9 REASONS LINUX BEATS WINDOWS 10

LINUXVOICE

AN~~ARCHY~~

LINUX IN THE UK

How Britain is leading the charge in free software and open data

INSIDE SUSE'S OPEN BUILD SERVICE

+

KDENLIVE Edit professional-quality video
RASPBERRY PI Control the physical world!
SEARCH ENGINES Find one that respects your privacy

116 PAGES OF AWESOME

JOHN SULLIVAN

FREEDOM!
Richard Stallman's boss talks freedom, formats and the future

BLOGGING

WORDPRESS
Spread your immortal words across the web with a customised blog

FREE SOFTWARE | FREE SPEECH
ISSN 2054-3778
October 2015 £5.99 Printed in the UK
9 772054 377001 10

ANDREWS & ARNOLD LTD

will make you the

LINE KING

BE THE MASTER OF YOUR OWN TELEPHONE

SIP2SIM® from Andrews & Arnold allows you to treat your mobile handset as a SIP endpoint, freeing you from your desk and without the need of a smartphone app. Insert the SIM into your phone, point it to your Asterisk, FreeSwitch or other SIP server (or a commercial SIP service) and experience the reliability and call quality you're used to on a mobile, but with the flexibility of a SIP handset.

No minimum term. SIM £5+VAT and £2+VAT per month. Calls from 2p+VAT per minute.

Call 033 33 400 220, email sales@aa.net.uk or visit www.SIP2SIM.uk to find out more

HAKUNA MATATA

Polygon Windows

The October issue

LINUX VOICE

Linux Voice is different. Linux Voice is special. Here's why…

1 At the end of each financial year we'll give 50% of our profits to a selection of organisations that support free software, decided by a vote among our readers (that's you).

2 No later than nine months after first publication, we will relicense all of our content under the Creative Commons CC-BY-SA licence, so that old content can still be useful, and can live on even after the magazine has come off the shelves.

3 We're a small company, so we don't have a board of directors or a bunch of shareholders in the City of London to keep happy. The only people that matter to us are the readers.

THE LINUX VOICE TEAM
Editor Graham Morrison
graham@linuxvoice.com
Deputy editor Andrew Gregory
andrew@linuxvoice.com
Technical editor Ben Everard
ben@linuxvoice.com
Editor at large Mike Saunders
mike@linuxvoice.com
Games editor Michel Loubet-Jambert
michel@linuxvoice.com
Creative director Stacey Black
stacey@linuxvoice.com
Malign puppetmaster Nick Veitch
nick@linuxvoice.com
Editorial contributors:
Mark Crutch, Marco Fioretti, Juliet Kemp, John Lane, Vincent Mealing, Simon Phipps, Les Pounder, Mayank Sharma, Valentine Sinitsyn.

GRAHAM MORRISON
A free software advocate and writer since the late 1990s, Graham is a lapsed KDE contributor and author of the Meeq MIDI step sequencer.

I have a lot to be thankful to Microsoft for. Its operating system made computers affordable, despite itself, and one of my first jobs was looking after some Windows 3.0 machines. But it was also what led me to Linux. In the late 1990s, I was trying to get back into programming, and wanted to write something that would specifically scratch my own itch. But I wanted my code to also be useful to anyone else with the same problem. I was genuinely shocked by the restrictive nature of what you could and couldn't share when you developed software on Windows using its own developer tools and APIs.

It was this realisation that led me back to Linux and Free Software (after a previously bad experience trying to run X11 on an Amiga 4000). All these years later, Microsoft is a different company. It attends open source events and has its own well-maintained GitHub account. But it still doesn't believe in open source enough to open more of its own code, or spend time opening its licences. To me, it's still opposed to sharing great ideas, and that, dear Linux Voice readers, is our killer feature.

Graham Morrison
Editor, Linux Voice

SUBSCRIBE ON PAGE 64

What's hot in LV#019

ANDREW GREGORY
"Running your own website is exciting and liberating, which is why *WordPress* makes such a great platform and subject." **p84**

BEN EVERARD
"Our feature on Windows 10. Any new Windows release is an ideal opportunity for reminding people why Linux is so awesome. " **p34**

MIKE SAUNDERS
"Us Brits seldom take pride in our history, but delving into our long computing heritage has been fascinating." **p20**

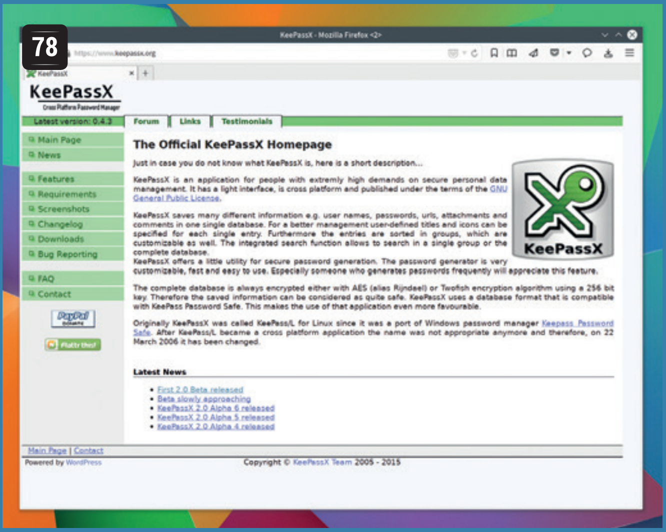
CONTENTS

October LV019

Celebrating a year and a half of server uptime, it's your new Linux Voice...

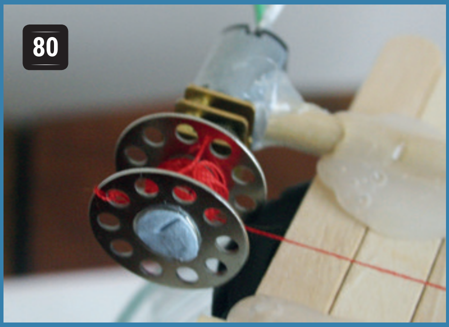
SUBSCRIBE ON PAGE 64

TUTORIALS



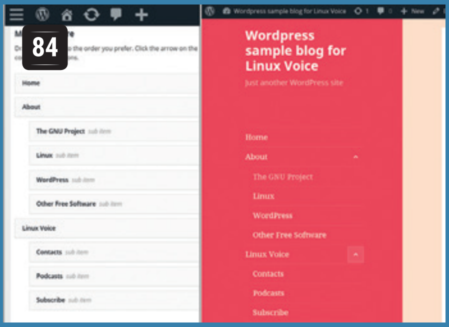
78 **Manage passwords with KeePassX**

Life online brings too many passwords to remember – so set up a password manager and sync your secrets across all your devices.



