand, if you are a tidy worker, opening one application at a time and closing the window when you are done, you may not find much benefit in tiling or the hybrid features of traditional stacking desktops. Similarly, if you prefer using a mouse, some tiling manager's reliance on keyboard commands may not suit you. On the other hand, if you regularly have 20 or more tabs open in your web browser, a tiling manager or a hybrid just might help you to organize your work and improve your efficiency. ■■■

Info

[1] Ratpoison: *https://www.nongnu.org/ratpoison/*
[2] Awesome Project Website: *https://awesomewm.org/*
[3] Debian Awesome Wiki: *https://wiki.debian.org/Awesome*
[4] XMonad: *https://xmonad.org/*
[5] Regolith: *https://regolith-linux.org/docs/getting-started/basics/*
[6] Pop!_OS: *https://pop.system76.com/*

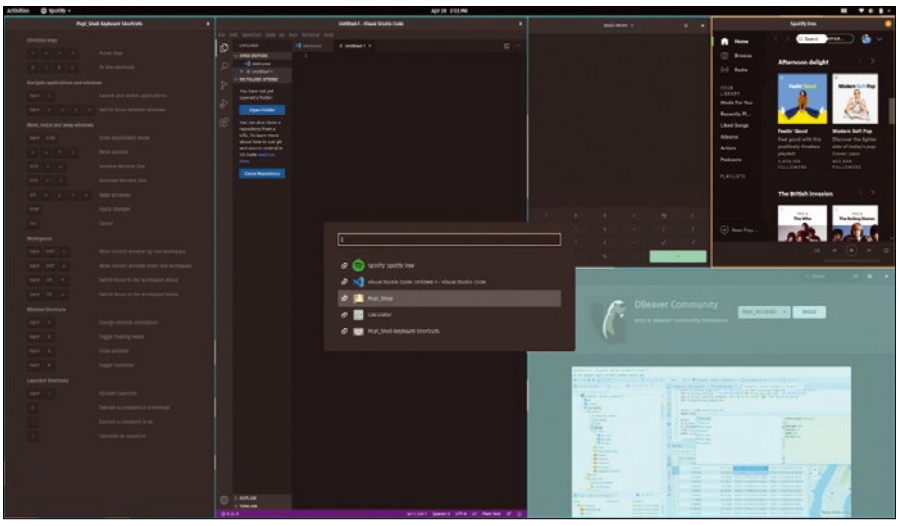


Figure 7: System76's Pop!_OS offers user-friendly, mouse-based tiling.

Get started with

SysAdmin
JOB HUB

Top jobs for IT professionals
who keep the world's
systems running

SysAdminJobHub.com

Talk to your Raspberry Pi in its native assembler language.

By Martin Mohr

Assembler programs run directly on the computer's hardware, which means they can reach nearly the maximum achievable speed of execution. Because assembler program code is very low level, writing the code is more complicated, but it is still the best choice for some tasks, especially on a computer such as the Raspberry Pi with its limited resources. Before you can start creating programs, however, you need to plumb the depths of the CPU and peripheral architecture.

Machine Code

To begin, it makes sense to clarify some terms. The CPU only understands machine code – zeros and ones or, more precisely, voltage levels that represent zeros and ones. Each command in machine code has a human-readable abbreviation that is easy to remember. These abbreviations are known as mnemonics and act as assembler commands. Assembler code is specific to a CPU architecture, which means that code for a Raspberry Pi (ARM) will not run on a PC (x86).

Programming in assembler on the Raspberry Pi can be approached in two ways: First, you can create an image in which you package the code and then boot the small-board computer (SBC) from that image to run the program. In other words, you degrade the Raspberry Pi to a microcontroller. With this method, the Pi runs without an operating system. Although you have full access to everything, you don't even get a shell.

The second way is to run the assembler program on the Raspberry Pi itself, which gives you the luxury of an operating system with everything that entails; however, you are limited in terms of direct access to the hardware. The second method was used for the example in this article.

Setup

A Raspberry Pi 3 with the current Raspberry Pi OS Lite provides the basis for my experiments. I will use Raspberry Pi Imager [1] to prepare the SD card, after which, I can boot from the card and get started right away, because all the tools needed for coding in assembler are included in the image. That said, an additional action provides more convenience and flexibility (see the "Activating SSH" box).

No Hello World

When you start working with a new programming language, the traditional approach is create a "Hello World" program; however, it takes a fair amount of assembler code and some understanding of strategies to generate even this simple output. Therefore, the first small assembler program only outputs the return

Lead Image © elenabsl, 123RF.com

Activating SSH

To remove the need for an additional monitor and keyboard, I recommend working on the Raspberry Pi over SSH. To get the service running correctly on first boot requires some minor intervention. To begin, create an empty `/boot/ssh` file on the SD card; the SSH daemon will then launch automatically at boot time.

If needed, redirect the output of the X server over SSH from the Raspberry Pi to the desktop PC with the `-X` option. This works best if you are also using Linux on the desktop computer. If your router supports local name resolution, use the

```
ssh -X pi@raspberrypi@local
```

command to open the connection. All graphical output from programs then end up on the desktop computer. If the local DNS does not work, use the IP address of the Raspberry Pi, which you can look up from the list of connected devices on the router.

code on the console, which indicates the status of a program on exiting. Bash stores this value in the `$?` variable, which you read with `echo $?`.

A return code of 0 means that the previously executed command ran without error; a value greater than 0 indicates an error. Listing 1 shows the example program `42.s`, so named because the return code value *42* is the result. (Note that the title of this article is 42 in binary-coded decimal (BCD) encoding, which represents each decimal section from 0 to 9 as 4 bits (i.e., half a byte – or a nibble, if you prefer.)

Assembler comprises relatively simple commands that do nothing more than move bytes back and forth, manipulate them, or react to a status bit, which makes it extremely important to document the code thoroughly and to use mnemonic identifiers where possible.

Labels are used in programming languages to mark points in the source code that serve as jump targets. The compiler exchanges the label for a physical memory address at build time, which clarifies the massive advantage of using labels: You do not need to calculate laboriously where in memory a particular command is located. Moreover, with each additional command you insert, all the addresses below it would move.

As in many other programming languages, you need to specify the starting point for a program in assembler. In Java and C the corresponding function is `main()`; in assembler you define the global label `main`. The label must precede the first line of code you want to execute, as shown in the assembler program in Listing 1.

The first line defines `main` globally so that the linker can find it. That label is then used in the second line, immediately followed by the first command, the `mov` command (short for move), which is used to move values (e.g., to store constant values in a register or to transfer the content of one register to another). To move values from registers into RAM or load them from RAM, you need the `str` (store register) and `ldr` (load register) commands instead. A register is a memory location on the CPU. An overview of the registers on the Raspberry Pi is shown in Table 1.

The program status register acts as the CPU's internal control register. The states of the individual bits tell you what results specific CPU register operations return. The commands for conditional jumps react to these bits. One well-known control bit is the zero bit (bit 30), which indicates that the value of an arithmetic logic unit (ALU) of a CPU, wherein all calculations and comparisons are performed, is *0*.

In the example in Listing 1, the `mov` command stores a value of 42 in the CPU register `r0`. When the program ends, the operating system reads the value of register `r0` and stores it in shell variable `$?`.

The `bx` command in the last line causes the CPU to continue the program at a different memory address – in this case, the address found in register `lr`. This register stores the address for program calls that the computer has to make after the program terminates.

The only question that now remains is how to generate an executable program from the assembler code. The workflows required to do this are very similar to the process of compiling C programs:

```
$ as -o 42.o 42.s
$ gcc 42.o
$ ./a.out
$ echo $?
42
```

The first line creates an object file from the assembler source code, which is then bound in the next line to the operating system to obtain an executable file. Unless you specify otherwise, this file is named `a.out`. You can execute the `a.out` program as usual. The final command shows that the program returns the value *42*.

Flash

Now that you have seen a simple assembler program, it's time for a more sophisticated project. The good old flash program is a good candidate; it simply flashes a single LED. The positive contact of the LED is connected to GPIO21 (BCM notation) with a 1kilohm series resistor. The connection is routed out on header pin 40, which is handy because you have a GND on header pin 39 right next door. This pin is connected to the negative terminal on the LED.

Of course, the program presents several challenges to those used to programming with high-level languages. The GPIOs of the Raspberry Pi support extremely versatile use, which means it is not easy to address them correctly in an assembler program. Forty-one registers,

Listing 1: 42.s

```
.global main  /* Entry point for the program */
main:
  mov r0, #42 /* Move value 42 to register r0 */
  bx lr       /* Return to calling program */
```

Table 1: ARM CPU Registers

Register	Mnemonic	Function
r0-r10	–	General registers without a special function
r11	*fp*	Frame pointer register
r12	*ip*	General register without a special function
r13	*sp*	Stack pointer
r14	*lr*	Link register
r15	*pc*	Program counter

each with a length of 32 bits, control the 54 GPIO pins of the chip installed on the Pi. (Not all of these GPIOs are accessible from the header; some of them are used internally by the Pi.) However, this still leaves you with a huge number of options for controlling the individual GPIOs. A detailed description of how the GPIOs work can be found in the BCM2835 peripherals data sheet [2] (pages 89-109).

Now that you know how to address the GPIOs, another problem raises its head. The Raspberry Pi's operating system prevents direct access to hardware resources. If you try to access the hardware directly, you will see a *Segmentation Fault* error message, which indicates that a memory protection violation occurred but gives you no additional clues as to where exactly the error occurred. Fortunately, the Raspberry Pi OS developers have provided a way to access the GPIOs without directly accessing the corresponding memory addresses. The operating system offers a driver to a special character file, `/dev/mem`, that is a mirror of main memory. A good description of this can be found on the Sonoma State University website [3].

Listing 2: flash.s

```
01 .globl main
02 .equ GPIOVAL, 0x200000 // Register value for the GPIO 21(BCM)
03 .equ GPFSEL2, 0x08     // Offset address for setting the GPIO mode
04 .equ GPIO_OUTPUT,0x08  // Define GPIO as an output
05 .equ GPFSET0, 0x1c     // Offset register set
06 .equ GPFCLR0, 0x28     // Offset register clear
07 .equ TIME, 0x8000000   // Wait value
08 main:
09 ldr r0,=gpiomem
10 ldr r1,=0x101002 // Open for reading and writing
11 mov r7, #5
12 svc #0
13 mov r4, r0
14 mov r0, #0
15 mov r1, #4096
16 mov r2, #3
17 mov r3, #1
18 mov r5, #0
19 mov r7, #192
20 svc #0
21 // r0 Contains the base address of the mapped GPIO memory area
22 ldr r1, =GPIO_OUTPUT // GPIO21
23 str r1, [r0,#GPFSEL2] // set as output
24 ldr r2, =TIME    // Wait in r2
25 ldr r1, =GPIOVAL // Register value in r1 for GPIO21
26 // Infinite loop
27 loop:
28   str r1, [r0,#GPFSET0] // Switch LED on
29   mov r10, #0           // Set r10 to 0
30   wait_on:              // Increment r10 to TIME
31     add r10, r10, #1
32     cmp r10, r2
33     bne wait_on
34   str r1, [r0,#GPFCLR0] // Switch LED off
35   mov r10, #0           // Set r10 to 0
36   wait_off: // Increment r10 to TIME
37     add r10, r10, #1
38     cmp r10, r2
39     bne wait_off
40 b loop
41
42 .data
43
44 gpiomem: .asciz "/dev/gpiomem"
```

The first block of the program in Listing 2 contains the definition of various constants with assigned values, which offers several advantages: On the one hand, it lets you use meaningful names in the program, and on the other hand, it lets you to load the registers with 32-bit values.

The `mov` command can only move values up to a certain size directly to registers. The next large block of instructions opens the `/dev/gpiomem` device and saves the base address of the mapped memory in register `r0`.

Supervisor calls are used in the `svc` block; put simply, these are something like calls to existing operating system programs. (The "Supervisor Calls" box provides additional information.) Initially, it is important that you have the option of accessing the GPIO from the address in `r0`.

The flash program starts in line 22 by first setting the mode for GPIO21 to output. It then loads the wait value into register `r2` (line 24), and line 25 stores the bit combination for switching GPIO21 on and off in register `r1`.

The GPIO registers work in a fairly simple way, and each of them has a specific task (set GPIO, clear GPIO, enable pullup, etc.). Each single bit in the registers corresponds to a GPIO pin. If you set the bit corresponding to GPIO21 in register `GPFCLR0` (register for clearing GPIOs), the GPIO drops off to 0, which is why the program loads the combination for GPIO21 into register `r1`.

Now all you need to do later is alternately move the `r0` register to the `GPFSET0` and `GPFCLR0` GPIO registers (lines 28 and 34). The command means: Store the contents of `r1` at the memory address that results from adding the contents of `r0` to the constant from `GPFSET0` and `GPFCLR0`, respectively. The two wait loops

Supervisor Calls

Supervisor calls (syscalls) are functions provided by the operating system to perform certain tasks. Each syscall has its own number with which it is called. On the Raspberry Pi, the numbers with the matching names can be output on the terminal with the

```
cat /usr/include/arm-linux-gnueabihf/asm/unistd-common.h
```

command. You need the `svc #0` command to execute a syscall in assembler, but you can also execute syscalls from high-level languages. When doing so, the number of the syscall must be in the `r7` register. Depending on the syscall, the registers `r0` to `r6` contain the associated parameters. The return value of the call always ends up in register `r0`. You can access the documentation for the individual syscalls in a terminal window by typing:

```
man 2 <name>
```

The `2` here indicates that you only want to search section 2 of the documentation.

increment the value in `r0` until it reaches the value of `TIME` (`r2`).

This procedure of creating a wait is not very smart, because one CPU core is counting continuously. You can use the `top` command to look at the CPU usage when the program is running. A CPU time-optimized program would use timers and interrupts, but it would be considerably more complicated in that case.

Finally, line 40 contains an unconditional jump command that jumps back to the `loop` label, thus running the program for all eternity.

Where To Go Next

Now that you are knee-deep in assembler programming, you might want to look into the subject in detail. I would recommend a tutorial, such as the one you can find on the Think in Geek website [4]. It explains the basics from A to Z, with many useful tips.

You can easily enter simple examples, like the ones from this article, in an editor such as Nano. However, if you are working on more complex projects, you will want a more powerful editor to make your life easier.

Several possible ways to upload files to the Raspberry Pi are at your disposal. I mounted the Raspberry Pi over SSH in the Ubuntu file manager. This simple approach usually works fine on a LAN. With more complex projects it doesn't make much sense to build all the files manually. Even the old-fashioned `make` will save you time and overhead.

Before you start to implement a function, always have a look at the list of syscalls. In many cases you will find something suitable. When using syscalls, you can assume that they do not contain any errors, which is worth its weight in gold, especially in assembler programming.

Conclusions

In this article I was only able to scratch the surface of assembler programming, and many details remain open. Getting started with this programming language is not difficult, and the individual commands are not complicated. Once you have looked into the CPU architecture, the meaning of assembler code is fairly easy to understand.

The tricky part begins as soon as you start using assembler to solve problems that are typically tackled in high-level languages. Even a small program will quickly grow to a few hundred commands. The advantages are the minimal code footprint and maximum execution speed, if programmed correctly. ■■■

Info

[1] Raspberry Pi Imager: *https://www.raspberrypi.org/software/*

[2] BCM2835 data sheet: *https://www.raspberrypi.org/app/uploads/2012/02/BCM2835-ARM-Peripherals.pdf*

[3] GPIO access by gpiomem: *https://bob.cs.sonoma.edu/IntroCompOrg-RPi/sec-gpio-mem.html*

[4] ARM assembler tutorial: *https://thinkingeek.com/arm-assembler-raspberry-pi/*

Author

Martin Mohr has experienced the complete development of modern computer technology. After finishing university, he mainly developed Java applications. The Raspberry Pi woke his old passion for electronics.

MakerSpace

Integrate hardware components with pluggable systems

Plugged In

Ecosystems with pluggable Raspberry Pi modules, sensors, and displays are a great choice if you don't want to – or can't – solder but still want to extend your hardware.

By Bernhard Bablok

If you use your Raspberry Pi to control sensors or displays, you will frequently have to deal with wiring problems or resort to using a soldering iron. Plug-and-play systems such as Adafruit STEMMA-QT [1], Seeed Grove [2], SparkFun Qwiic [3], and DFRobot Gravity [4] provide connection systems for electronic components. A system recently introduced by Tinkerforge [5] can also be plugged together, but it plays in a different league, because it comes with an additional microcontroller for management tasks on each module.

The systems I will discuss in this article offer special cable connections on the devices (Figure 1). Common to all systems are reverse-polarity-proof connectors and symmetrical cables. In other words, the cable has connectors that look identical on both sides, but they are not the same internally and only fit one way. Cables also are color coded, but this is not significant from a technical point of view.

Advantages and Disadvantages

Even though you have four manufacturers with four different pluggable systems, you do not have to make a decision and be tied to one ecosystem. However, caveat emptor still applies; I will look at the details a little later.

First, I'll look at the obvious advantages. Of great importance is the stable

Figure 1: Connection types in the Adafruit (STEMMA/STEMMA-QT, left), SparkFun (Qwiic, center), and Seeed (Grove, right) systems.

Lead Image © dmitr1, 123RF.com

connection: The pluggable systems are self-locking, which is helpful not only during prototyping, but later when building solutions, as well. Even the classic jumper cables with Dupont connectors can sometimes come loose inside a housing, especially when exposed to vibrations or shocks.

The standardized connections offer another advantage. The widely used inter-integrated circuit (I^2C) is standardized on the bus, but each breakout comes with its own sequence of voltage, ground, SDA, and SCL, usually because of the arrangement of the pins on the installed chip. The connector systems put an end to this setup; as a user, however, this convenience comes at the price of a larger breakout and more complicated cable routing.

Cable Only?

On closer inspection, the different systems are surprisingly compatible. The oldest system, Grove (2010), uses a proprietary four-pin connector with a pin spacing of 2mm. Adafruit's STEMMA connectors (2014), on the other hand, rely on four- or three-pin JST-PH connectors, a widely used system. However, Japanese solderless terminals (JST, although they were developed in Germany) is not a standard. Adafruit only rarely uses the original STEMMA; sensors in particular are only available with STEMMA-QT, which I'll get back to later.

The four-pin STEMMA connector is intended for I^2C and the three-pin variant for pulse-width modulation (PWM)/analog/digital connections. The four-pin STEMMA is in principle cable-compatible with Grove thanks to an identical pin-out sequence, as well as with Gravity from DFRobot through the three-pin connection. STEMMA connectors – contrary to Adafruit's claim – do not match up with Grove connectors. Adapter cables can be made very easily, though.

To use the right terms, the plugs are the things with pins that sit on the components, whereas the connector on the cable is a socket, although this terminology is counterintuitive and not what people actually call them; thankfully, what people mean is usually clear from the context.

The small STEMMA variant, known as STEMMA-QT (2017), uses the JST-SH connector, with a pin spacing of only 1mm. QT (cutie) is an intentional play on words. The advantage of STEMMA-QT is that it takes up less space on the breakouts, which is offset by the disadvantage that the smaller connectors cannot handle as many plugging operations. JST does not publish an exact number, but it will be somewhere in the double-digit range.

Adafruit deliberately chose the STEMMA-QT connector format to be compatible with SparkFun's Qwiic (2017), both of which are limited to I^2C only.

At the cable level, the systems from the four manufacturers form two groups: STEMMA, Grove, and Gravity use large connectors with 2mm spacing, and STEMMA-QT and Qwiic use JST-SH connectors with 1mm spacing. Adapter cables or special breakouts with matching connectors convert between the individual systems (Figure 2).

Fine Differences

Mechanical compatibility is not everything. For example, sensor and breakout components will only work with 3.3V or 5V. A normal BME280 sensor breakout (3.3V) will therefore not work on an Arduino Uno (5V) without a voltage converter; therefore, STEMMA-, Grove-, and Gravity-compatible controllers or devices come with voltage converters so that, as a user, you don't have to worry about the voltage. Qwiic only supports the 3.3V variant, so Qwiic sensors will not work with 5V microcontrollers.

Both Grove and Gravity use the four-pin connector for more than just I^2C. For example, Grove has a double push button (Figure 3) on offer, but because STEMMA only supports I^2C, there is little point in plugging a button like this into a STEMMA HAT (a hardware attached on top add-on board). The same is true for Gravity devices with a universal asynchronous receiver/transmitter (UART) connection.

All in all, however, the differences between the systems are not so major that hobbyists have to commit for all time. You can pick and choose components to suit needs and availability. Many I^2C components are on offer simply because of SparkFun, with its more than 150 components. Apart from I^2C,

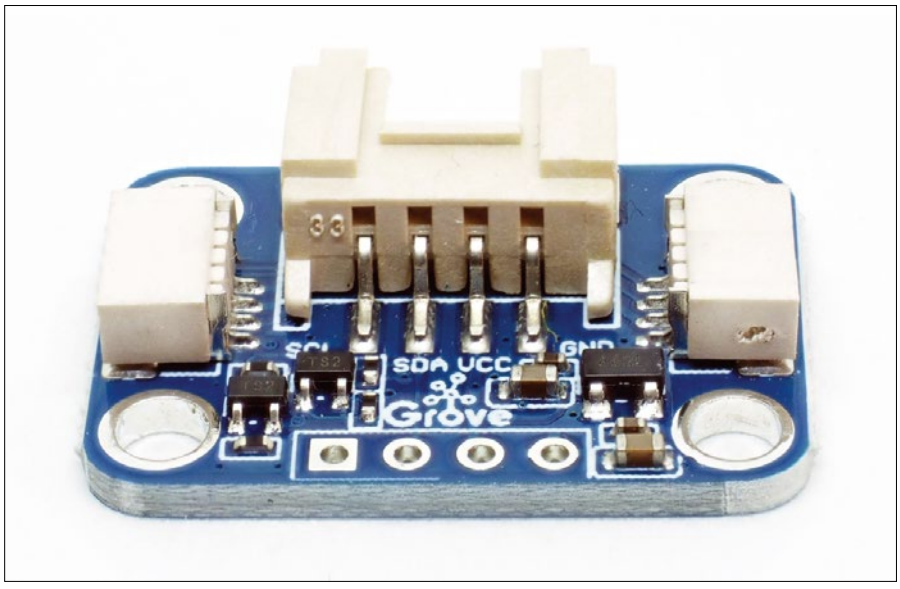


Figure 2: A Grove hub with a Qwiic connector.

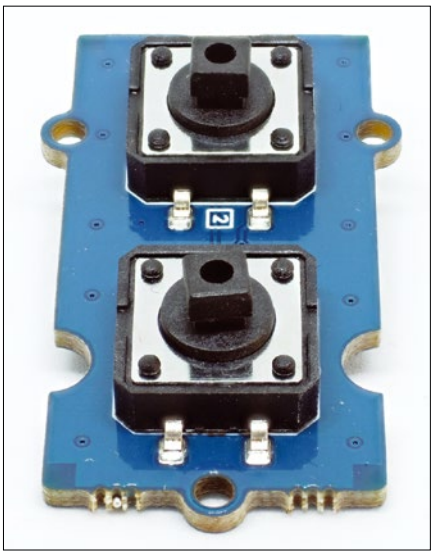


Figure 3: A double push button from the Seeed Grove portfolio.

however, things look a little more sparse, because Adafruit does not offer much in terms of three-pin products, and DFRobot's Gravity components are difficult to get in Germany – which is why I chose Grove.

Application Examples

In a typical application scenario, a small-board computer (SBC) or microcontroller controls several sensors and outputs the data on a display. System providers offer suitable HATs for this purpose – or shields for the Arduino world.

The SparkFun Qwiic HAT has four I²C connections, although your devices must be on different addresses [6]. The Grove HAT (Figure 4), on the other hand, offers different connections – an I²C, PWM, and UART each – as well as two digital connections with two connected GPIOs each and three analog-to-digital converters (ADCs) with two channels each. The HAT has its own microcontroller for analog input and is available for $10 (~EUR10) – and in a variant with even more connections for the large Raspberry Pi models. The ADC connector alone is worth the money. For a Pico, it is best to go for the Maker Pi Pico by Cytron with its six sockets [7].

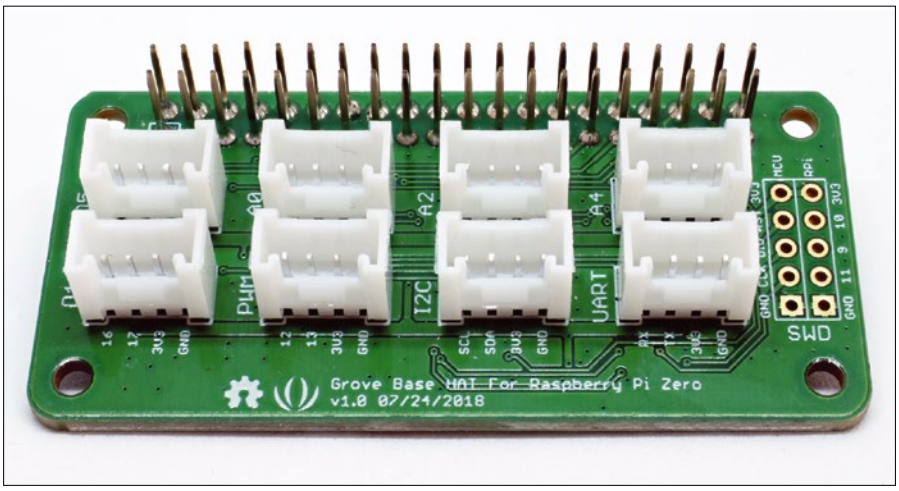


Figure 4: A Grove HAT for the Raspberry Pi Zero.

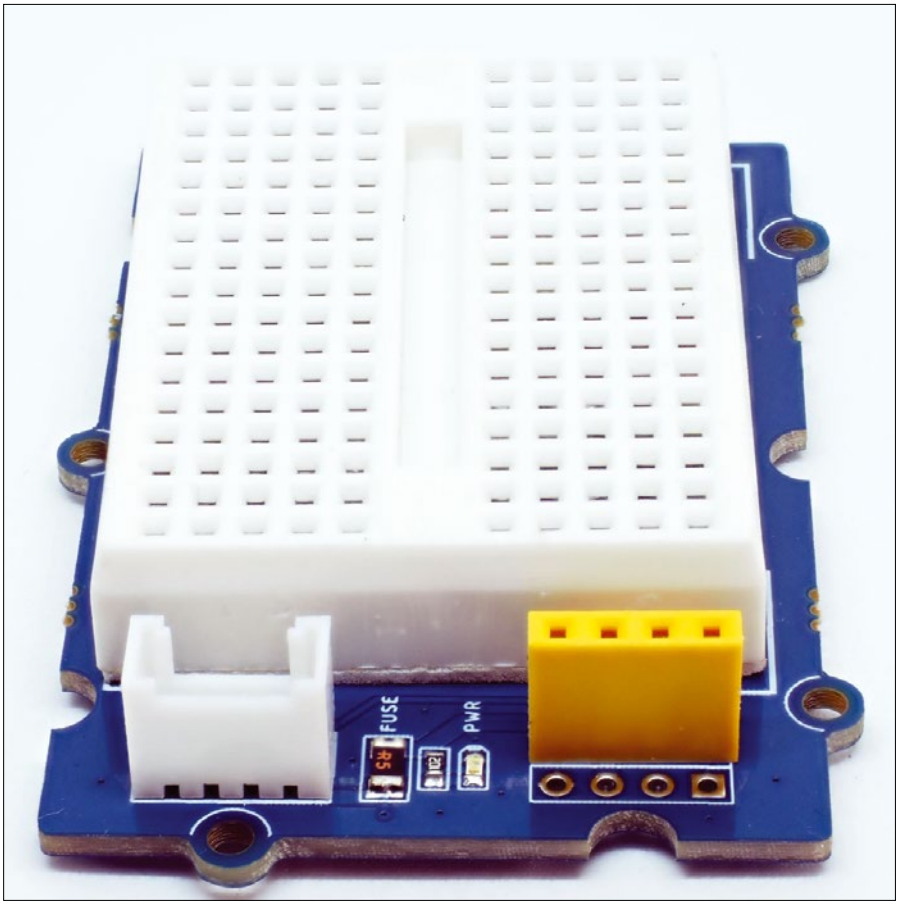


Figure 5: A mini-breadboard with Grove connector and fuse.

The double push button in Figure 3 ($2.40/EUR2.20; $2.10/EUR1.90 as a single button) is clear evidence that the plugin system is useful for solder agnostics. The cost might seem quite high at first sight, in that typical 6x6mm buttons cost just a few cents; however, the Grove push button sits on a carrier board with a socket on the back, which means you can mount it easily in a housing. As a bonus, it comes with caps in different colors. Even if DIY is not rocket science, the time you save easily justifies the price.

Grove tags these components as optimized for assembly with a *P* suffix (for "panel"). The button is also available in a version with the connector pointing upward, which is ideal for prototyping. A mini-breadboard ($3.20/EUR3) is also intended for the same purpose (Figure 5). You can use it to test your own components. A fuse (aptly labeled *FUSE* on the board) protects the connection.

Adapter cables with Dupont plugs or sockets at one end provide an alternative to normal breadboards. You can use them to connect Grove devices directly to a Raspberry Pi or Pi Pico, and vice versa to connect existing sensors with classic pins to a Grove HAT. The other manufacturers also offer this kind of adapter cable.

If you want your device to have a permanent Grove connection, a small proto shield (Figure 6) offers the solution ($2.10/EUR1.90). Strictly speaking, you would still need voltage converters, but for your own tinkering, you can do without them as long as you know your device's limits.

Assembly-optimized breakouts, a mini-breadboard, and the proto shield are good arguments for the Grove system, against which other manufacturers can't compete.

Conclusions

None of the plugin systems take the work of programming off your hands.

However, you can concentrate fully on the software instead of being driven to despair by assumed programming errors that turn out to be bad connections after hours of troubleshooting. Nor do the manufacturers leave you high and dry with the programming: Each manufacturer provides wikis and sample code for its products. Seeed even gives developers a complete Python library [8] for Grove, which gives you a standardized way to control the sensors.

If you are not a hard core solderer or a penny-pincher, then it is definitely worthwhile investing in a cable. The system decision is more about the HAT (or the shield) than the sensors, which can be connected with an adapter cable. If you are a newcomer, Grove is the best choice. Seeed's system offers the greatest flexibility, and the connectors are more robust than those of the smaller alternatives STEMMA-QT and Qwiic, which are restricted to I^2C. ■■■

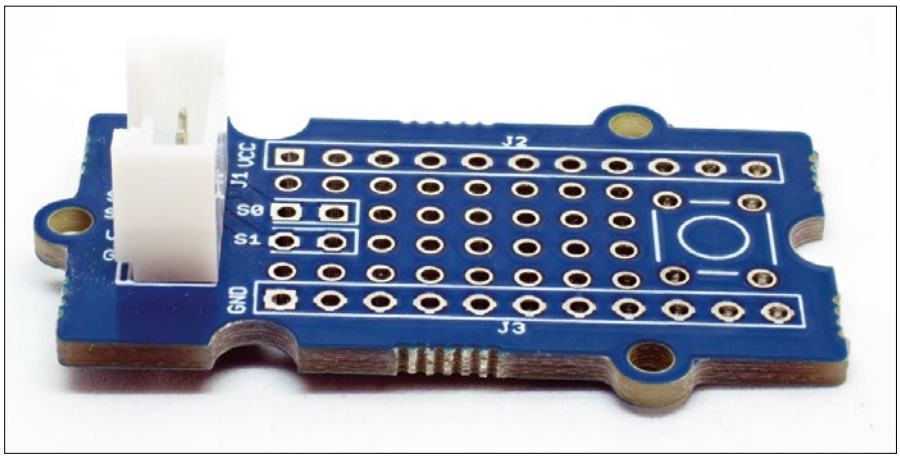


Figure 6: Proto shield for a permanent Grove connection.

Info

[1] STEMMA: *https://www.adafruit.com/category/1005*

[2] Grove: *https://www.seeedstudio.com/category/Grove-c-1003.html*

[3] Qwiic: *https://www.sparkfun.com/qwiic*

[4] Gravity: *https://www.dfrobot.com/gravity.html*

[5] Tinkerforge: *https://www.tinkerforge.com/*

[6] Qwiic HAT hookup guide: *https://learn.sparkfun.com/tutorials/qwiic-hat-for-raspberry-pi-hookup-guide*

[7] Maker Pi Pico: *https://www.cytron.io/p-maker-pi-pico*

[8] Python library for Grove: *https://github.com/Seeed-Studio/grove.py*

Author

Bernhard Bablok works at Allianz Technology SE as a SAP HR developer and likes to relax by listening to music, cycling, or going for walks. Besides this, he focuses on Linux-related topics, programming, and small-board computers. You can reach him at *mail@bablokb.de*.

FOSSLIFE

Open for All

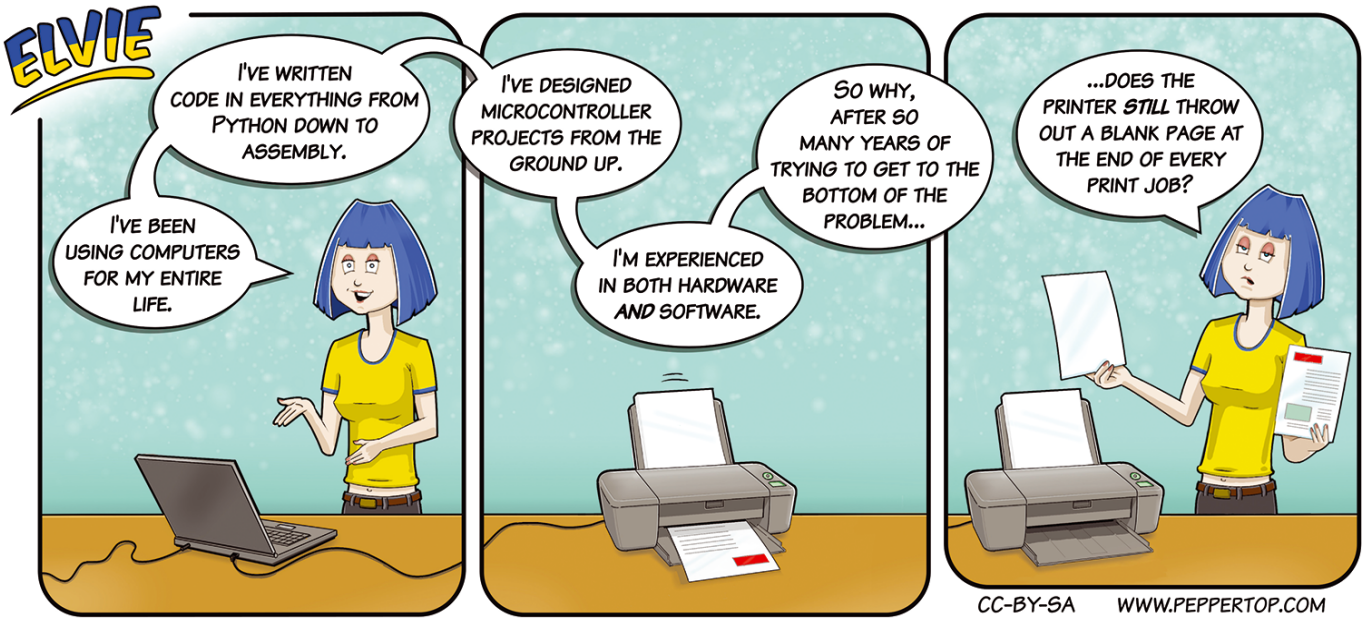
News • Careers • Life in Tech
Skills • Resources

FOSSlife.org

For most of us, interacting with a computer desktop is so natural that it hardly occurs to us that our actions are tailored to the design of our tools. With different tools, our desktop experience is much different – and some users would say, much better. Tiling window managers are an old technology that offers a very different experience, and recently they've received a flurry of new attention. An article earlier in this issue described some leading tiling window managers that are receiving attention from Linux users. In this month's Linux Voice, we take a deeper look at one of the candidates: XMonad. Also inside, we'll introduce you to a new RAW image converter that is built for better performance.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE



Turn your ideas into reality!