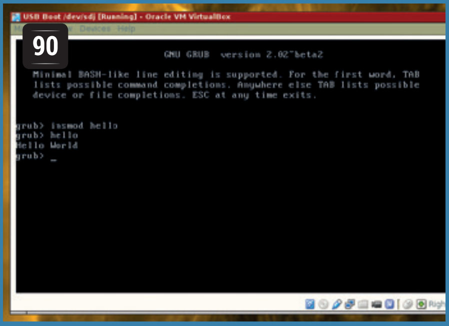
80 **Raspberry Pi: Write a controller for a crane**

Interact with the physical world using a HAT and some Python.



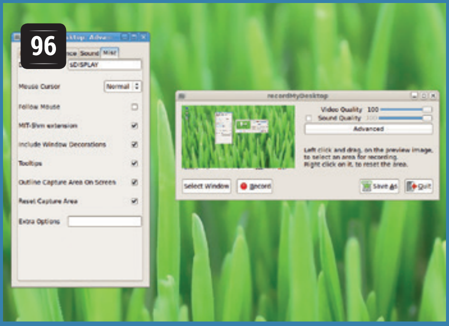
84 **WordPress: Build a custom blog**

Control every aspect of this flexible, simple CMS.



90 **Grub 2: Add a custom command**

Extend the functionality of the Linux bootloader.



96 **Kdenlive: Edit video with Free Software**

Turn raw footage into a masterpiece.

100 **Old code: Prolog**

Natural language for computers.

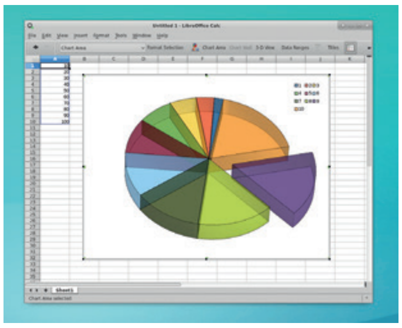
104 **Code Ninja: GNU parallel**

Keep your CPU cores happy.

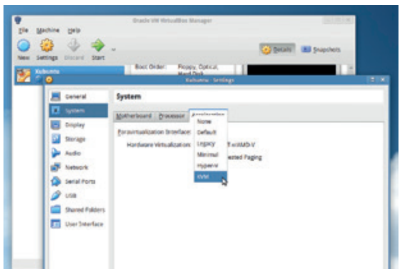
106 **Faster code with profiling**

Fix the slow bits in your code.

REVIEWS



50 **LibreOffice 5.0**
The standard bearer for cross-platform Free Software keeps getting more refinement, more features and more polish.



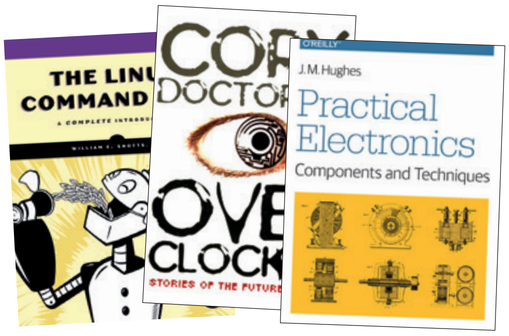
52 **VirtualBox 5.0**
Virtualise as many Linux distributions as your RAM banks can handle.

53 **Drawpile**
Collaborate on sketches, plans or games of snakes and ladders with distributed graphics.

54 **Geany**
Write your code in the luxury of this simple, clean integrated development environment.

55 **Shadow of Mordor**
Pipe-smoking don JRR Tolkien never played games on Steam, but he would have liked this.

56 **Books** *The Linux Command Line*: because sometimes the man page just isn't enough.



NEWSANALYSIS

The Linux Voice view on what's going on in the world of Free Software.

Opinion

On selling Open Source software

Freedom isn't free – and nor is Free Software, necessarily.

Simon Phipps is ex-president of the Open Source Initiative and a board member of the Open Rights Group and of Open Source for America.

One of the more common questions on the user support mailing lists and forums I follow for large open source projects concerns paying for open source software. When someone sees CDs for sale on eBay, for example, they assume that must be a scam. That old confusion in the terminology of free software leads them to believe that it's wrong to charge for software – this is just another of the problems that comes from the ambiguous term 'Free Software'.

As I've previously written here, there are indeed scams on the internet around open source software. But charging for the software itself is not one of them. It's allowed – implicitly in the Free Software Definition and explicitly in the Open Source Definition – and while it is not the objective, the freedom to charge money is an important consequence of the presence of software freedom. If you are not permitted to do so, that's a good diagnostic of its absence. As the GNU project says, "we encourage people who redistribute free software to charge as much as they wish or can. If a licence does not permit users to make copies and sell them, it is a non-free licence."

"The confusion in the terminology of Free Software leads people to believe it's wrong to charge for it ."

Being free to charge to deliver software is so fundamental that everyone has the freedom to do so – including for zero fees. That makes it uneconomic to charge unless you're adding a valuable service, as anyone can get the same software without paying you. The value of any commodity is decided at the margins, and if you're competing with something that's zero cost, the most you can charge is the same – zero. But you can charge if you add something of value. Burning a CD and mailing it to you, for example, may be a valuable service if you have poor connectivity.

A few projects do have concerns about you selling their software. Mozilla would rather you did not attempt to sell *Firefox* for example, and uses its trademark as a tool to prevent it. While there are excellent arguments on both sides, since *Firefox* truly is open source developers have taken the code, removed Mozilla's trademark and released the browser under names like *IceWeasel* and *IceCat* so that it can be included in Debian GNU/Linux. That's not so it can be sold; it's just to be absolutely sure there can be no question about Debian's open source credentials.

Filthy lucre

While it's definitely a basic part of software freedom, one reason people think it's not allowed is that it's uncommon in open source software. Even including adverts in software is considered poor taste in open source, and since any developer can get the same source code and simply remove them they are uncommon in open platforms.

This is how we can see that Windows 10, although apparently "free", is not free in any sense that actually delivers freedom. According to reports, the new version includes adverts in such basic applications as the calculator – and even in *Solitaire* – along with a demand for a subscription fee every year to stop them being displayed. With no access to the source code, there's no way round this. Seeing adverts like this is a pretty good indication that your software doesn't come with software freedom included!

"Windows 10, although apparently "free", is not free in any sense that actually delivers fredom."

You chose... poorly

So please don't write to complain when you see open source software for sale. It's perfectly legitimate to do so, and there's nothing any open source community can – or should – do to prevent it, apart from making the software freely available themselves to make the practice uneconomic. And please do tell any of your friends unfortunate enough to be stuck with a proprietary operating system that's trying to gouge them for pennies for every tiny app that they have a choice. With proprietary software and service, the old adage is true that if you're using something for free, you aren't the customer – you're the product. When even your calculator wants to charge you money, you know there's something wrong. You have a choice – use it.

North Korea • Debian • Fedora 23 • Microsoft • NSA • OwnCloud

CATCHUP

Summarised: the biggest news stories from the last month

1 Fedora 23 to get Cinnamon spin

By default, the Fedora distribution ships with Gnome 3 as its desktop environment, but "spins" are available that provide alternatives. They share the same Fedora base packages, but have KDE, Xfce, LXDE and Mate on top. One desktop that has been conspicuous by its absence, however, has been Cinnamon, as pioneered in Linux Mint. Well, Fedora 23, which is due to arrive around the end of October, will have its very own Cinnamon spin thanks to developer Dan Book. Thanks!

2 Microsoft gives big chunk of cash to OpenBSD

What a time to be alive. Microsoft, once generally regarded as the arch enemy of FOSS, is making steps to repair its relationship with the community. In July, the company became a gold sponsor of the OpenBSD foundation – which means it donated somewhere between $25,000 to $50,000. This helps Linux indirectly, as virtually every Linux distribution ships with *OpenSSH*, a tool developed by the OpenBSD project. Who knows what else the Redmond behemoth plans...

3 LibreOffice 5.0 coming up with Wayland support

LibreOffice 5.0 should be available by the time you read this, and will run on the Wayland compositor thanks to new *GTK 3* support.

4 Debian drops support for SPARC machines

SPARC isn't the most popular CPU architecture out there, but it's still crunching numbers in data centres around the world. SPARC was originally developed by Sun, and is now owned by Oracle. The Debian project has decided to remove SPARC support in upcoming releases, as few developers were spending time keeping it up to date. This isn't the end of open source Unix on SPARC, however: the NetBSD and OpenBSD projects will carry on supporting it for a while.

5 NSA releases open source security toolkit for Linux

The USA's National Security Agency doesn't have many fans in the FOSS world, after the uncovering of its massive spying and surveillance programmes. Still, if you fancy running some NSA code on your Linux box, try SIMP, the System Integrity Management platform. This provides a "complete management environment focused on compliance with SCAP profiles and industry best practice". Sounds awesome, right? **https://github.com/NationalSecurityAgency/SIMP**

6 Lenovo starts shipping Linux laptops in India

Finding laptops with Linux pre-installed isn't so easy in the west, but if you happen to live in India (or you're planning a trip there), you can now pick up a ThinkPad L450 for 48,000 rupees (that's about £480, or $750). That's around £100 cheaper than the Windows model, and the machine has an Intel Core i3 CPU, 4GB of RAM and a 500GB hard drive. It's bundled with Ubuntu. Hopefully the machines will be a success and we'll see them in other markets.

7 North Korean distro watermarks files

Red Star Linux, the official distribution of hermit kingdom North Korea, isn't particularly freedom friendly to its users. Security analyst Florian Grunow has discovered that the operating system watermarks files such as office documents and images with hidden codes, based on the hardware's serial numbers. In theory, this should make it easier for Kim Jong-Un and his cronies to discover the source of any anti-regime content. **http://tinyurl.com/redstarlol**

8 OwnCloud 8.1 released with better performance

OwnCloud is rather good for hosting your own "cloud" (ie document collaboration and file sharing), but has been lacking in certain areas. The new 8.1 release is a step forward: the documentation has been greatly improved and more tightly integrated with the interface, making it more welcoming for new or non-tech-savvy users. Performance has been boosted by up to four times with some operations, and the Documents app has been refined. **www.owncloud.org**

DISTROHOPPER

What's hot and happening in the world of Linux distros (and BSD!).

OpenSUSE Leap

First milestone arrives.

SUSE has plenty of distros on the go right now. There's OpenSUSE, the community-supported variant that many of us use at home. Then there's Tumbleweed, a rolling-release version of OpenSUSE that gets the latest packages as soon as they're built and tested – rather like Arch. And then there's SUSE Linux Enterprise, a long-term support distro designed for businesses and a competitor to Red Hat Enterprise Linux. And now, to top it all off, we have OpenSUSE Leap 42.1.

This is a distribution based on the SUSE Linux Enterprises (SLE) sources, so it's similar to CentOS. But with Leap, the SUSE team wants to bridge the gap between the OpenSUSE and SLE projects. Maintaining multiple distros takes a lot of time and developer effort, so Leap is an attempt to have one core distro and then alternative sets of packages on top of it.

Leap 42.1 is a development snapshot release; the goal is that future releases of OpenSUSE will be based on the SLE sources, but with newer packages where it makes sense for home desktop users. Whereas the current SLE release is using Linux kernel 3.12, Leap uses kernel 4.1, which has much improved hardware support.

The Tumbleweed rolling-release distro will continue as well, and the SUSE team sees the differences between the different distros like so: "Leap will target people who are looking for a stable base for workstations and servers. and Tumbleweed addresses people who are looking for the latest and greatest Free and Open Source software and sometimes can live with a little instability." Hop over to **https://en.opensuse.org/openSUSE:Leap** for more information.

OpenSUSE is going through some major changes right now, but the end result should mean more consistency between the different SUSE distros.

ArchAssault

An Arch Linux variant with maximum security in mind.

Over on our Linux Voice forums (**http://forums.linuxvoice.com**), reader ddickinson recommended that we take a look at ArchAssault, an "Arch Linux layer for penetration testers". What does "layer" mean in this sense? Well, it's a repository and set of packages that can be added to an existing Arch Linux installation – but you can also install ArchAssault from scratch. And also like Arch, it's geared towards advanced users.