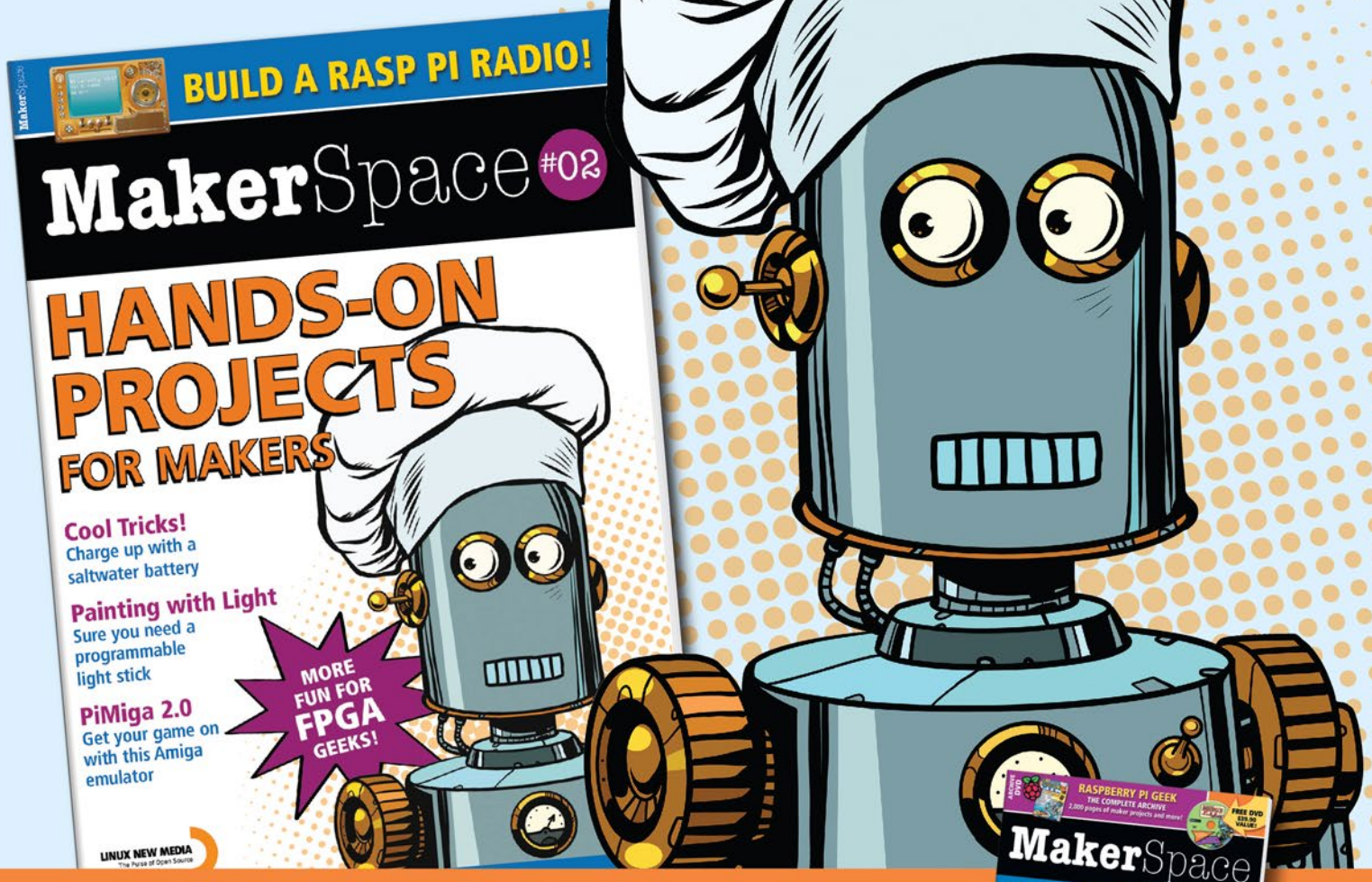
This is not your ordinary computer magazine! *MakerSpace* focuses on technology you can use to build your own stuff.

If you're interested in electronics but haven't had the time or the skills (yet), studying these maker projects might be the final kick to get you started.

This special issue will help you dive into:

- **Raspberry Pi**
- **Arduino**
- **Retro Gaming**
- **and much more!**

MakerSpace #02



ALSO LOOK FOR MAKERSPACE #01 AND ORDER ONLINE:

shop.linuxnewmedia.com/specials

MADDOG'S DOGHOUSE

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

Open source software and hardware are the best choice to protect against supply chain disruption. BY JON "MADDOG" HALL

Planning for the unexpected

Supply chain disruption is a fact of life in the business world, and recent events indicate that the problem could be getting worse. Regardless of whether the disruption is caused by armed conflict, economic pressure, natural disaster, or ordinary fluctuations in the business cycle, companies need to know they can get the supplies and support necessary to keep their businesses going.

I have worked for companies that insisted on having two suppliers for every single part they needed to run their business, in case one of the suppliers went out of business or could not meet the supply goals at some particular period of time. This flexibility is one of the reasons why people started to move to Unix systems in the early 1980s instead of staying with arguably better operating systems such as VMS, MVS, MPE, etc. (For those Unix diehards who are insulted that I mention these operating systems as "better than Unix," remember that the Unix of that day was not the robust Unix systems of 1992 and afterwards.) Yet these same companies would buy a crucial part for their business from one supplier of system software: Microsoft, citing that they could get their Microsoft operating system from system integrators such as DEC, or HP, or IBM ….

Knowledge of the Windows operating system is confined to a single company, and much of the work on Windows occurs within a single geographical area (Redmond, Washington). On the other hand, GNU/Linux is developed by people all over the world, and the source code for the system is held on servers in almost every country. This built-in diversity provides natural protection against the problem of supply chain disruption.

Another issue along the same lines is the increased use of cyber attacks disrupting business. As I have described in previous articles, I see no clear argument for whether closed source or open source is inherently more secure. The traditional arguments of "security through obscurity" (closed source) and "many eyes looking at the code" (open source) both seem to be fallacious given normal circumstances. However, in this context, the real advantage to free software (particularly) is in the mean time to repair (MTTR), based on having the source code for the software you are using and being able to generate the patch yourself, instead of waiting for a (perhaps disinterested) software provider to do the work. Admittedly, this takes some forethought in obtaining the source code for the systems and applications you use. But we are talking about unusual times.

Another issue is replacement hardware, in case that is also affected by a shortage. GNU/Linux is known for its ability to work on older equipment that has fallen out of support by other operating systems, giving you the chance to re-purpose some of that older equipment if a newer piece breaks.

And if you do replace your hardware, I would hope that you use open hardware, which other manufacturing sites could duplicate, in case the first company disappears. Finally, getting support services is easier with free and open source software. If needed, you might get your support from countries such as Argentina, Brazil, or any other country where there is Internet connectivity and known expertise in GNU/Linux. This is not 1991 anymore. Linux Professional Institute (LPI) has certified over 200,000 professionals in over 180 different countries. Be careful out there … have a backup strategy. ■■■

Exploring the XMonad tiling window manager

Graphical Interface

Many users never look back once they get started with a tiling window manager. A close look at XMonad shows why. BY PETROS KOUTOUPIS

I am not much of a graphical guy when it comes to computing. I know this sounds a bit cliché for a *nix nerd, but I live in the command line. I am a software developer and all I need is both vi/Vim and grep (no flame wars please) and a command line, preferably Bash, to use them in. I do all of my development out of locally hosted headless virtual machines and SSH into those virtual machines via a wonderful feature called port forwarding. The graphical desktop environment tends to be an afterthought for me, but even though that may often be the case, I still prefer it to be functional and allow me to remain productive.

I miss the days of simplicity when less was more. Nowadays, there are more desktop environments to choose from. Some of which are very lightweight while others dip into the heavier side of things. And while choice is never a bad thing, too much choice can often be intimidating. For instance, which desktop environment is best suited for you? Which features or functions are you looking for most? Are you looking for a composite window manager or a tiling window manager? Wait, what? What are composite and tiling window managers?

Window Managers

Many of us *nix users have grown used to the mainstream desktop environments, which include Gnome, KDE, Xfce, LXDE, or your preferred desktop environment, and it becomes difficult to fathom that others (still) exist – but they do. As unique as some of them may be, each is powerful in their own right. But before I dive into one particular desktop environment, I wish to cover the basics of window managers.

What is a window manager? In short, it is a piece of graphical software that controls the placement of windows (bearing your applications) in a windowing environment. The outcome of such a thing is referred to as a desktop environment. There are different types of window managers.

The *composite window manager* will draw each window separately and allow for them to overlap in either a 2D or 3D environment. And for those graphical environments that are not composite window manager environments but still allow for overlapping windows, those are referred to as *stacking window managers*. In order to emulate the look and feel of overlapping windows, the environment must be (re) drawn from the background window first, all the way to the foreground. When the user decides to bring one of the background windows forward or decides to open a new window, each window behind it will be redrawn.

Then you have the *tiling window manager*. Windows are in essence displayed side-by-side or above and below the others, resembling that of a set of tiles. This is the type of window manager I will explore here.

Last, you have the *dynamic window manager*. What makes the dynamic window manager unique is its ability to dynamically switch between composite and tiling window managers.

Introducing XMonad

Now that I have explained the very basics and key differences of window manager types, I will move onto a specific, and probably my most favorite, tiling window manager: XMonad [1]. XMonad is both a powerful and lightweight tiling window manager written in Haskell. It isn't often that you hear of Haskell, but XMonad runs well with it.

Installation and Initial Startup

While you can always build from source (and there is absolutely nothing wrong with that), most modern Linux distributions will provide precompiled binaries and packages for installing XMonad. To install XMonad on an Ubuntu or other Debian-based distribution, type the following on the command line:

```
$ sudo apt install xmonad dmenu
```

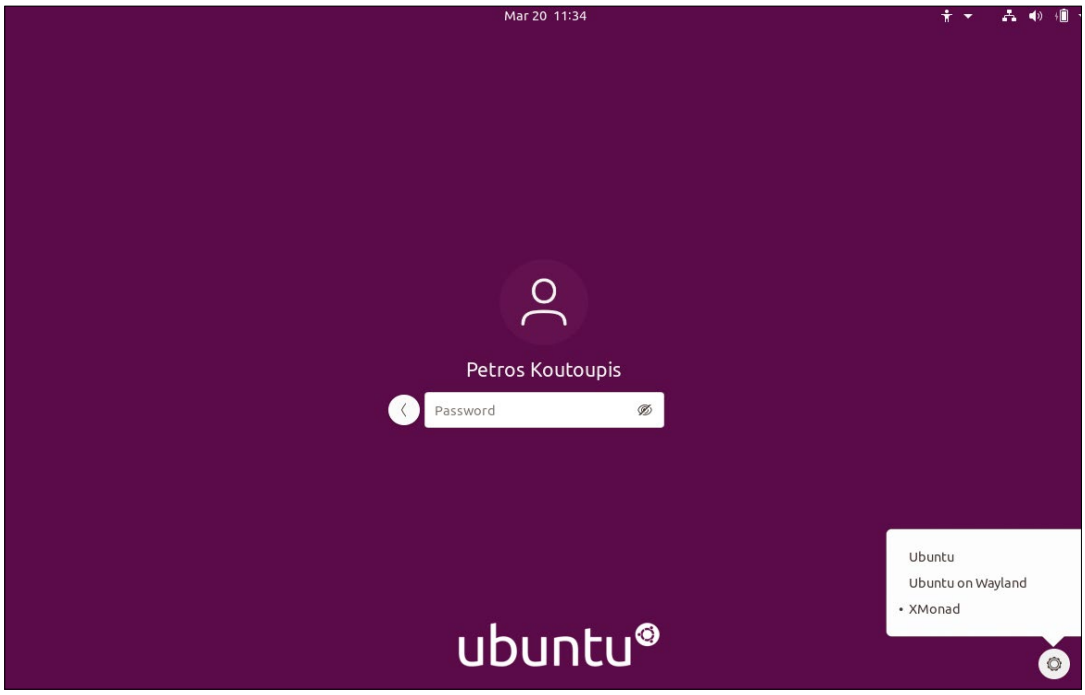


Figure 1: Select XMonad at the login screen.

You may immediately notice that you are not only installing XMonad but also a dynamic and very lightweight launcher menu for X called `dmenu` [2]. You will soon know why you are going to need this menu application.

As soon as the applications (and their dependencies) are installed, log out of your current environment and, under your user profile (in the login manager), be sure to have XMonad selected. Then log back into the system (Figure 1).

At first, you may find things a bit concerning: You will immediately be greeted by a black screen, almost as if nothing is loaded. Do not worry. This is XMonad. I will cover some customizations later in this tutorial, but for now, I'll start with the basics.

General Usage

It is important that you remember your mod key. I will be referencing this button a lot in this guide. On a traditional keyboard, this will be your Alt key (unless it is remapped), but if you are on a Macintosh-style keyboard, it will be your Option key. Alt is the default, and I will show how to remap this later in this article. Now, launch a first terminal window by pressing mod+Shift+Return. Viola! A terminal window will suddenly appear and completely cover the screen (Figure 2).

To open up yet another terminal window, repeat the previous steps of pressing mod+Shift+Return (Figure 3).

Each new tile (sometimes referred to as window pane) secures its location to the left while the previous one(s) get shifted toward the right.

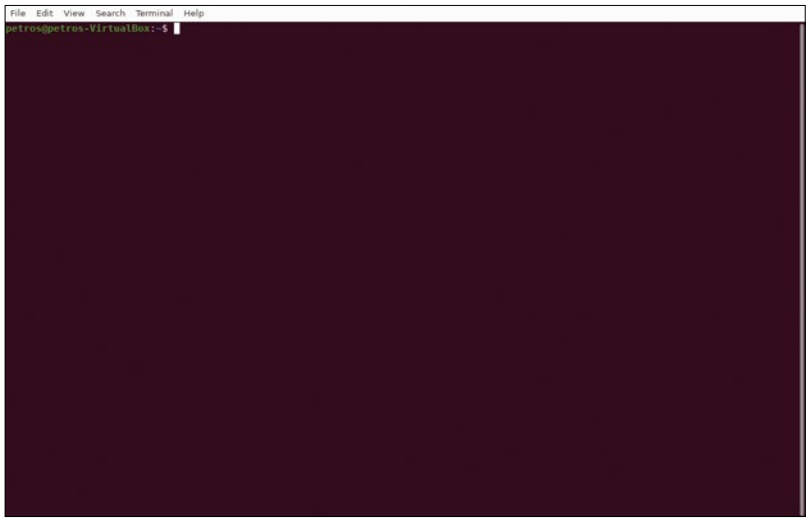


Figure 2: Opening the first application will cover the entire screen.

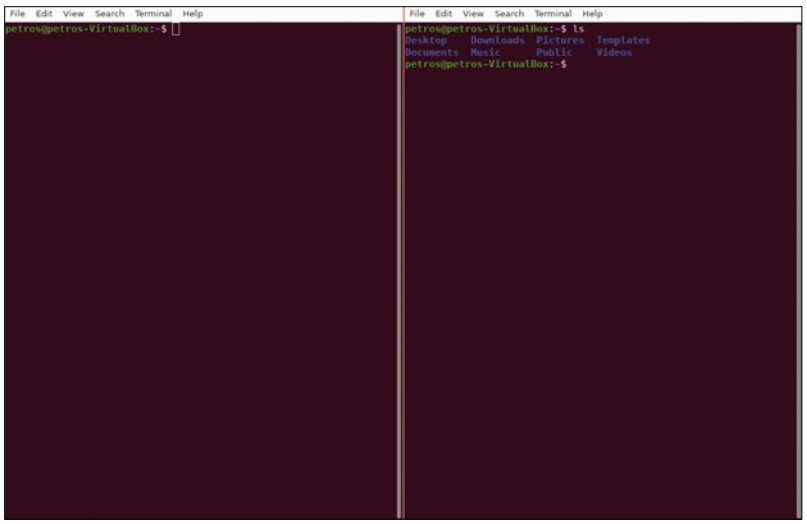


Figure 3: Opening the second application will split the entire screen.