Penetration testing is the process of using tools to break into another computer, but not maliciously (otherwise it's plain old cracking). For instance, if you're about to flip the switch on some internet-facing web, email and database servers, it's a good idea to run some penetration testing tools against them beforehand.

ArchAssault includes a range of software in this field, such as the Metasploit Framework, which tries to attack a machine using hundreds of different security holes. There's also Faraday, a similar tool, along with the Snort intrusion detection system which tries to identify break ins and attempts to replace critical system files. The ArchAssault team tries to stick as closely as possible to normal Arch packages, but will occasionally offer alternatives with additional security-related features enabled.

Of all the (many) Arch spinoffs we've seen, this is one of the most impressive, with an active team, good documentation and regularly updated packages.

Love Arch? Try securing your installations (and then trying to break in to them)!

News from the *BSD camps

What's going on in the world of FreeBSD, NetBSD and OpenBSD.

In the Linux world, various mechanisms exist for containing programs and restricting their access to a limited set of resources. The idea being, if there's a security hole in a piece of software and crackers exploit it, they won't be able to get very far afterwards. *AppArmor* and *SELinux* are the two most notable examples of these mechanisms.

Theo de Raadt, lead developer of OpenBSD, has started work on a similar system called *Tame*. Designed to restrict programs into a "reduced feature operating model", *Tame* isn't a bolt-on tool, but rather a system call that developers can place directly into their programs. Early on in your code, before you do anything of importance, you call **tame()** with a series of flags describing what operating system features you need. For instance, you can say that your program needs to call **malloc()**, or have write access to the filesystem, or access network sockets.

Anything else is blocked by default. A program can place multiple calls to **tame()**, with further restrictions each time, but for these restrictions can never be dropped. If a program that uses *Tame* tries to perform a restricted operation or access something it shouldn't, the kernel will terminate it with the **SIGKILL** signal.

Theo de Raadt has demonstrated the capabilities of *Tame* by introducing it into various OpenBSD tools such as *cat* and *ping*. "In the simplest non-network programs, network access is disabled. In simple network programs, file access goes away. That is the trend." See **http://tinyurl.com/pdn9rma** for all the details of how this is making OpenBSD even more secure.

Tame on OpenBSD allows programs to restrict their abilities and minimise damage if they're exploited.

Alternative OS news

Haiku is a project we've been keeping an eye on over the last decade. Whereas Linux and the BSDs are designed as all-round OSes for use on servers, desktops and mobile devices, Haiku has a much narrower focus. It originated as an open source clone of BeOS, a multimedia-oriented operating system that achieved some minor success in the late 90s, but was ultimately doomed when few PC vendors would ship it (preferring to stay on good terms with Microsoft instead).

While Haiku has some Unix-like elements, its very much its own OS with its own libraries, drivers, toolkit and desktop design. Currently it's still in the alpha stages of development – the latest release being Alpha 4 – but it's certainly usable enough for testing, with various apps including a *WebKit*-based web browser. Its system requirements are low; you can get it running on an old Pentium III box with 256MB of RAM, so it's a good way to revive old hardware.

Alex Dörfler, one of Haiku's most prolific contributors, has announced a new launch_daemon to replace the old boot process. Dörfler describes it as "a similar solution to Apple's *launchd* and Linux's *Systemd*", but whereas *Systemd* is growing into a fully fledged base system for Linux, launch_daemon focuses on the boot process.

With launch_daemon, programs can be started automatically during the boot sequence, and it's easier for users to disable certain services. In addition, it allows programs to start based on events – for instance, when a network interface comes up. See the Haiku project's website at **www.haiku-os.org** to try the latest releases.

Originally inspired by BeOS, Haiku aims to be a fast and lightweight operating system for desktops.

GAMING ON LINUX

The tastiest brain candy to relax those tired neurons

MORE ENGINES!

Michel Loubet-Jambert is our Games Editor. He hasn't had a decent night's sleep since Steam came out on Linux.

The *Unity* game engine is finally getting a Linux editor, meaning that Linux users will be able to create games natively on what is quite possibly the most popular game engine at the moment. The engine has had Linux support for a long time now and was one for the earlier mainstream engines to do so, bringing us games like *Wasteland 2*, *Guns of Icarus*, *Gone Home* and *Kerbal Space Program*.

A native Linux editor has been demanded by developers for a long time now, so the news that it has been worked on behind the scenes since 2011 has been met with a very positive response. In a blog post, one of the lead developers stated that a Linux editor has been one of the most requested features, so it seems that many developers want to ditch proprietary operating systems in favour of Linux for their game development.

More games!

The editor, which was originally released for OS X in 2009 and later Windows, will first be released as an experimental build and then become more stable if it proves popular.

Chances are that if you've played games in recent years, you will have played a *Unity*-powered one, and the prospect of more games being developed atop Linux rather than other OSs is an exciting one to say the least, especially with other big engines like *Unreal* releasing editors for Linux recently.

ARK: Survival Evolved

A great survival game that provides tonnes of fun, even in early access.

Just one glimpse at *ARK* is enough to tell that this game will be one of the biggest titles of this year. With its intriguing world, stunning visuals and tonnes of dinosaurs, it seemed to be one of those games where the screenshots and trailers set the bar too high. However, it is clear that, even in the early stages of this game, *ARK* has not disappointed.

Other than the game's curious setting and very welcome lack of zombies, it doesn't necessarily reinvent the survival genre. It follows the usual narrative of being put in the shoes of a half-naked survivor stranded on an island with no idea about how you came to arrive there. What *ARK* does, however, even in its current Alpha state, is refine the genre and create something akin to what many gamers would envision as the "perfect" survival game. Even as features are being added, the game feels huge, with a massive varied map, lots of dinosaurs to tame and turn into beasts of burden, tonnes of building options and plenty of challenge. Similarly, with local and online options, the player can choose between more laid back or aggressive games.

With higher-end hardware, the game is one of the best-looking of this year.

Being in early access isn't without its drawbacks, and in the first few weeks of release, *ARK* had plenty of glitches. However, the game is being patched-up and developed at an incredible pace and the issues are disappearing fast. The biggest omission at the moment is the game's story, though the curious objects scattered around the island create a sense of intrigue which hopefully the story will make good use of.