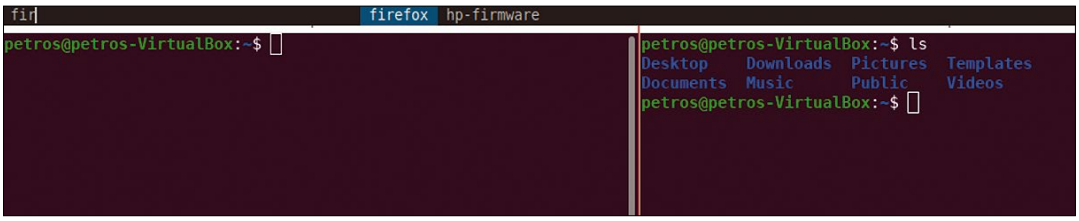


Figure 4: `dmenu` in action.

But what if you do not want a terminal? This is where `dmenu` comes into the picture. Press mod+P to invoke the `dmenu`. It will be located at the very top of your screen in its own panel (Figure 4). `dmenu` supports autocompletion which can and probably will come in handy in the future, but for now I will launch the Firefox web browser.

As one would expect, the result will showcase three tiled application windows (Figure 5). Note: The application or tile in focus will be highlighted in red. A newly opened tile will hold the focus and remain active until you either move your mouse cursor to the desired pane or you press mod+K or mod+J to move focus down or up in your workspace.

Let us say that you do not like the current tiling layout. Pressing mod+Space will throw the windows into widescreen mode (Figure 6).

Or if you wish for the current active tile to be full screen, pressing mod+Space again will do just that (Figure 7).

XMonad supports multiple workspaces (nine in total), and you can repeat the previous exercises across all of them by using mod+1 for workspace one, mod+2 for workspace two, and so forth.

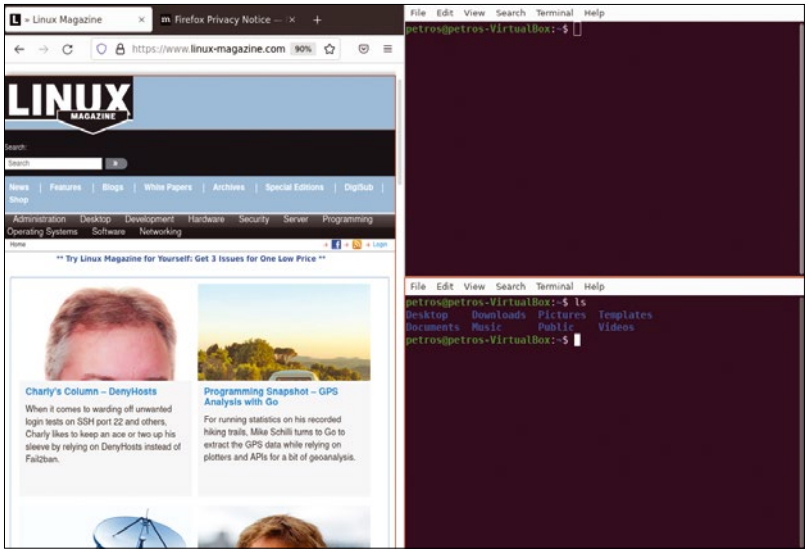


Figure 5: Opening the third application.

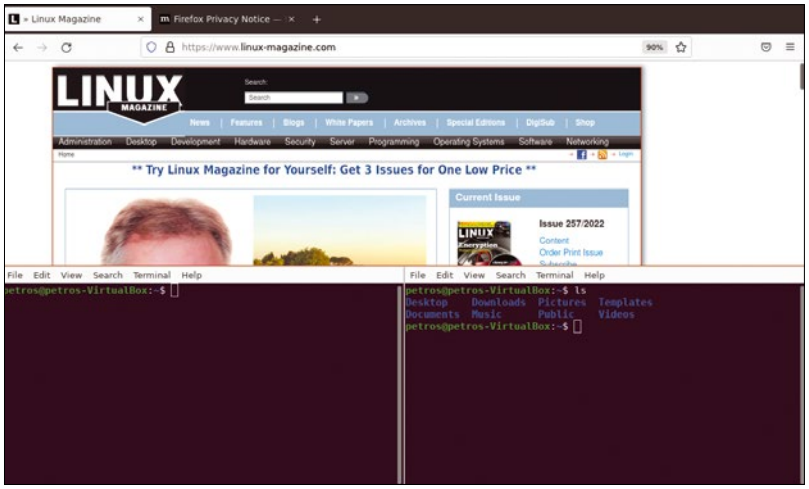


Figure 6: All of the applications displayed in widescreen mode.

Customizing XMonad

I will now shift the focus to customizing the desktop environment. Who wants to log into a machine and be greeted by a black screen as empty as my soul? In order to accomplish this, you will need to leverage the magic workings of the lightweight image viewer package called `feh`. To install `feh` type:

```
$ sudo apt install feh
```

And using a text editor, create the file `~/.xsessionrc`. Add the following contents to that file:

```
#!/bin/bash
feh --bg-scale /usr/share/backgrounds/Sunset_of_
  Peloponnesus_by_Simos_Xenitellis.jpg &
```

Note: You can replace the image and its path with one of your choosing (Figure 8).

If the `~/.xsessionrc` file already exists, then append the second line from the above example to that file.

You can immediately run this configuration by typing:

```
$ source ~/.xsessionrc
```

Or you can log out (mod+Shift+Q) and log back in.

By now, you may have come to the conclusion that XMonad is a bit lacking in the system information department. For instance, what time is it? What is the date? Where is my battery life at, or where can I obtain network statistics on my connected Ethernet interface port? Yes, you can obtain all

these answers from the command line, but if one can configure a way to have it all displayed in a panel or menubar, wouldn't that be even better?

Here is where things get really exciting. Xmobar [3] will fill in the gaps here. It is a minimalist text-based status bar, and you install it with:

```
$ sudo apt install xmobar
```

I will start with a simple time and date display in the status bar. In order to format xmobar, I need to create (or modify) the `~/.xmobarrc` file in the home directory and place the contents of Listing 1 in that file.

It is especially important to include the following three lines as they will prevent xmobar from disappearing as soon as you open new application tiles:

```
, lowerOnStart = False
, hideOnStart = False
, allDesktops = True
```

The following lines run and format the output of the `date` command:

```
, commands = [ Run Date "%a ➥
  %b %_d %Y %H:%M:%S" "date" 10
                    ]
```

The `template` lines format the placement, style, and color of the generated output:

```
, template = " }{ <fc=#ee9a00>%date%</fc> " ➥
  }
```

Next, tell XMonad to load xmobar at the startup (i.e., login) of the desktop environment. In order to do this, modify the `~/.XMonad/XMonad.hs` Haskell file (Listing 2). Most of what you find in Listing 2 is your standard template for loading xmobar.

It is especially important to include the lines

```
, handleEventHook = ➥
  handleEventHook defaultConfig ➥
  <+> docksEventHook
```

immediately after:

```
{ manageHook = manageDocks ➥
  <+> manageHook defaultConfig
, layoutHook = avoidStruts  ➥
  $  layoutHook defaultConfig
```

This too will help prevent xmobar from disappearing as soon as the applications launch.

You will also notice that your mod key is now remapped to your Super (or Windows) key. This will probably be more convenient for most.

Figure 7: The active application displayed in fullscreen mode.

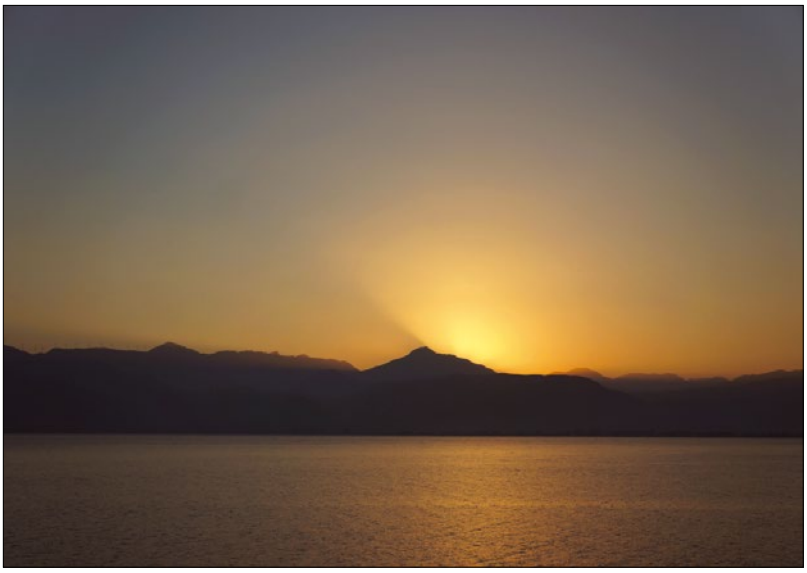


Figure 8: The new background.

Listing 1: Place in .xmobarrc

```
01 Config { font = "-*-Fixed-Bold-R-Normal-*-13-*-*-*-*-*-*-*"
02          , borderColor = "black"
03          , border = TopB
04          , bgColor = "black"
05          , fgColor = "grey"
06          , lowerOnStart = False
07          , hideOnStart = False
08          , allDesktops = True
09          , position = TopW L 100
10          , overrideRedirect = False
11          , commands = [ Run Date "%a %b %_d %Y %H:%M:%S" "date" 10
12                       ]
13          , sepChar = "%"
14          , alignSep = "}{"
15          , template = " }{ <fc=#ee9a00>%date%</fc> "
16          }
```

Listing 2: Loading xmobar

```
01 import XMonad
02 import XMonad.Hooks.DynamicLog
03 import XMonad.Hooks.ManageDocks
04 import XMonad.Util.Run(spawnPipe)
05 import XMonad.Util.EZConfig(additionalKeys)
06 import System.IO
07
08 main = do
09     xmproc <- spawnPipe "/usr/bin/xmobar /home/petros/.xmobarrc"
10     XMonad $ defaultConfig
11         { manageHook = manageDocks <+> manageHook defaultConfig
12         , layoutHook = avoidStruts  $  layoutHook defaultConfig
13         , handleEventHook = handleEventHook defaultConfig <+> docksEventHook
14         , logHook = dynamicLogWithPP xmobarPP
15                         { ppOutput = hPutStrLn xmproc
16                         , ppTitle = xmobarColor "green" "" . shorten 50
17                         , ppHiddenNoWindows = xmobarColor "grey" ""
18                         }
19         , modMask = mod4Mask     -- Rebind Mod to the Windows key
20         } `additionalKeys`
21         [ ((mod4Mask .|. shiftMask, xK_z), spawn "xscreensaver-command -lock")
22         , ((controlMask, xK_Print), spawn "sleep 0.2; scrot -s")
23         , ((0, xK_Print), spawn "scrot")
24         ]
```

Now that the files are created or modified, you will need to log out of and back into the desktop environment. As soon as you do, the Haskell file will recompile and you should immediately notice an informative panel at the top of the screen (Figure 9).

We will add more information to xmobar. Revisit the original `~/.xmobarrc` file and modify the `commands` field to register the following:

```
, commands = [ Run Network "enp0s3" ↵
  ["-L","0","-H","32","-normal",↵
  "green","--high","red"] 10
        , Run Date "%a %b %_d %Y ↵
          %H:%M:%S" "date" 10
                    ]
```

And modify the `template` field with:

```
, template = " }{ %enp0s3% | ↵
  <fc=#ee9a00>%date%</fc> "
      }
```

As soon as you reload xmobar, a new field showcasing the network speeds on interface `enp0s3` will be noticeably present (Figure 10).

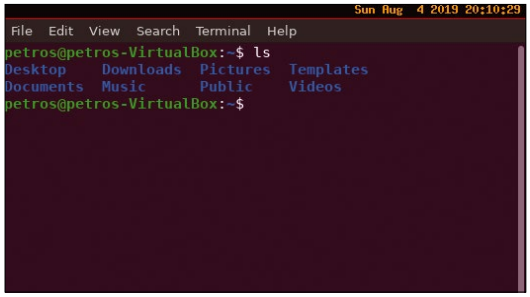


Figure 9: From this point on, you will forever know the time and date inside XMonad.

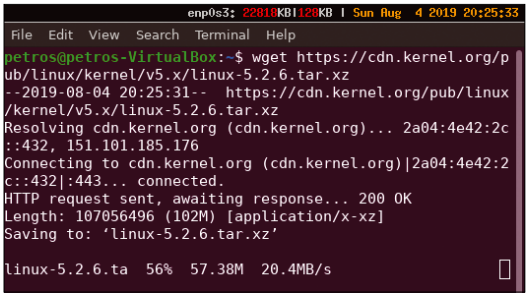


Figure 10: Xmobar displaying the download and upload speeds on the preferred Ethernet interface port.

Listing 3: .xmobarrc Updates

```
Config { font = "-*-Fixed-Bold-R-Normal-*-13-*-*-*-*-*-*-*"
       , borderColor = "black"
       , border = TopB
       , bgColor = "black"
       , fgColor = "grey"
       , lowerOnStart = False
       , hideOnStart = False
       , allDesktops = True
       , position = TopW L 100
       , overrideRedirect = False
       , commands = [ Run Network "enp0s3" ["-L","0","-H","32","--normal","green","--high","red"] 10
                    , Run BatteryP ["BATC"]
            ["-t", "<acstatus><watts> (<left>%)",
             "-L", "10", "-H", "80", "-p", "3",
             "--", "-O", "<fc=green>On</fc> - ", "-o", "",
             "-L", "-15", "-H", "-5",
             "-l", "red", "-m", "blue", "-h", "green"]
            600
                    , Run Date "%a %b %_d %Y %H:%M:%S" "date" 10
                    ]
       , sepChar = "%"
       , alignSep = "}{"
       , template = " }{ %battery% | %enp0s3% | <fc=#ee9a00>%date%</fc> "
       }
```

If this were a laptop and you also wanted to add battery life status, only a couple modifications would need to be made to the updated `~/.xmobarrc` file (Listing 3).

Conclusion

This article described the different types of window managers for graphical desktop environments and also focused on the very powerful and lightweight tiling manager, XMonad. Once you get used to the various aspects and functions of a tiling graphical environment, your fingers never have to leave the keyboard, and your productivity is never limited. Besides, you can even expand more on xmobar and add more information relevant to you and your operating environment. It does not need to end here. ■■■

Info

[1] XMonad: *https://XMonad.org/*

[2] dmenu project page: *https://tools.suckless.org/dmenu/*

[3] xmobar GitHub repository: *https://github.com/jaor/xmobar*

The Author

Petros Koutoupis is currently a senior performance software engineer at Cray (now HPE) for its Lustre High Performance File System division. He is also the creator and maintainer of the RapidDisk Project (*www.rapiddisk.org*). Petros has worked in the data storage industry for well over a decade and has helped pioneer the many technologies unleashed in the wild today.

FOSSPicks

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

Not content with his coffee machine simply making coffee, Graham has recently been trying to upgrade his faithful machine with an Arduino and REST API.

BY GRAHAM MORRISON

Research assistant

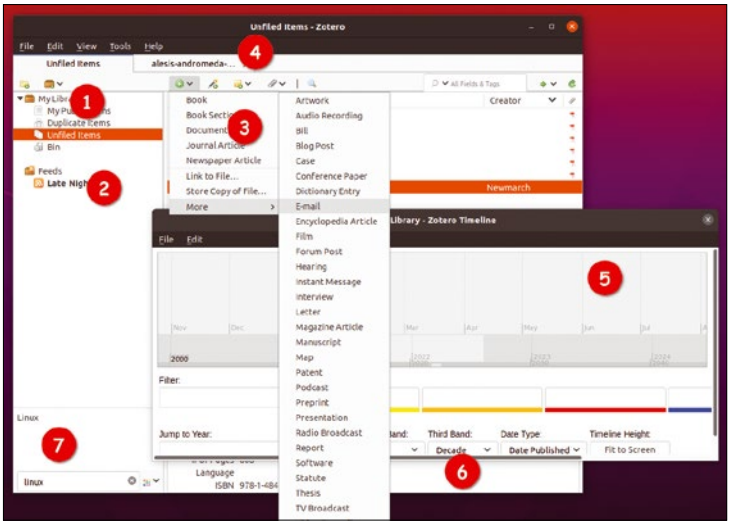
Zotero 6

This is one of those pieces of software you discover that leaves you wondering how you'd not encountered it before. Especially when it's at version 6, and especially when it does something better than the majority of similar applications out there. Zotero calls itself a research assistant, but it's really a document collection manager, document reader, and notes and annotation tool, with a powerful emphasis on making it easy to add and organize things. It's been developed specifically to help academics to create and collate their own collections of research papers, but it's just as useful for anyone with a folder full of manuals, web links, typed notes, and any other word-based miscellany.

Zotero's function is easier to understand by looking at the main application window. This is split into three resizable columns, with the left column holding the files and virtual folders of your library, the middle view displaying the individual files for a selected virtual folder, and the right column showing context-sensitive details for your selected document. It feels a lot like a development environment, and almost everything about these views can be changed and modified. New documents are dragged into your collection, optionally tagged, moved to a folder, or saved into a different top-level collection, and one of Zotero's great strengths is the diversity of these sources. It works best with PDF files, but it can also handle archived websites and both RSS and OPML subscriptions, much like an RSS reader. You can even add ISBN values for published papers, books, and magazines, and the reference information for those publications will be automatically retrieved.

This functionality is augmented by what are called "connectors." These are specific add-ons for web browsers that integrate directly with Zotero, allowing you to add to your library as you're browsing the web. When you want to save a website, click on the *Zotero* button and the page is automatically archived and added to your library. Double-clicking on these from the main application will open your browser, but you're looking at an offline version so that you can be sure the page contents won't change. Double-clicking a PDF, however, will open an internal PDF viewer, and it's one of the best we've used on Linux. It's quick, accurate, and makes it easy to skip through large documents. Opened PDFs also appear as tabs in the main application view, so it's easy to keep more than one open and work across different chunks of text. Tags, notes, highlights, and annotations can be added to any of these documents, but unlike more traditional PDF readers, these don't change or update the source material. Changes are instead stored separately where they can be more easily integrated or cited from your own writing or new documents. You can also write within the application editor itself, which is useful for your own thoughts and ideas.

While all this is brilliant for academic papers, it's equally good for any collection you might want to document, whether it's sci-fi books or magazines, and it's one of the best applications we've found for both managing and organizing that collection, and for keeping your own personal notes on each item and the entire collection all in one place.



1. Library: Create separate collections for your documents and organize them into folders. **2. Feeds:** Use RSS to keep on top of new publications and website updates. **3. Media types:** All kinds of media can be added alongside text-based documents. **4. PDF viewer:** PDF files are opened as background tabs, and the viewer itself is excellent. **5. Timeline:** Track your additions, annotations, publications, and edits over time. **6. Metadata:** Use tags and search terms to better access text in your library. **7. Views:** Panels can be stacked, and there are supporting browser scripts for browser integration.

Project Website

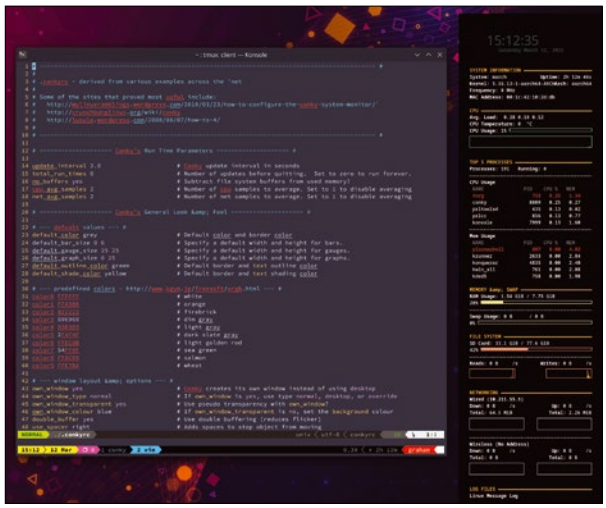
https://www.zotero.org

System monitor

Conky

This wouldn't be FOSSPicks without some form of system-monitoring tool. And while we usually look at new and shiny monitoring tools as they appear, Conky has been around for a while. It was featured in this very magazine back in 2009, and it has since continued to develop and add new features and many new users. This means it's more mature than the typical monitor, benefiting from the stability and efficiency that comes from an established project, as well as from an active community sharing tips and configuration files. This community is important because Conky's best feature is its incredible flexibility – Conky can be whatever monitoring tool you want it to be. There is a default layout, but no one uses it. Instead, most of us use Conky to create our own perfect monitoring configuration without having to create an entire tool from scratch. All this flexibility is thanks to a configuration file which, unlike many configuration files, is easy to understand, easy to modify, and easy to expand.

Through the configuration file, you can reference any one of the 300 built-in objects, including the usual suspects of CPU, memory, storage, and network statistics but also fields with advanced control over media players and even IMAP and POP3 email, among many others. You also have a lot of control over how this data is formatted, by referencing any object within a string, and how it's presented as either a value or one of several chart types. If you can't find built-in support for your favorite statistic, it's equally easy to integrate a shell command into the configuration file or even use the internal Lua scripting engine. The



Conky has been around for a long time, but it's still being developed and now has a huge library of example configuration files.

end result is either a floating panel or background widget built to your exact specifications that can integrate perfectly with your desktop, from font size to color palette. And because Conky is so well established, you can find many example configuration files online and steal various monitoring snippets to build your ideal solution.

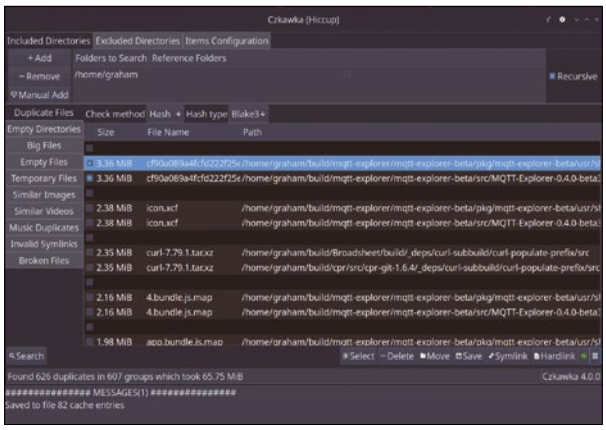
Project Website
https://github.com/brndnmtthws/conky

Duplicate file detector

Czkawka

Duplicate file removing tools are a niche but long-lived software category, especially on Windows where there's a small cottage industry selling utilities to do exactly this. Similar tools exist on Linux, such as `rdfind`, `fslint`, and `dupeguru`, but they're seldom updated and can feel slow and cumbersome on modern systems. This leaves many of us to either create our own command-line incantations or simply kick the problem into the future with more storage. Czkawka, however, is a modern duplicate file remover that will feel familiar, while also being modern, fast, and efficient. This modernness and efficiency, like many recent projects, comes from being built with Rust. Even if you keep tight control of your filesystem, Czkawka still finds duplicates and provides the perfect options for managing them.

There's both a command-line front end and a GUI built with GTK3, which has a lot in common with the GUI for FSlint. You can add locations you want to scan to a top panel, including directories you wish to exclude, and whether you want a scan to be recursive or not. Pressing *Search* will launch the duplication detector process, which we found to be much quicker than the alternatives, by orders of magnitude. This is obviously vitally important when you're scanning gigabytes of data, and the results are listed in the center of the display. Empty files and broken symlinks are also listed, and you can also easily find large files on their own. You can then filter results by type, with special cases for music, video, and



Czkawka means "hiccup" in Polish, and it can scan for duplicate files by filename or size and three different types of hash.

image duplicates. Duplicates can be automatically selected and moved, deleted, and saved, with two more options to create either a symlink or a hard link. These last two options replace the files with a link to a single instance, saving storage without breaking anything that might rely on the file being in one of the duplicate locations.

Project Website
https://github.com/qarmin/czkawka

Python library

Rich

This is a bit of an odd one for FOSSPicks because it's not purely a standalone utility, application, or server. Rich is primarily a third-party library for Python programmers that helps them create beautiful text-based output. If you're any kind of Python programmer, you'll know there are probably dozens of such libraries, so what makes this one special? It's wonderfully engineered and documented and can easily be used by any script programmer who might have more ambitious plans for their output. All you need to do is create a console object within your code and use this to print whatever output you require. The formatting comes from using Rich's "renderables," which is a library of objects that can be used to easily format tables, progress bars, tree views, columns, Markdown, and code examples, all within print commands or log output.

The output looks fantastic and quickly solves the problem of formatting text manually. You don't have to worry about justifying text, 8-bit and 4-bit color, or even adding emoji. It's all handled automatically. You can even access these functions if you don't want to write the code yourself, thanks to an accompanying `rich-cli` tool. This is a standalone binary that takes all kinds of text input and uses Rich's routines to generate perfectly formatted output on your command line. Pipe through code or a logfile to see syntax highlighting, for example, or a Markdown text file to see formatted output. CSV files are automatically placed into a table, and you can even give Rich a URL to render as text. It's special because you don't have to be a Python programmer to find it useful.

The Rich command-line tool includes a pager flag to manage output, an HTML export option, and further options for style and theme switching.

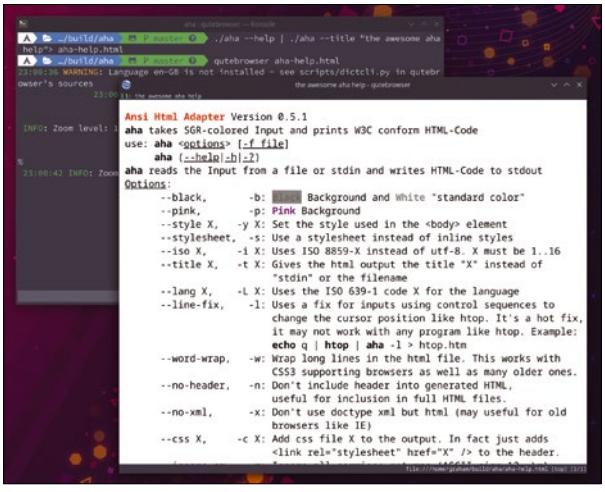
Project Website
https://github.com/Textualize/rich

ANSI to HTML

aha

As much as we'd like to write that this is a recreation of the Yamaha DX7 synthesizer, famously used by the band A-ha on their pop hit "Take On Me" in 1985, we can't, because it isn't. This `aha` is instead almost the inverse of Rich (above) because it takes colorized terminal output as an input and converts it into HTML for publishing online. ANSI color and layout is included in the output, alongside any text styles and formatting, making the resultant HTML a surprisingly effective visualization of command output, and the source of a decent local hosting service. This might sound like a simple process, but `aha` does this in such a useful way that after using it for a few things, you quickly start to think of many others. The tool is tiny, with no dependencies, and easily built from the makefile if you have the build essentials package for your distribution installed. This leaves you with the `aha` binary which you can then pipe things into.

Type the command, `aha --help | aha --black --title "aha help"> aha-help.html`, for example, and you're piping `aha`'s own help output into itself so that it can convert the SGR-colored help text into properly written HTML output. This can then easily be viewed in a web browser or hosted on a server. You could formulate a command that runs periodically, updating a web page with system statistics, file contents, log messages, or any other kind of mash-up of the commands you can put together on the command line. This makes `aha` the perfect example of a simple command that can be an integral part of a much more complex chain of commands, which is exactly how the command line should be used.

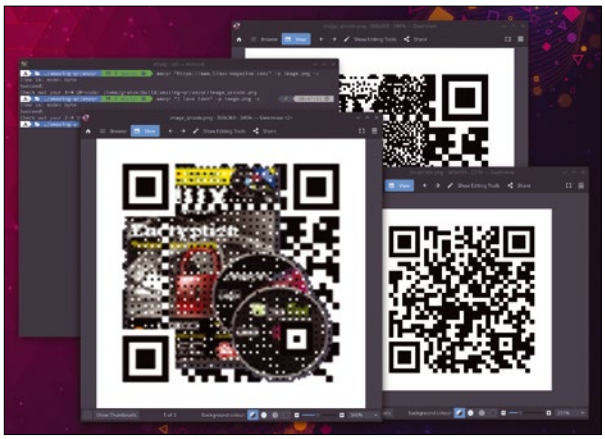


While terminals such as Konsole can convert a copy to HTML, `aha` does this as a simple, useful command.

Project Website
https://github.com/theZiz/aha

QR generator

Amazing-QR



Amazing-QR can even generate animated QR codes that can be saved as GIF files.

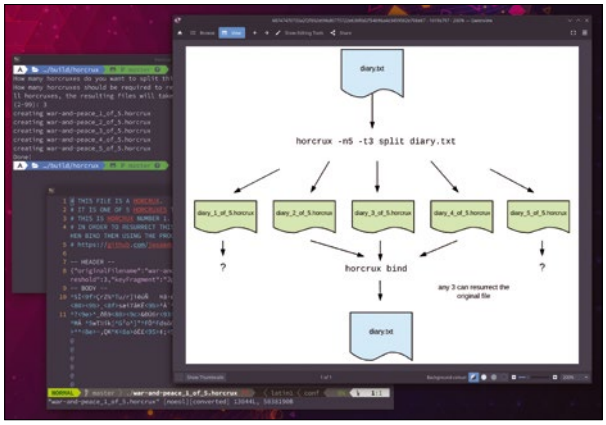
QR codes, despite a slow start, have become ubiquitous. They're now used for everything almost everywhere, from restaurant opening times to NFT ownership. And very much like NFTs, they don't actually store anything more useful than a URL; username; or a long, difficult-to-copy hash code, with a typical data limit of around 3KB. They're successful because scanning a QR code is more convenient than copying a long unique URL manually or relying on OCR to automatically extract the data from text. If only they weren't so boring to look at! And this is where Amazing-QR can help.

Amazing-QR is a small Python utility that can encode any data you provide into a self-generated QR code that it saves by default as a PNG file. The *X* and *Y* pixel size of the QR code is calculated from the amount of data to encode alongside the selected error correction level, which defaults to the highest (*H*). Both of these options can be customized, as can the output image file format. The result is a QR code that can be widely interpreted and used just like any other QR code. But there's more. So far, we're only generating the same boring old QR codes, and Amazing-QR has a couple of unique tricks up its sleeve. By adding the `-p` argument followed by an image's filename, the image is itself incorporated into the QR code. The result is a crude black and white representation of the image behind the code, but crucially, the QR code still works and is much more interesting than the codes we've all become bored of. You can take this even further with the `-c` option, which takes a color image and integrates this into a color QR code. This looks even better and is totally unlike any QR code we've seen in the wild.

Project Website
https://github.com/x-hw/amazing-qr

File splitter

horcrux



Horcrux uses Shamir's Secret Sharing (SSS) to encrypt a file split into several pieces.

Back in the days before the World Wide Web, back when BBSs were common and, if you were lucky, the Internet was accessible via dial-up, downloading large files was difficult. There were several reasons for this, but the most important one was that it was hard to maintain a slow connection for a long period of time. Combine this with the popularity of sharing files on Usenet, and it became common to split large files into smaller files that could later be recombined or stored across several floppy disks. The brilliantly named horcrux is a modern version of the same functionality, only this time designed specifically to help keep your files secure with encryption. Horcrux is a command-line tool that's been designed to offer a form of secure encryption without the need for a private key or a passphrase. It does this by splitting a file into a selectable number of pieces with a selectable degree of redundancy. The redundancy lets you recombine the original file from a number of split files less than the total. You might choose to split your personal diary into 10 pieces, for example, but make the diary reconstructible from just five. The idea is that you distribute each of these pieces to locations you can presumably remember better than a passphrase, bringing at least the redundancy number together if you ever need to access the file. Locations could be physical, such as a USB stick or optical disc, or digital, such as cloud storage or a remote server. Only two commands are needed to do all this, `horcrux split` and `horcrux bind`, and you're prompted for the numbers of pieces and redundancy when necessary. It's a great idea, and not unlike its namesake totems in the Harry Potter books, only hopefully without the same consequences if they're ever destroyed.

Project Website
https://github.com/jesseduffield/horcrux

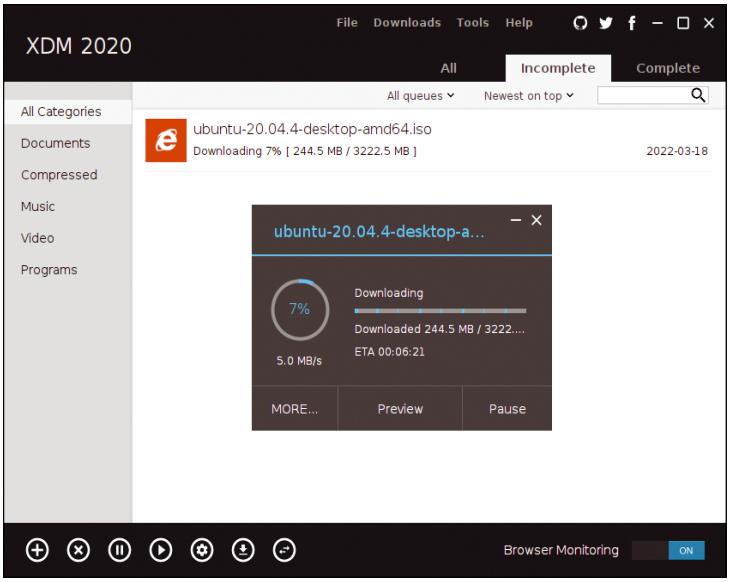
Download manager

XDM

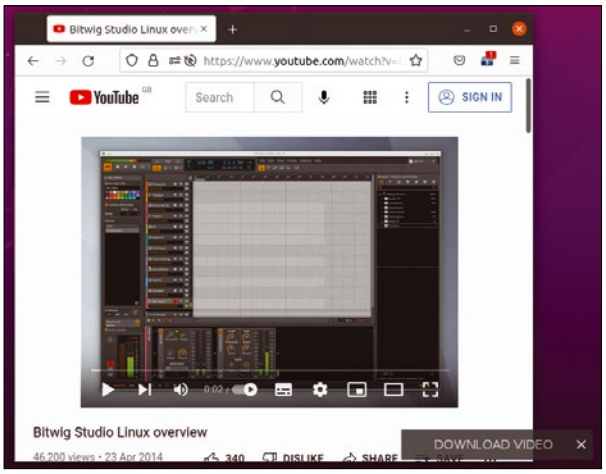
Back in the days when the Internet was a lot slower for most of us, and a lot less consistent, download managers used to be a thing. These were separate applications you could launch to offload your download duties, such as grabbing the latest distribution ISO, or a suite of links off a web page. There were many to choose from, all with features such as bandwidth limiting, or server load balancing, suspend and resume, and destination folder determination based on download type. The combination of broadband Internet and modern browsers have mostly replaced these requirements with browser functions. Because we typically always run a browser, it's now acceptable to leave the browser running while things download in the background.