Website http://store.steampowered.com/app/346110 **Price** £22.99

***ARK* is the first major Linux title to use the impressive new *Unreal Engine*.**

"The game feels huge, with a massive varied map, lots of dinosaurs to tame… and plenty of options."

The Magic Circle

Explore and complete an unfinished game in this dark comedy.

An unfinished game usually isn't a selling point, unless referring to one of those "so bad it's good" simulator games. However, *The Magic Circle* is very different, providing a hilarious story where the player "play tests" a game stuck in development hell, attempting to fix it and using workarounds to get through the unplayable mess.

The main comedy is provided through the game's egomaniacal lead developer and his disgruntled employees, but also through the fact that they have not yet programmed the hero to have any abilities, needing instead to mess with the game's code in order to progress.

There are great vocal performances, which help to confirm that this is in fact a well thought-out game and not just an excuse to publish something broken under the guise of irony. That said, it is difficult to call the game fun at times, and should be viewed instead as a good piece of video game satire, focussing on dialogue over gameplay.

Website http://store.steampowered.com/app/323380 **Price** £14.99

A variety of visual styles convey a disjointed mess of a game.

Anna's Quest

A point-and-click gem inspired by the stories of the Brothers Grimm.

Full of deep, dark woods and witches who eat children, *Anna's Quest* draws on a storybook narrative appealing to younger audiences, while also giving players familiar with the adventure genre something to smile about. The genre's audience is certainly an ageing one, with many perhaps having children the same age as when they first discovered adventure games, and this is probably the most sincere attempt made at welcoming those potential players.

The game doesn't shy away from the dark themes prevalent in German folk tales, but its good dose of humour and the heartwarming story of a little girl trying to save her sick grandpa make the themes feel far from disturbing. The humour itself takes a turn away from the dry humour and witty protagonists favoured in adventure games, instead delivering more innocent chuckles while not doing away with entertaining characters.

The beautiful 2D graphics are reminiscent of the adventure games of the late 1990s.

Puzzles are perhaps easier than most, but manage to cater to established players while not alienating potential newcomers, and the excellent artwork and animation will please everyone.

Website http://store.steampowered.com/app/327220 **Price** £14.99

ALSO RELEASED...

Portal Stories: Mel

This is undoubtedly one of the best community-created mods out there and on par with the original *Portal 2*. As per the title, the game tells the story of Mel, an employee of Aperture Labs. The puzzles are considerably harder, while the story adds a lot to the *Portal* universe and is thus highly recommended to fans of the series.
http://store.steampowered.com/app/317400

Coffin Dodgers

This fun racer ditches the go-karts and the cute animal drivers in favour of mobility scooters and pensioners trying to avoid the grim reaper. Aside from its blunt humour, the game is a good deal of fun and possibly the closest thing to *Mario Kart* available on Linux, with its split-screen local multiplayer and controller support. The ability to tune-up and customise the mobility scooters is a nice added bonus.
http://store.steampowered.com/app/320540

Knights of the Old Republic II

This classic critically acclaimed *Star Wars* game got ported to Linux month, complete with a few features like controller support, achievements and support for high resolutions. The game takes place some 4,000 years before *The Phantom Menace*, in a time where the Jedi were being destroyed by the Sith. If you like *Star Wars*, RPGs or story-rich games, this one is worth picking up.
http://store.steampowered.com/app/208580

LINUX VOICE YOUR LETTERS

Got something to say? An idea for a new magazine feature? Or a great discovery? Email us: letters@linuxvoice.com

LINUX VOICE STAR LETTER

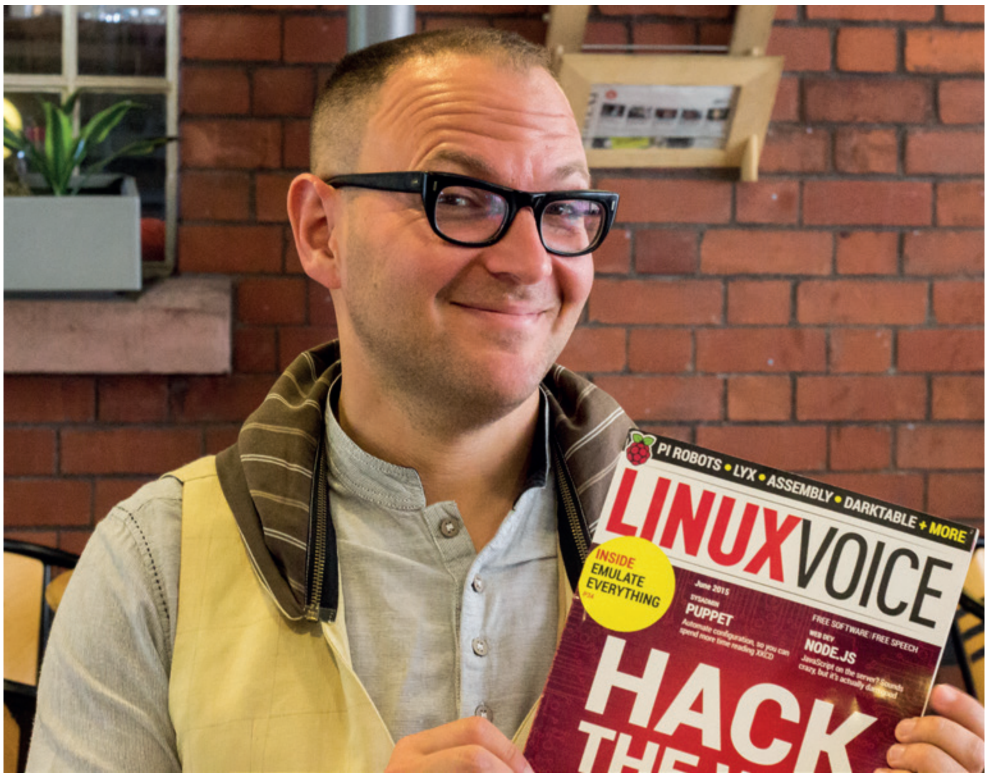
JOIN US NOW AND SHARE THE SOFTWARE

I just wanted to thank you for excellent reviews that I find in your magazine. I've just read the interview with Cory Doctorow and I found it really inspiring. Many previous ones, as well as opinions by Simon Phipps, were also great.