But there are still many reasons for running a download manager, and Xtreme Download Manager (XDM) rolls them all into a single application. XDM is a modern download manager that accepts you'll be using your browser most of the time and offloads your downloads to a separate set of processes and screen real estate. This really helps if you download a lot of files or still need complex rules for specific file types. It's also a tool that's going to be most useful for people who view a lot of videos because it can also convert those videos on the fly or preview their content before a file has completed downloading. Without a plugin, and with a simple click or paste, video from many different sites can be downloaded directly and converted to a more portable format. It can do this with its own browser plugin, or by monitoring the clipboard, and downloads can be scheduled and even the system shutdown on completion.

XDM will download more than one file at a time, to make the most of your bandwidth, and it promises to accelerate downloads by up to five times thanks to its "segmentation technology." In our tests, it was certainly fast, but no faster than a good server source. You can also set an option to download a single file at a time or limit the download speed for all downloads to a specific bitrate. This option is particularly useful when you're sharing a limited bandwidth network with people who might be in a video conference.



XDM can manage a single download, multiple downloads, or a playlist of downloads across HTTPS, FTP, MPEG-DASH, Apple HLS, and Adobe HDS protocols.



The browser plugin will selectively monitor your video playback state and offer to download matching streaming videos.

While not necessary, the accompanying browser plugin does make using XDM much easier, because it's from the browser you're most likely to find the things you want to download. There's a specific add-on for most popular browsers, including Firefox, Chrome, and Opera, and the add-on will add an option to that browser's right-click menu, allowing you to add to the download queue directly. The add-on can optionally monitor whether a video is playing on the current page. If it does detect a video, it flashes up an option to add that video to the download queue. This is a powerful feature which, when combined with the format transcoding, is a great way of pushing your streaming queue from a browser session to something more convenient, such as a television or tablet.

Project Website

https://xtremedownloadmanager.com

Pixelated RPG

Ambermoon.net

Released a year after the Amiga 1200 in 1993, Ambermoon was an amazing role-playing game that could take full advantage of the Amiga's capabilities and maturity. It featured beautifully hand-drawn graphics, and both a top-down mode for adventure views and a 3D first-person mode for dungeons and cities. The sound was also wonderfully evocative of the era, full of 8-bit wholesomeness. But of course, that era has never particularly left us, and it could be argued that the gameplay, sound, and aesthetics of that time are now more broadly popular than they ever were in the 1980s and 1990s. Which leads us to this brilliant recreation of the original Ambermoon for our modern Linux systems, imaginatively called "Ambermoon.net" because it's been rewritten in the language of .NET (C#).

The download includes a PDF of the manual, a chart of runes, and even a PDF version of the location map. These are of course digital copies of the originals, but the unusual care the developer has gone to by including them makes you feel a little like you might have felt opening the physical game box in 1993. The game itself is still absolutely worth playing and features a mixture of 2D Zelda or Ultima-like countryside and villages with a revolutionary (for the time) Doom-like, 3D texture-mapped, first-person view for dungeons and cities. This latter part is particularly clever, featuring bitmap graphics distorted to look like polygons – much like an even earlier title called Alternate Reality on the Commodore 64. There's a great story that links to the earlier games in the series, and you're pulled through an adventure to explore, collect items, enlist followers, cast spells, manage resources, and ultimately defeat whatever malevolent power controls your destiny. Not bad for a 30-year-old game.

Ambermoon.net is an open source recreation of a famous Amiga game, but what's most remarkable is that the project produces updated binaries for a real 68000-based Amiga!

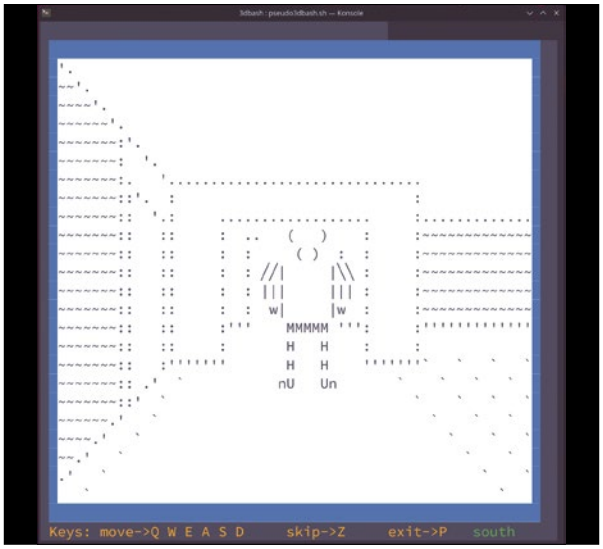
Project Website
https://github.com/pyrdacor/ambermoon.net

Bash 3D engine

pseudo3dbash

One of the most unique things about Ambermoon (above) was that, when it was originally released in 1993 on the Amiga, it was able to create a 3D texture-mapped view for the player as they walked around dungeons and cities. Unlike on the PC, this was a difficult trick for the Amiga to pull off, and Amiga fans at the time were desperate to see games similar to Wolfenstein-3D, which had come out the year before on the PC. But even then, 3D first-person views had been around for almost forever, despite the relatively complex mathematics required to calculate the view. Even the humble Sinclair ZX81, with its meager 16KB of RAM, ran a game called 3D Monster Maze, and most home computers of the '80s had their equivalents.

This is what pseudo3dbash is for Bash terminals. It's a prototype pseudo 3D engine written purely in Bash's own scripting language. This means you can install and run the game on almost any device with a Bash prompt, and because the script is so accessible, it's very easy to understand and modify. The graphics themselves are rendered as simple ASCII, but it's easy to see the maze, and there are various monsters to encounter and an exit to find. As with many games, you turn left and right with the *A* and *D* keys, forward and backwards with *W* and *S*, and *Q* and *E* to "strafe" left and right. The source code implements a pseudo-3D engine using lots of expanded `if` statements, making it a great first coding project or something that could be used as the basis of a 3D engine in another language. It could also be easily expanded to use a graphical framework other than ASCII characters and might work well when converted to Python using the Rich Python library we looked at earlier.



Bash is actually quite a capable programming environment and a good place to start your coding adventure.

Project Website
https://github.com/diejuse/pseudo3dbash

RAW development with Vulkan darktable

Dance on a Volcano

The RAW converter Vulkan darktable outpaces its competitors with a modern node-graph-based architecture and massive use of the GPU.

BY ANNA SIMON

Open-source photographers have access to a massive selection of RAW developers, the two best-known and most mature representatives of this category being darktable [1] and RawTherapee [2]. Both are aimed at demanding amateur users and professional photographers and have an enormous range of functions.

Some time ago a RawTherapee fork named ART [3] saw the light of day. ART tries to make it easier for beginners and technically less-experienced users to get started. In addition, a new version of LightZone [4] was released a few months ago, after a gap of several years. PhotoFlow [5] and Filmulator [6] are also two interesting new programs.

This group has recently been joined by a completely new program: Johannes Hanika, the founder of the darktable project, is now working on a RAW converter, the biggest highlight of which is its processing speed. Because the tool uses the Vulkan programming interface [7], which was previously mainly used in computer games, he has dubbed the software Vulkan darktable, or vkdt for short. (See the "Interview with Johannes Hanika" box.)

After almost three years of development, the software is now relatively mature. Although vkdt can't quite keep up with its big brother darktable or RawTherapee in terms of the feature set, it's well worth a peek for inquisitive users.

Sucks Less

On the vkdt GitHub page [8], Hanika refers to his program as a "darktable which sucks less,"an allusion to the *suckless.org* project, which focuses on minimalist software. Some of the better-known *suckless.org* applications include the dwm tiling window manager, the dmenu program menu, and the keyboard-driven surf web browser. The developers of these applications consider most software to be too complex, too bloated, and overloaded with too many features. In their opinion, a computer program should be limited to the bare essentials, not only in terms of the feature set, but specifically in the amount of code. For example, *suckless.org* set itself a fairly ambitious limit of 2,000 lines of code for the dwm window manager.

The project's website [9] characterizes vkdt as "an experimental image processing graph," which implies that it is not a traditional layer-based photo editing program like Photoshop but a node-based software like Nuke [10], which is used in video and film post-production for compositing and special effects. The operating principle of this kind of program is based on nodes, which are arranged in a node graph. A node stands for nothing other than an editing step (i.e., an effect such as exposure or color correction). The special feature in vkdt is that each node can have several *inputs* and *outputs*. This means, for example, that you can merge multiple images and then process the results as a single image.

If you visualize the complex editing of a photo with vkdt, you get an acyclic directed graph that resembles a subway map more often than not. Thus far, only video editing programs have consistently implemented this principle, but tools such as Lightroom were already closer to node-based editing than to layer-based editing. The adjustment layers feature in Photoshop is also a move in this direction. It should be noted at this point that some users have been using node-based software such as Nuke or Natron [11] for editing photos for quite some time.

No Experiments

Don't be confused by the word "experimental" in the vkdt description. Vkdt is already quite suitable for practical work. The word instead expresses that the program makes use of technologies that have not yet been used for photo editing software in this way.

Vkdt also resembles video editing software in terms of processing speed. According to Johannes Hanika, vkdt computes about 20 times faster than traditional RAW converters, although it also processes 20 times more data. This is only possible because it uses the Vulkan framework to run all the computations on the graphics card. GPUs have many more cores than CPUs, so they can perform many calculations simultaneously.

This mainly benefits graphics applications and computer games, which calculate many pixels synchronously in this way. However, most photo editors and RAW developers so far have mainly – many even exclusively – used the CPU. Before vkdt, darktable was the RAW converter that most consistently put GPU acceleration into practice (using the older OpenCL programming interface). Having said this, even darktable still uses the CPU for many tasks.

Fast

However, vkdt's speed is not an end unto itself. Traditional RAW developers only ever compute the area of the photo currently on view in the preview window before the final export. If you move the view, depending on your computer's processing power, it can take some time until you get to see the preview. If you view the image in the full-screen view, the program usually only works with a low-resources version of the photo in order to save computing power and thus time.

For some effects, however, the software cannot calculate a correct preview in this way because this always requires the entire, or full-resources, image as a starting point. Creating a preview that is more or less correct is therefore only possible with massive programming overhead. The preview does not precisely match the final exported image, especially if viewed at different zoom levels or if the view is shifted. This problem is completely eliminated with vkdt because the application always computes the entire image at full resolution and keeps it in the graphics card's memory. If you view a photo in vkdt at 100 percent scale and move the display area, you don't have to wait for the program to compute a preview.

One of the features not yet offered by other RAW developers is support for 10-bit output. Apart from vkdt, only Photoshop and Krita can do this. This means that appropriately designed screens display, say, color gradients in a precise way and without streaking. And all editing modules consistently work in the linear RGB color

Interview with Johannes Hanika

Linux Magazine (LM): *Why and for whom are you developing vkdt?*

Johannes Hanika (JH): Why? Probably because I can. I don't think the old pipeline is sustainable anymore. You can't focus on everything as a developer working on your own. In the case of darktable, [the focus] was certainly more on the interface, the feature set, and the workflow – not at all with a view to a pipeline architecture. It's hard to change that now. And for whom? The same answer as for the original darktable. First of all, for myself, to avoid some other software getting on my nerves. But I'm also happy to share with anyone who wants to use vkdt, as well, or [who] can even contribute something. The darktable community has always been very friendly – and put a lot of ideas and work into the project and made it massive fun for me.

LM: *When you started developing darktable, didn't you secretly aim to make a better Lightroom?*

JH: One important aspect was that all the commercial stuff never ran on Linux. Even RawTherapee wasn't open source under the GPL back then, and it just wouldn't set up on my system. I never used Lightroom, but saw a few screenshots, and I thought the concept of a workflow specifically for photographers was a good idea. So, I certainly wasn't hell bent on writing something better but instead wanted to help people get along in the computer world without Microsoft and Adobe.

LM: *What other features are you planning for vkdt?*

JH: Some examples: I'm currently working with David Tschumperlé (G'MIC) on porting his noise reduction software to the GPU. His code is based on a neural network and is therefore very computationally intensive. These things are becoming realistic in real time if you have fast underpinnings.

LM: *Where do you see vkdt five years from now?*

JH: I don't plan that far ahead. But I hope by then that GPUs on a par with the 3080 Ti will be very widespread and that the matching programming technique will be mainstream. Then software like vkdt can be delivered without any worries. In darktable there is this strange system with OpenCL, that you don't even dare to link at all, because it means that many users can't install GPU drivers.

LPM: *Would you still like to have co-developers for vkdt? If so, maybe not too many, right, because that would make the code too confusing?*

JH: You're right about the developers. The darktable code has not necessarily improved for having many developers. Sure, the community has had fun adding new features all the time. But maintaining this kind of code is a pain. If vkdt develops into a larger community project, I would actually be inclined to apply slightly stricter standards for contributions.

LM: *You have three small children and a university career – where do you even find the time and energy for vkdt?*

JH: Three kids yes, a career maybe not: I'm not aiming for a management position and don't earn a lot of money. Besides, I also need framework code for GPU applications at work, so there are synergy effects. Most people who finish their computer science studies with a doctorate probably have jobs with higher salaries and greater influence. But, of course, we are in teaching, and I also supervise PhD and MSc/BSc students.

LM: *Then let's hope you at least continue to enjoy your work and make rapid progress with vkdt. Thank you very much for the friendly talk!*

space, per ITU-R recommendation BT.2020 [12] (Rec. 2020 or BT.2020), which defines various aspects of ultra high-definition TV (UHDTV) such as screen resolution (4K/8K), frame rate (24fps/120fps), and color depth (10/12-bit), and RGB color space. This means that the original image data is retained for longer and the modules work more efficiently, especially when reconstructing overexposed or underexposed image areas. In addition, there is no need for constant reconversion between color spaces, which prevents information loss.

Getting Started

There are already precompiled binary packages of vkdt for most common distributions, and you can easily set them up using your Linux version's package manager. However, for Ubuntu 20.04, Linux Mint 20.x, and Arch Linux (plus its derivatives such as Manjaro), you still have to compile the program.

The following tutorial is therefore divided into several sections: First I will look into installing or compiling vkdt, before moving on to describe how to set up color management and introduce the main editing tools. Finally, I will look into configuring the graph and creating camera profiles for the noise reduction filter.

OBS Repository

The vkdt developer provides packages for many common distributions via the Open Build Service (OBS). But you first need to install the driver for your graphics card, including the Vulkan components. Open source drivers are usually installed automatically when you set up the operating system. However, you often have to install the proprietary driver retroactively for NVIDIA cards. In most cases, the Vulkan driver files come on the disc along with the driver, but sometimes you have to add individual packages.

Next you call the OBS repository page [13] for vkdt in the web browser and click on the icon for the operating system you are using. Two links will then appear lower down. Use the first, *Grab repository and install manually*, to add the repository to the system's software sources. This is the recommended approach so that you can update vkdt easily later on. Lower down on the page, you will now see instructions on how to add the repository to the package sources and install vkdt. Open a terminal window, copy the four command lines from the OBS repository page one after the other, paste them in the terminal, and press the Enter key each time to confirm that you want to run the commands.

Alternatively, you can download an installation package directly. To do this, click *Grab binary packages directly* and then click on the operating system version you are using. In the download dialog that opens, select *Save file* and click *OK*. The package usually ends up in the `~/Downloads/` folder. Open a terminal, change to this directory, and type `ls -l` to display its content. The name of the vkdt package always starts with `vkdt_`, but the rest changes depending on when it was created and on the package format.

Listing 1 shows the command for installing from the filesystem on a system with Debian package management. Apt checks all dependencies and adds them if necessary. However, the vkdt developer tries to keep the number of dependencies as low as possible. You may find out that Apt does not display many dependencies, or none at all, and instead just installs vkdt.

Listing 1: Installing the vkdt Package

```
$ sudo apt install ./vkdt_0~git1637146454.a12c186-0_amd64.deb
```

Compiling vkdt

Basically, compiling vkdt does not cause any major worries. The list of dependencies on GitHub is sort of complete but often does not give you the exact package name. And the name may not be easy to locate. This is why I compiled the program on Ubuntu 20.04 and on Manjaro and tried to put together more accurate and complete dependency lists (Listing 2 and Listing 3). For Ubuntu 20.04 and systems based on it, you need to install the Vulkan SDK from LunarG [14] first. The Vulkan version in the Ubuntu repository is outdated.

After setting up the dependencies, download the vkdt source code from GitHub. You may need to install `git` first for this. Then open a terminal window and enter the commands from Listing 4. `git` will then create a directory `vkdt/` and copy the source code of the program into it. Change to the folder and then load the submodules (Listing 4, lines 3 and 4).

Be sure to compile vkdt with Exiv2 support. The software will only recognize the orientation of the photos after you've done so (and be able to create camera profiles for the noise reduction filter). To do this, create a `config.mk` file in `vkdt/bin/`. Open `config.mk.defaults`, which is in the same directory, copy its contents, and paste it into `config.mk`. Now, find the lines `# vkdt_USE_EXIV2=1` and `# export vkdt_USE_EXIV2` in `config.mk`. Uncomment both by removing the hashtag at the start of the line, and then save `config.mk`.

Then change to the `vkdt/bin/` directory and compile the program by calling `make`. Compiling will take a few minutes even on fast hardware. If all goes well, you will end up with several new files

Listing 2: Ubuntu Dependencies

```
$ sudo apt install libvulkan-dev glslang-tools glslang-dev libglfw3 libglfw3-dev libimgui-dev libpugixml-dev libpugixml1v5
   libstdc6++ libstdc++-10-dev libstdc++-9-dev libjpeg-turbo-progs libjpeg-turbo8 libjpeg-turbo8-dev libjpeg8 libjpeg8-dev
   zlib1g zlib1g-dev libjpeg-dev libomp-10-dev libomp5-10 libomp-dev make pkg-config clang libclang-dev rsync cmake
   libexiv2-dev libexiv2-27
```

Listing 3: Arch Linux and Manjaro Dependencies

```
$ sudo pacman -S vulkan-headers glslang glfw-x11 pugixml libstdc++5 zlib libjpeg-turbo openmp make pkgconf clang rsync cmake exiv2
```

and subdirectories in `/vkdt/bin/`, including the `vkdt` program file. Last but not least, add the `vkdt/bin/` path to the `PATH` variable in your `.bashrc` to be able to conveniently start the program from anywhere without specifying the path.

Listing 4: Downloading the Source Code

```
01 $ git clone https://github.com/hanatos/vkdt.git --recursive
02 $ cd vdkt
03 $ git submodule init
04 $ git submodule update
```

Getting Acquainted

Your best bet is to start the program in a terminal window by typing the `vkdt` command, followed by the definition of the path on which the image files and/or camera raw data reside (e.g., `vkdt ~/pictures/`). This opens a window with the management module, in which vkdt builds thumbnails of the photos. They take up the major part of the window.

On the right, you will see a sidebar from which you can access various administrative functions (Figure 1). There are the *settings*, *collect*, and *tags* groups. The *settings* section does not offer too many functions as of yet; you will only see the *hotkeys* button. If you click on it, a dialog appears to let you define keyboard shortcuts, but this does not work smoothly at this time.

To get started, the *collect* module is more important. Press the *open directory* button at the bottom to add more directories with photos to the software database. A dialog now appears; it is basically a very simple file manager. However, it does not display files, only directory names. One of the folders is always marked with a *[d]*. Clicking *[d] ?* takes you to the parent directory. Then change to a directory by single-clicking on its name to navigate to a folder containing photos. Then press *OK* bottom right to add the folder's contents to the database. The file manager closes and the new thumbnails start to appear in the window.

If you click on one of the thumbnails, vkdt displays additional function groups in the sidebar.

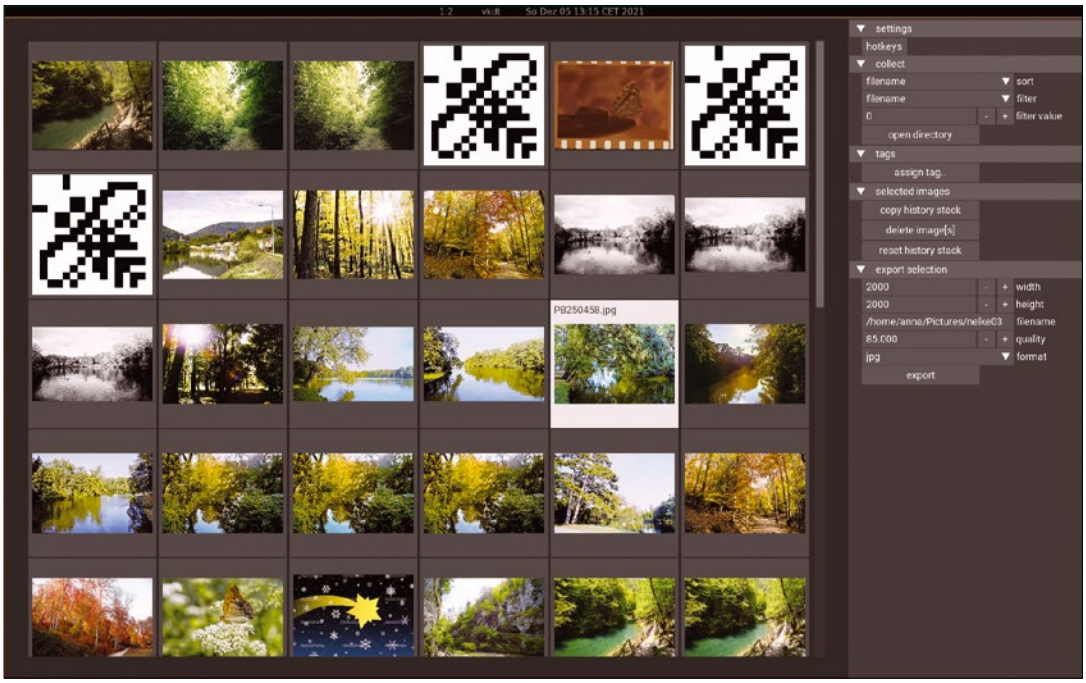


Figure 1: The vkdt user interface is reminiscent of darktable.

Figure 2: The editing module consists of a large preview window, a waveform histogram, and the tool panel.

For example, you can tag the photo using the *assign tag* button in the *tags* group. Double-clicking a thumbnail opens the photo in the Edit module and displays a large preview (Figure 2). You can zoom into the photo in the usual way with the mouse wheel or single-click with the center mouse button to scale the view to 100 percent. You can click and drag to move the view. Then press the center mouse button several times to return to the full-screen view. Press Esc to switch back to the administration module.

Color Management

Before you start editing photos, you need to set up color management. You do this with a small Python script named `read-icc.py` in the `vkdt/bin/` folder. To run the script, Python must be in place on the system. For Debian and Ubuntu, the required packages go by the names of *python3-minimal* and *python3-numpy*. The script is missing from the DEB and RPM vkdt packages; you need to download from GitHub in this case.

Change to the `/vkdt/bin` directory and run `read-icc.py` when you get there, using the profile for your monitor including the directory path as a command-line argument (Listing 5, lines 1 and 2). The tool reads the data from the monitor profile and creates a small text file named `display.profile` in `vkdt/bin/`. In the next step, you may need to customize the filename and copy the file to the correct directory. The `display.profile` filename must contain the xrandr monitor name so that vkdt can work with it. This is the name that vkdt will show you at startup time.

Then call vkdt in the terminal (line 4), minimize the program window and go back to the terminal where you started vkdt. You will see various messages there (lines 6 and 8). The screen name is `HDMI-0` on my lab system. I changed the second part of the name to `HDMI-0` to reflect this in the file we just created, `display.profile`. The full filename is now `display.HDMI-0`. Vkdt can handle up to two monitors with profiles.

If you installed vkdt on Debian with Apt, you need to copy the renamed file to the `/lib/vkdt/` directory; you need root privileges for this. If you compiled the application from source code yourself, leave the profile file where it is. The next time you launch the program, the message regarding the missing monitor profile should have disappeared.

There are many different types of monitor profiles. `read-icc.py` will occasionally be unable to use a profile because it turns out to be too complex. I recommend that you use DisplayCAL [15] to create a profile for your monitor and select the simplest profile type, *single gamma + matrix*, in the profile settings there.

Listing 5: Color profiles

```
01 $ cd vkdt/bin/
02 $ ./read-icc.py /<path>/<profile>.icc
03 [...]
04 $ ./vkdt ~/images/
05 [...]
06 [gui] monitor [0] HDMI-0 at 0 0
07 [...]
08 [gui] no display profile file display.HDMI-0, using sRGB!
```

Developing RAWs

The editing module provides access to the editing tools via the wide sidebar to the right of the preview window. You will find the *favourites*, *tweak all*, *pipline config*, and *esoteric* tabs here.

The *favourites* tab gives you rapid access to important settings such as image brightness or clarity. The *Tweak all* tab lets you adjust the settings of all active nodes with more granularity (Figure 3), and you can configure the vkdt node graph in *pipeline config*. This is where you can enable or disable individual nodes, or move them to the back or front. The *esoteric* tab gives users access to the more unusual features such as presets or playing animations.

Under the *tweak all* tab, the most important node here is *colour 01*, where you adjust the image brightness (i.e., the exposure and white balance). Click on it to expand the node and display the adjustment options. The first slider, named *exposure*, is responsible for brightness. Slide it to the right to lighten the photo or to the left to darken it. The three sliders lower down, *white red*, *white green*, and *white blue*, adjust the white balance.

By default, vkdt takes care of the camera's white balance, which is often slightly yellowish. If this is

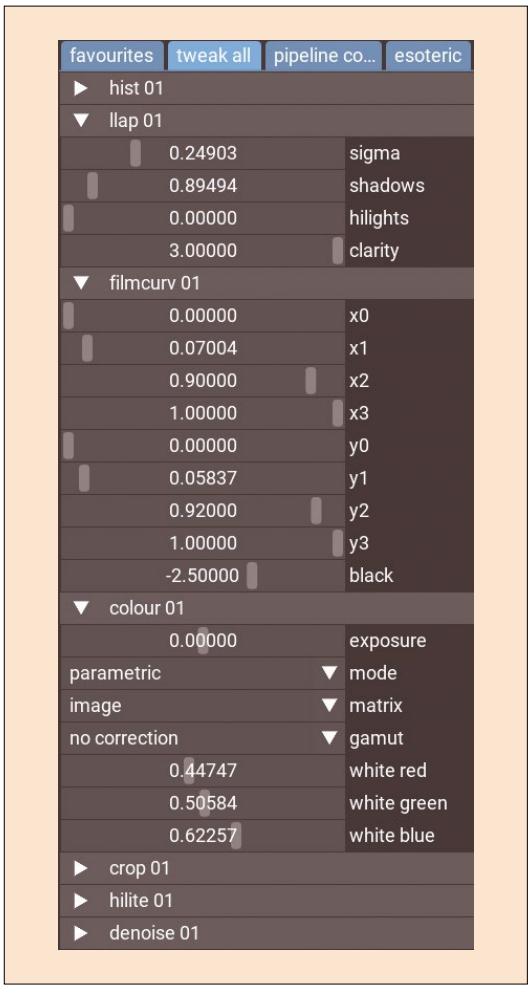


Figure 3: The main modules for adjusting contrast, brightness, and color are *colour*, *filmcurv*, and *llap*.

the case, you need to move the *white blue* slider to the right to enhance the blue component and counteract the yellow cast. In most cases, this will first create a different color cast, such as a red cast. Then move the *white red* slider a little to the left. Another possibility would be to lower the red and green components. You can correct a green cast either by reducing the amount of green or by increasing the amount of red or blue. You can correct a cyan cast by increasing the red value.

The *filmcurv 01* and *llap 01* modules are responsible for contrast enhancement and partly also for color intensification. Disable them and you will see the unprocessed, low-contrast, and colorless RAW image. In the case of *filmcurv 01*, this is a tone curve that vkdt applies to the image. By default, the S-shaped curve brightens the light tonal values and darkens the dark ones, resulting in an overall contrast improvement. Because the module works in the RGB color space, the colors are intensified at the same time.

The values *x0* through *x3* and *y0* to *y3* represent four points on this curve. *x0* and *x1* and *y0* and *y1* are in the dark area; *x2* and *x3* and *y2* and *y3* are in the light zones. The *x* values move the points to the left or right, and the *y* values move them up or down. If you move the points to the right or up, this lightens the tonal value areas in question. Dragging them down or left makes them darker. Using the *black* slider, you can also adjust the brightness by applying a logarithmic curve. If you change this, the image is brightened or darkened, with the darker pixels being affected more by the change in brightness.

The local Laplacian filter *llap 01* changes the local contrast of the image, which is also known as the clarity (Figure 4). Drag the *clarity* slider to the right to increase the local contrast. However, this mainly changes the clarity for pixels of medium brightness. You can use the *sigma* value to determine how bright or dark they can become. If you drag the slider to the right, the algorithm evaluates more pixels as midtones. You can use the *shadows* and *hilights* sliders to adjust the local contrast in the brightest and darkest areas of the image.

Using the *crop 01* module not only lets you crop the photo but also straighten it and correct the perspective (Figure 5). Drag the *rotate* slider a little to the right to rotate the image to the left. If you want to tilt the image to the right, first move the slider all the way to the right and then a little to the left. To crop the photo, first enter the length-to-width ratio in the *aspect ratio* input box. A value of *1* creates a square cropping frame, while *1.5*

Figure 4: The *llap* module uses image pyramid technology to change the local contrast.

creates a landscape frame with a ratio of 3:2, and *0.75* a portrait frame with a ratio of 4:3.

Then click the *crop:crop start* button. Move the mouse pointer to a position near the right or bottom edge of the crop frame, press the left mouse button, and drag the mouse left, right, up, or down to resize the frame. If you want to change the vertical position of the frame, move the mouse cursor to the frame's upper edge, press the left mouse button, and move the mouse up or down. To position the rectangle horizontally, position the mouse cursor on the left edge of the rectangle, press the left mouse button, and drag the mouse to the left or right. Once you are satisfied with how you have set the size and position of the frame, click *crop:crop done*.

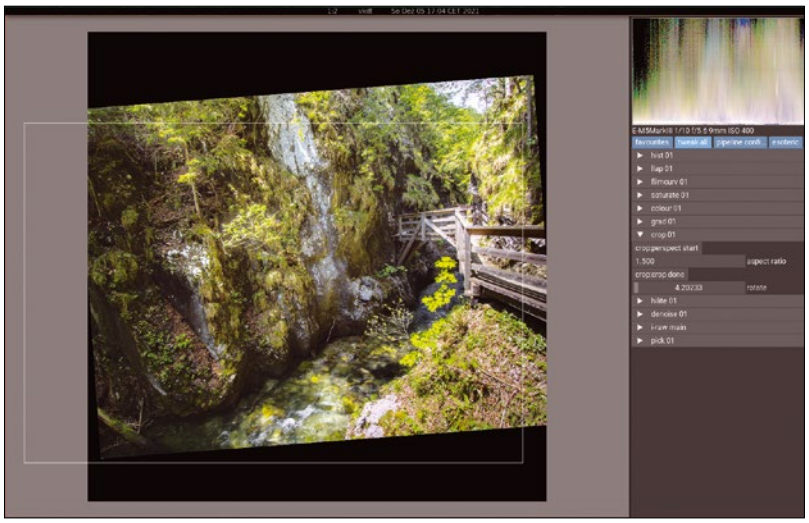


Figure 5: Using the Rotate and Crop tool takes some getting used to.

Figure 6: In *pipeline config*, you can move, insert, and disable modules.

Configuring the Node Graph

There are two important nodes missing from the default vkdt graph: color saturation tuning and sharpening. To add them to the graph later on, switch to the *pipeline config* tab (Figure 6). You will see two lists containing modules and nodes. The upper list contains all of the active modules, but not all of them are editing modules. For example, *i-raw main* is an input module, and *display main* and *display hist* are visualization and review modules. Curved arrows connect the active modules to each other.

Expand the *llap 01* and *filmcurv 01* modules. You will now see several buttons for each of the modules. On the far right you have *input* and *output*, where the *output* from *filmcurv 01* is connected to the *input* for *llap 01*. To the left, you will see *move up* and *move down*. If you click *move down* on *llap 01*, the whole module moves down one level to sit below *filmcurv 01*. There is a *disconnect* button below *move up*. This disables modules, which then appear in the *disconnected* list at the bottom. You will also see the *saturate 01*, *lens 01*, and *zones 01* modules there.

Expand the *saturate 01* module and click *insert before*. The button turns red, and a new button named *before this* appears in the top module list for the expanded modules (Figure 7). Click on *filmcurv 01*. The *saturate 01* module now inserts itself between *filmcurv 01* and *colour 01*. Switch back to the *tweak all* tab, and you'll see *saturate 01* there too between *colour 01* and *filmcurv 01*. Expand the module and drag the slider to the right to increase the intensity of the colors.

The module used to sharpen photos is named *deconv* because it uses a deconvolution algorithm. However, it does not initially appear in the *disconnected* list. You can add the module to the list by selecting it in the *pipeline config* tab at the very bottom of the drop-down list in *module* and then clicking *add module*. Now expand *deconv 01*, click *insert before*, and add it under *crop 01*.

This node is not actually a conventional sharpening filter. It is only intended to counteract any blur of the image during image capture, for

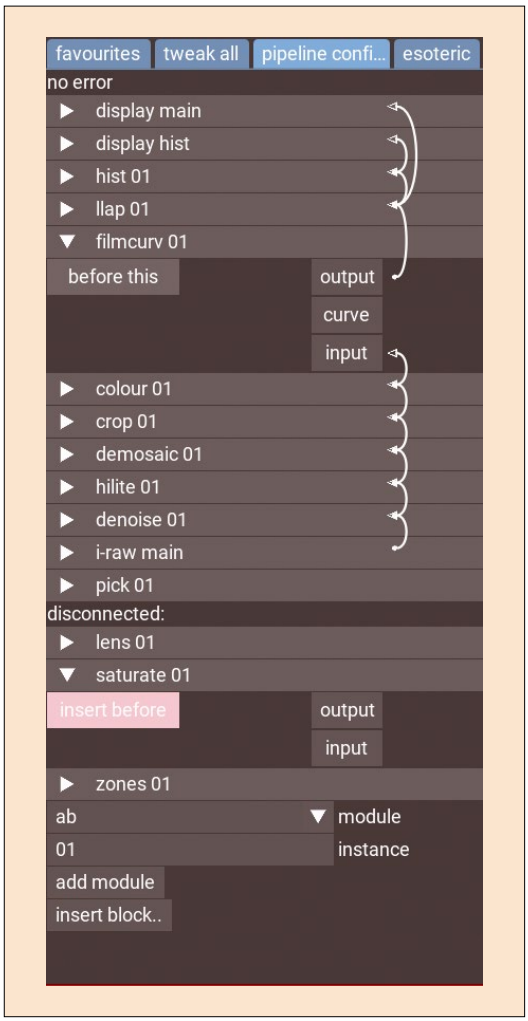


Figure 7: To insert a module, first click the *insert before* button and then click *before this*.

example, due to diffraction blur. (Diffraction blur refers to a reduction of sharpness in optical images due to the diffraction of light at the aperture of a camera, telescope, or microscope lens.) This is why the module has two controls. The *sigma* value lets you specify how pronounced the blur of the image was during image capture, while *iterations* lets you specify how often vkdt applies the algorithm to the image: the higher the two values, the stronger the sharpening effect.