Reading your magazine from the very beginning I find myself more and more enjoying not the technical tutorials but the articles that help me to even more believe in FLOSS, that show me broad picture of this software in today's world and that encourages me and motivates me to act in favour of digital freedoms.

Dawid Węckowski

Andrew says: Wow, thanks David! If we're making someone out there believe in Free Software a little bit more, we're doing something right. Long may it continue!


PI ROBOTS • LYX • ASSEMBLY • DARKTABLE + MORE
LINUXVOICE
INSIDE EMULATE EVERYTHING
PUPPET
NODE.JS
HACK
Author, campaigner and speaker Mr Cory Doctorow makes a lot of sense.

FREE AS IN SHOOTING, JUMPING ETC

Dear LV, I am so grateful for your information on privacy. I now use *TextSecure* and *RedPhone*, and am also setting up using *Tor*. Loving the whole mag but there is always one section I think could push software more in line with the rest of the mag's values more, and that's gaming. Most games reviewed are Steam games, a service which seems just as bad for DRM and privacy. I know free software gaming isn't great, but what about pushing DRM-free games like those on the Humble Store or gog.com?

Stephen

Mike says: The fact that Valve has chosen Linux rather than Windows for its platform of choice is a massive news. It's worth celebrating – but then so are FOSS games, which always get a mention in our FOSSPicks pages.

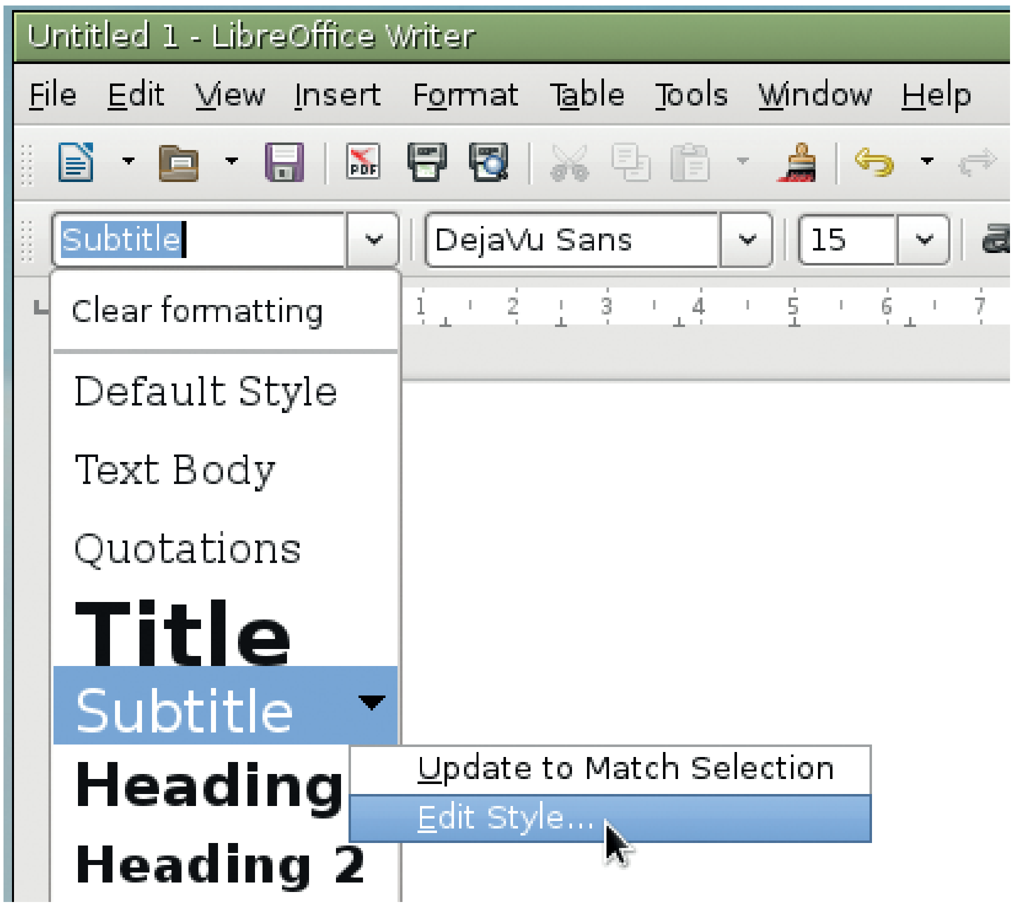
Gondor feels empty without Sean Bean. For Frodo!

LIBREOFFICE: YAY OR NAY?

This is in response to Sarah McKie asking for a beginner's tutorial on *LibreOffice* [in LV016]. Sarah with all due respect there are hundreds of functions in *LibreOffice*, and within these functions there are over a thousand procedures and it was proven that the average user uses but nine of these functions so my question is what part of the functions and procedures will you use? Which part should be a beginner's tutorial? There are many books on the market for this as well as the ever-so-informative websites, so NO this is not needed nor would it be possible for LV to please everyone it would be like trying to please Linux users on which desktop to use. If you aren't happy, learn to change it: I did, my desktop does not exist in any distro that is on the market because I took the time to learn how to change it to suit me. Isn't Linux beautiful?