It is relatively easy to make the vkdt pipeline unusable, so be careful when moving, inserting, and deactivating modules. Not every node can reside at any location. For example, *demosaic 01* must not be at the end of the graph because it handles descreening of the RAW image. If you insert an effect in the wrong place, this disables other nodes and the preview image turns black. In this case, you need to reset the graph to the default. To do this, press Esc to return to the management module and select the image for which you want to reset the graph. Then expand the *selected images* module in the sidebar on the right and click *reset history stack*. Proceed in the same way if you are dissatisfied with the editing results and want to start all over again.

Removing Image Noise

We have not looked at one important module so far: *denoise (01)* suppresses image noise. It has two sliders. Use *strength* to specify the extent to which the module suppresses noise in general and *luma* to control whether it also removes luminance noise. The further you move this slider to the right, the stronger the noise reduction. Be careful here: If you suppress luminance noise too heavily, you will lose image detail. If the slider is on the far left, *luma* will only reduce color or chrominance noise.

Noise reduction does not work correctly at first. If you drag *strength* to the right, the image in the preview window turns black or green, or you see artifacts such as colored dots. This is because you have not yet created noise profiles for your camera. Vkdt uses these profiles because the camera sensor noise depends on several factors, including the brightness of the pixels and the ISO value. Vkdt first examines how much noise different brightness pixels produce for different ISO values.

To create noise profiles use the bash `noise-profile.sh` script in the `vkdt/bin` directory. If you installed vkdt from the OBS repository, you will need to do a little customization work. Open the script with a text editor and look for the `vkdt-cli` string; it appears twice in the script. Now remove `./` in front of `vkdt-cli` and save the file. Otherwise, the script would look for `vkdt-cli` in the `vkdt/bin/` directory. But because you did not compile the program, it is not there.

Then run the script and use the raw file you are editing as the command line argument (Listing 6). This will create several files. In `vkdt/bin`, you will find a file with a suffix of `nprof.jpg`. Its name is composed of your camera's name and the ISO value of the photo. In `vkdt/bin/data/` there is a new `nprof/` folder that contains a file with the same extension. The program again named this file after your camera and the ISO value of the photo. If you installed vkdt as a DEB package, you need to recopy all these files and folders to `/lib/vkdt/`.

If you restart vkdt now, noise reduction should work correctly. Once you have created noise profiles, you can use them for other photos. Note, however, that you need a separate profile for each ISO value.

Listing 6: Noise Profile

```
$ cd /vkdt/bin/

$ /noise-profile.sh /home/username/Pictures/filename.raw
```

Exporting Photos

After doing all that work, you will want to save the results of your efforts. To do so, press Esc to switch back to the administration module and select the image you want to export. Then expand the *export selected* module in the sidebar on the right. In the *width* and *height* input fields, type the size you want the exported photo to be. Leave the values at 0 if you do not want to change the image size.

For *filename*, specify the filename of the photo, including the directory path. If you want to save the photo in a lossless format for further editing, select PFM [16] as the *format*. Finally, click *export*. Vkdt automatically converts JPEGs to sRGB; it saves PFMs in the linear Rec. 2020 color space. However, the program does not embed color profiles in the files, and PFM files are likely to be upside down.

Outlook

Vkdt offers a variety of other editing options, including drawn masks that let you apply an effect to certain areas of the image only. You can also use vkdt to edit Magic Lantern videos (i.e., moving images in RAW format that were captured with Canon's EOS cameras). Vkdt also has rudimentary animation and rendering capabilities.

Conclusions

The results the new vkdt RAW developer delivers are not yet as good as what darktable or RawTherapee offer – of course, hundreds of developers have worked on both competitors. The vkdt user interface in particular takes some getting used to, and it shows that the developer, Johannes Hanika, is not necessarily a user interface specialist.

The software is currently aimed less at professional photographers or people moving away from commercial programs, but rather at technology lovers who enjoy taking pictures and at image editors who like to explore. The program is already quite fun to work with, and it already seems more mature than the first versions of darktable. Vkdt has the potential to become the best RAW converter, especially if more developers can be found and are capable of, say, porting the new selection functions from RawTherapee. ■■■

Info

[1] darktable: *https://www.darktable.org*

[2] RawTherapee: *https://www.RawTherapee.com*

[3] ART: *https://github.com/Benitoite/ART*

[4] LightZone: *https://github.com/ktgw0316/LightZone/releases*

[5] PhotoFlow: *https://github.com/aferrero2707/PhotoFlow*

[6] Filmulator: *https://filmulator.org*

[7] Vulkan: *https://www.vulkan.org*

[8] vkdt on GitHub: *https://github.com/hanatos/vkdt*

[9] vkdt: *https://jo.dreggn.org/vkdt/*

[10] Nuke: *https://www.foundry.com/products/nuke-family/nuke*

[11] Natron: *https://natrongithub.github.io*

[12] Rec. 2020: *https://en.wikipedia.org/wiki/Rec._2020*

[13] vkdt (OBS): *https://software.opensuse.org/download.html?project=graphics%3Adarktable%3Amaster&package=vkdt*

[14] Vulkan SDK from LunarG: *https://vulkan.lunarg.com/sdk/home#linux*

[15] DisplayCAL: *https://displaycal.net*

[16] Portable FloatMap: *http://www.pauldebevec.com/Research/HDR/PFM/*

LINUX NEWSSTAND

Order online:
https://bit.ly/Linux-Newsstand

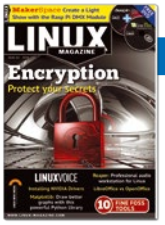
Linux Magazine is your guide to the world of Linux. Monthly issues are packed with advanced technical articles and tutorials you won't find anywhere else. Explore our full catalog of back issues for specific topics or to complete your collection.

#258/May 2022

Clean IT

Most people know you can save energy by changing to more efficient light bulbs, but did you know you can save energy with more efficient software? This month we examine the ongoing efforts to bring sustainability to the IT industry.

On the DVD: Manjaro 21.2 Qonos and DragonFly BSD 6.2.1



#257/April 2022

Encryption

This month, we survey the state of encryption in Linux. We look beyond the basics to explore some of the tools and technologies that underpin the system of secrecy – and we show you what you need to know to ensure your privacy is airtight.

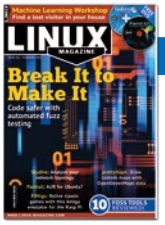
On the DVD: Linux Mint 20.3 Cinnamon Edition and deepin 20.4

#256/March 2022

Facial Recognition

Biometrics got a boost recently with the arrival of Microsoft's Hello technology. Now the open source world is catching up, with an innovative tool appropriately called Howdy. Facial authentication might not be ready for the CIA yet, but we'll help you get started with Howdy and explore the possibilities of authenticating with a glance.

On the DVD: antiX 21 and Haiku R1/ Beta 3



#255/February 2022

Break It to Make It

Fuzz Testing: Ever wonder how attackers discover those "carefully crafted input strings" that crash programs and surrender control? Welcome to the world of fuzz testing. We introduce you to the art of fuzzing and explore some leading fuzz testing techniques.

On the DVD: Parrot OS 4.11 and Fedora Workstation 35

#254/January 2022

Phone Hacks

Eventually phone manufactures just give up on supporting old hardware. If you're not ready to abandon that hardware yourself, you might find a better alternative with LineageOS — a free Android-based system that supports more than 300 phones, including many legacy models that are no longer supported by the vendor. We also explore PostmarketOS, a community-based Linux distribution that runs on several Android devices.

On the DVD: Ubuntu 21.10 and EndeavourOS 2021.08.27

#253/December 2021

OpenBSD

BSD Unix has been around longer than Linux, and it still has a loyal following within the Free Software community. This month we explore the benefits of a leading BSD variant from the viewpoint of a Linux user.

On the DVD: Tails 4.22 and Q4OS 4.6

FEATURED EVENTS

Users, developers, and vendors meet at Linux events around the world. We at *Linux Magazine* are proud to sponsor the Featured Events shown here.

For other events near you, check our extensive events calendar online at *https://www.linux-magazine.com/events*.

If you know of another Linux event you would like us to add to our calendar, please send a message with all the details to *info@linux-magazine.com*.

NOTICE

Be sure to check the event website before booking any travel, as many events are being canceled or converted to virtual events due to the effects of COVID-19.

openSUSE Conference 2022

Date: June 2-4, 2022

Location: Nuremberg, Germany

Website: *https://events.opensuse.org/conferences/oSC22*

The openSUSE Conference is the annual openSUSE community event that brings people from around the world together to meet and collaborate. The organized talks, workshops, and BoF sessions provide a framework around more casual meet ups and hack sessions.

Open Source Summit North America

Date: June 21-24, 2022

Location: Austin, Texas and Virtual

Website: *https://events.linuxfoundation.org/open-source-summit-north-america/*

Open Source Summit is the premier event for open source developers, technologists, and community leaders to collaborate, share information, solve problems, and gain knowledge, furthering open source innovation and ensuring a sustainable open source ecosystem. It is the gathering place for open-source code and community contributors.

Events

Kubernetes on EDGE Day Europe	May 16-17	Valencia, Spain	https://events.linuxfoundation.org/
KubeCon + CloudNativeCon Europe 2022	May 16-20	Valencia, Spain	https://events.linuxfoundation.org/
Future.HPC 2022	May 17-18	Virtual Global Event	https://events.altair.com/future-hpc/
ISC High Performance 2022	May 29-June 2	Hamburg, Germany	https://www.isc-hpc.com/
openSUSE Conference 2022	June 2-4	Nuremberg, Germany	https://events.opensuse.org/conferences/oSC22
OpenJS World 2022	June 6-10	Austin, Texas	https://events.linuxfoundation.org/openjs-world/
cdCon	June 7-8	Austin, Texas + Virtual	https://events.linuxfoundation.org/cdcon/
SUSECON Digital 2022	June 7-9	Virtual Event	https://www.susecon.com/
Storage Developer Conference (SDC EMEA)	June 14	Virtual Conference	https://www.snia.org/events/sdcemea
ODSC Europe 2022	June 15-16	London, UK + Virtual	https://odsc.com/europe/
ITEXPO Florida	June 21-24	Fort Lauderdale, Florida	https://www.itexpo.com/east/
Open Source Summit North America	June 21-24	Austin, Texas + Virtual	https://events.linuxfoundation.org/
Xen Developer & Design Summit	June 28-30	Bucharest, Romania + Virtual	https://events.linuxfoundation.org/xen-summit/
USENIX ATC '22 & OSDI '22	July 11-13	Carlsbad, California	https://www.usenix.org/conference/
The Open Source Infrastructure Conference	July 19-20	Berlin, Germany	https://stackconf.eu/
GUADEC 2022	July 20-25	Guadalajara, Mexico	https://events.gnome.org/event/77/
Icinga Camp Berlin 2022	July 21	Berlin, Germany	https://icinga.com/community/events/icinga-camp-berlin-2022/

Images © Alex White, 123RF.com

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to *edit@linux-magazine.com*.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at:
http://www.linux-magazine.com/contact/write_for_us.

Authors

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Copy Editors
Amy Pettle, Aubrey Vaughn

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
Lori White

Cover Image
© skorzewiak, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7679420

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Publisher
Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the disc provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2022 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media USA, LLC, unless otherwise stated in writing.

Linux is a trademark of Linus Torvalds.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by hofmann infocom GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Approximate
On Sale Date

UK / Europe	Jun 04
USA / Canada	Jul 01
Australia	Aug 01

Please note: On sale dates are approximate and may be delayed because of logistical issues.

Issue 260 / July 2022

Privacy

You can't really hide on today's Internet – unless you get serious about protecting your privacy. Next month we look at some recent tools for staying out of sight.

Lead Image © Pavlo Syvak, 123RF.com

Preview Newsletter

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: ***https://bit.ly/Linux-Update***

ISC IS BACK IN PERSON!

TRANSFORMING THE FUTURE

CONFERENCE

PARALLEL PROGRAMMING

SYSTEM ARCHITECTURE

WORKSHOPS

MACHINE LEARNING

EXHIBITION

QUANTUM COMPUTING

APPLICATIONS & ALGORITHMS

AND MORE...

TUTORIALS

Come join 3,000 high performance computing (HPC) practitioners, vendors, and enthusiasts.

WHAT TO EXPECT?

- Technical program (in person and online)
- HPC exhibition (in person and online)
- Reunion with peers, friends, and collaborators
- Safe and health-protocol-compliant environment

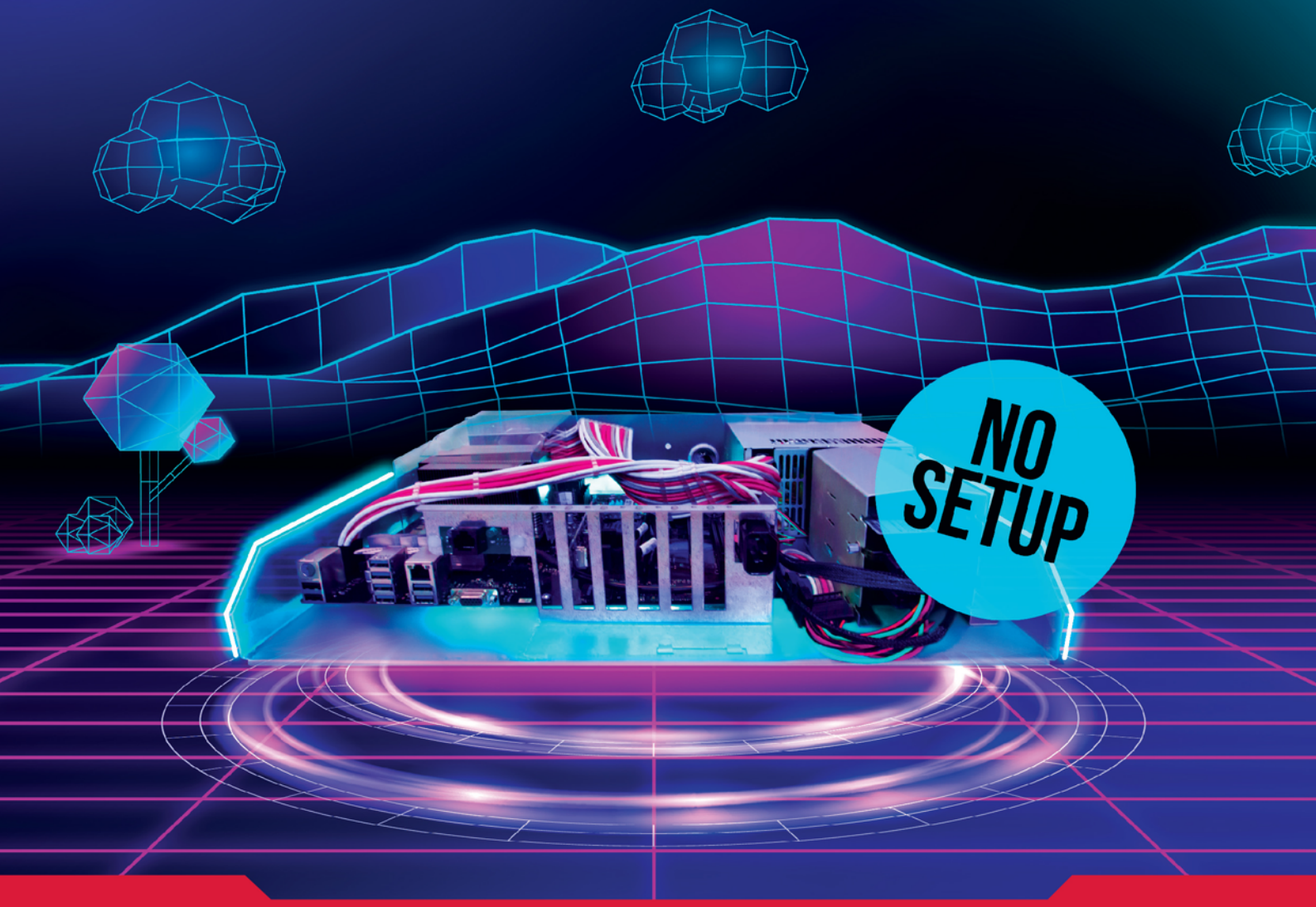
After two years of an online presence ISC returns to you as an in-person event. If you're involved or interested in HPC, don't miss out on making valuable connections at ISC 2022.

ISC High Performance
The HPC Event.

ISC 2022 | MAY 29 – JUNE 2, 2022
HAMBURG, GERMANY isc-hpc.com

HETZNER

NO SETUP FEE
SEASONS SPECIAL



SEASONS SPECIAL
Save up to $109.50!

Ready for new beginnings? So why not start a new project?

Expand your infrastructure with our AX dedicated root servers with AMD Ryzen™ CPUs and benefit from mind-blowing performance and our no setup offer on top!

AMD RYZEN

e.g. Dedicated Root Server AX51-NVMe

- ✔ AMD Ryzen™ 7 3700X Simultaneous Multithreading
- ✔ 64 GB DDR4 ECC RAM
- ✔ 2 x 1 TB NVMe SSD
- ✔ 100 GB Backup Space
- ✔ Unlimited traffic
- ✔ No minimum contract
- ✘ ~~Setup fee $65~~

Setup fee $ 0 Starting monthly at $ 62

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. This sale is good while supplies last. All rights reserved by the respective manufacturers.

hetzner.com/special