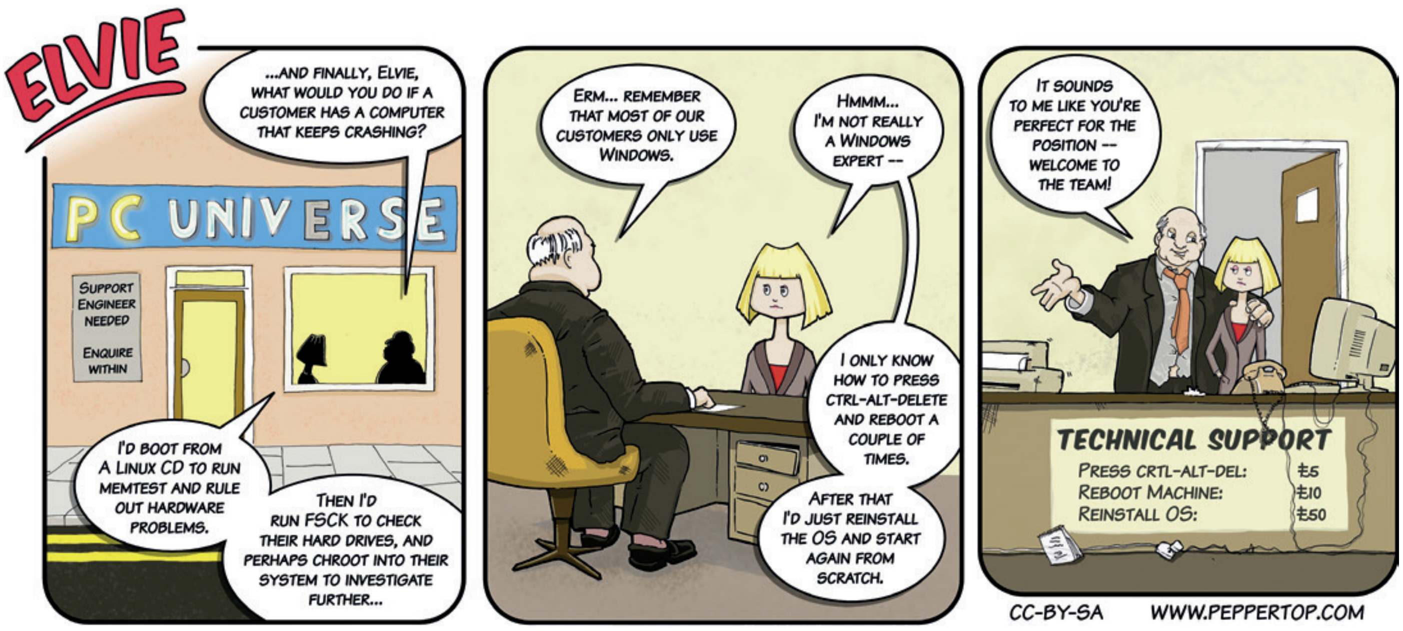
Eugene

If you create a lot of documents with *LibreOffice*, paragraph styles can save loads of time – but if you don't they're useless. That's a bit of a quandary.



Andrew says: I'm not sure it's a valid argument to say that something is complicated, so we shouldn't go near it. I also don't think it's all that helpful to say "go and look on some websites", because there's so much information out there that it can be overfacing – as Mitch Kapor once said, getting information out of the internet is like trying to drink from a fire hydrant. Part of our job is to filter the mass of information into something more digestible.

Where I do agree with you is that it would be pretty hard to choose which bits of *LibreOffice* to put into a tutorial series. I never use the presentation tool, for example, but I know that if I were just 5% more proficient with spreadsheets, I'd save hours every month. All we need to do is find someone to write a tutorial… then I could spend less time juggling columns of numbers about and more time on the fun stuff.


ELVIE
...AND FINALLY, ELVIE, WHAT WOULD YOU DO IF A CUSTOMER HAS A COMPUTER THAT KEEPS CRASHING?
PC UNIVERSE
SUPPORT ENGINEER NEEDED ENQUIRE WITHIN
I'D BOOT FROM A LINUX CD TO RUN MEMTEST AND RULE OUT HARDWARE PROBLEMS.
THEN I'D RUN FSCK TO CHECK THEIR HARD DRIVES, AND PERHAPS CHROOT INTO THEIR SYSTEM TO INVESTIGATE FURTHER...
ERM... REMEMBER THAT MOST OF OUR CUSTOMERS ONLY USE WINDOWS.
HMMM... I'M NOT REALLY A WINDOWS EXPERT --
I ONLY KNOW HOW TO PRESS CTRL-ALT-DELETE AND REBOOT A COUPLE OF TIMES.
AFTER THAT I'D JUST REINSTALL THE OS AND START AGAIN FROM SCRATCH.
IT SOUNDS TO ME LIKE YOU'RE PERFECT FOR THE POSITION -- WELCOME TO THE TEAM!
TECHNICAL SUPPORT
PRESS CRTL-ALT-DEL: £5
REBOOT MACHINE: £10
REINSTALL OS: £50
CC-BY-SA WWW.PEPPERTOP.COM

TUX: SHOULD HE STAY OR SHOULD HE GO?

FATTY BUM BUM

Tux's time is clearly not up. Linus seems to be doing everything in the 'overweight and sitting on his bum' department to emulate him.
John R Hudson

Andrew says: As I live in a glass house, it would be remiss of me to throw stones in the 'overweight and sitting on his bum front'. But at least I've made an effort to update my look since 1996. As has Linux; the moment we got Compiz and the spinning cube of wonder on the desktop, we stopped being second-class citizens in in desktop glitz, and now Linux can give you the best-looking system out there. That's why Tux needs to go!

IN THE BRINY

Further to David's letter in Sept 15 issue, I too would like Tux to be retired in his present appearance. While nowadays there's probably a greater chance of seeing, say, Ubuntu or Red Hat's logo on something commercial rather than generic Linux, there is a need for an all-encompassing umbrella logo. The current Tux, where I've seen it, looks slightly embarrassing, plus it's also hard to reproduce in monotone, since an outline of the logo doesn't stand out as a penguin, but more of a 'splat' of ink! I believe a streamlined son (or daughter) of Tux logo is required to match our lean and speedy operating system of choice, just like a penguin in water.
Chris Sandles

Graham says: Yes! Pengins are graceful, streamlined creatures, but not on land, where their stumpy legs make them wobble about like Mike Saunders after two pints of shandy. Now that Linux is more than "just a hobby OS" (as Linux Torvalds set out to produce way back on 1991), it makes perfect sense to transfer our feathered friend to his natural environment.

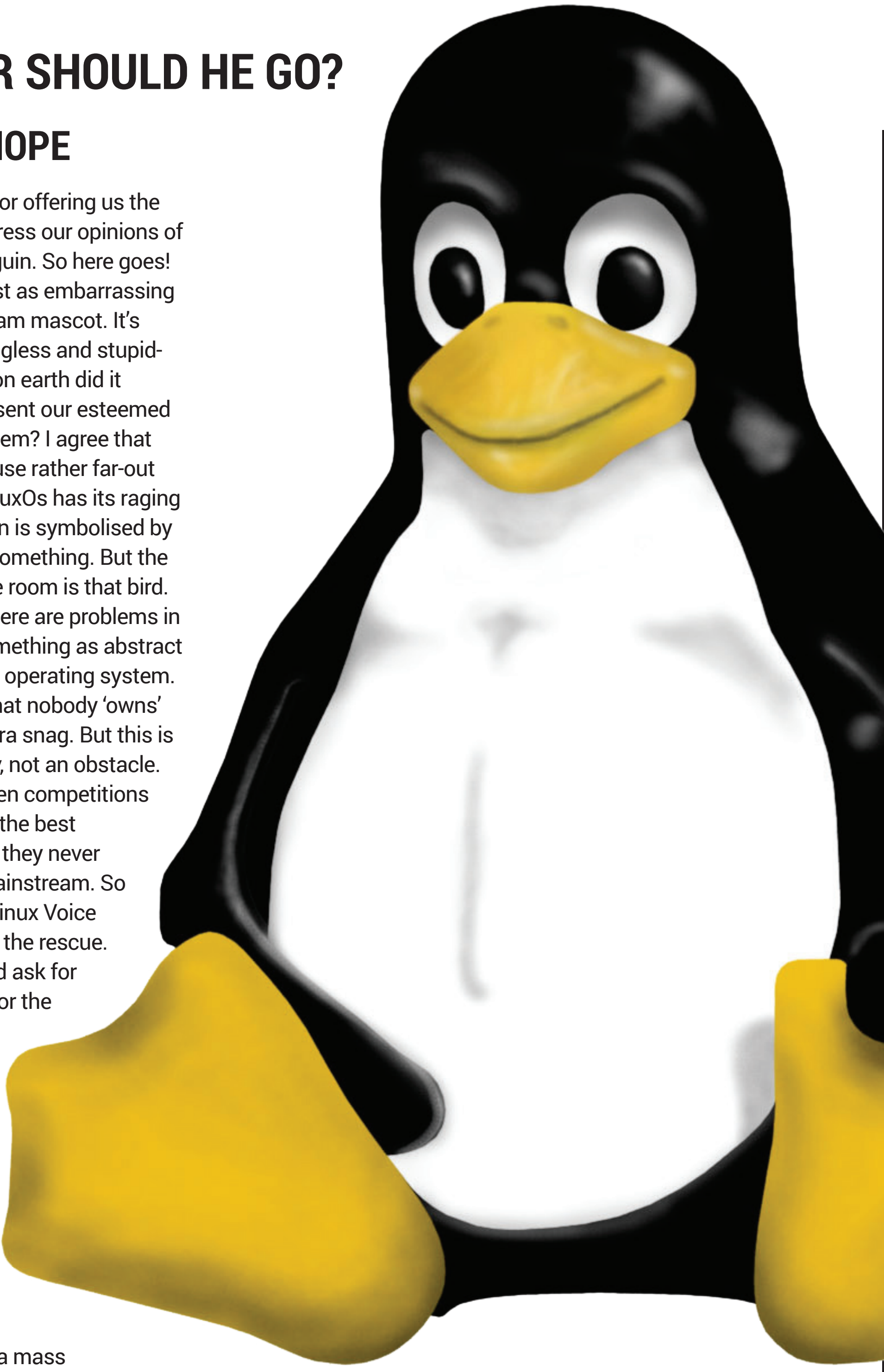
A NEW HOPE

Many thanks for offering us the chance to express our opinions of the Linux penguin. So here goes! This icon is just as embarrassing as a soccer team mascot. It's obese, meaningless and stupid-looking. How on earth did it come to represent our esteemed computer system? I agree that some distros use rather far-out images; PCLinuxOs has its raging bull and Debian is symbolised by a big swirl of something. But the elephant in the room is that bird.

Of course there are problems in visualising something as abstract as a computer operating system. And the fact that nobody 'owns' Linux is an extra snag. But this is an opportunity, not an obstacle. There have been competitions in the past for the best Linux logo but they never entered the mainstream. So this is where Linux Voice could come to the rescue. First, you could ask for submissions for the new Linux logo and your LV team could choose the winner. Second, we need backing to promote a new image for Linux with a mass publicity campaign. LV could head this with a grant from your admirable donations from your annual profits.

Microsoft has its 'four flags', Apple has an apple (?), Intel has its 'Inside' sticker and so on. So, please let's get rid of that member of the Spheniscidae family.
Maurice George

Andrew says: If only Linux had some sort of foundation to promote publicise and maintain brand awareness; maybe it could update the logo to something more fitting? But on a serious note, not all football mascots are useless – look no further than H'Angus the monkey, mascot of Hartlepool FC. He ran for mayor on a promise of free bananas for local school children, and won. He was re-elected twice, but the people of Hartlepool abolished the position of mayor after they discovered that he wasn't a real money, but a man called Stuart. Mascots matter.


Fond of sitting about, growing ever more corpulent, Andrew thinks it's time to change the Tux logo.

YOUR AD HERE

Email andrew@linuxvoice.com to advertise here

LUGS ON TOUR

PGDay UK – user group meeting

Josette Garcia reports on what's new in Python or how to enjoy a bunch of friends.

PGDay UK took place at 30 Euston Square on 7 July. The location is perfect as it's easy to get to via underground, buses or trains.

After the usual registration, refreshments, welcome and introductions came the long-awaited talk "Working in Open Source" by Liam Maxwell, CTO for Her Majesty's Government. This talk was followed by Magnus Hagander, president of PostgreSQL Europe, who alerted us to the new key features of the next release which is now available in alpha and soon to be available in beta before the final version is released towards the end of the year. Some new key features include:

- "Upsert" (INSERT ... ON CONFLICT): 9.5 brings support for "UPSERT" operations.
- BRIN (Block Range Indexes): BRIN stands for Block Range INdexes, and store metadata on a range of pages. At the moment this means the minimum and maximum values per block.
- pg_rewind: pg_rewind makes it possible to efficiently bring an old primary in sync with a new primary without having to perform a full base backup.
- Import Foreign Schema: with PostgreSQL 9.5, you can import tables en masse. You can also filter out any tables you don't wish or limit it to just a specific set of tables.
- Inheritance with foreign table: Foreign tables can now either inherit local tables, or be inherited from.

David Kennaway showed us the challenges presented in a financial services environment and how *PostgreSQL* fits into the strategy at Goldman Sachs.

Mike Lujan from Manchester University talked to us about the AXLE project... but what is the AXLE project? AXLE, advanced analytics for extremely large European databases, brings together a diverse group of researchers covering hardware, database kernel and visualisation all focused on solving the needs of extremely large data analysis. The project partners are 2ndQuadrant, Barcelona Supercomputing Centre, Portavita, the University of Ljubljana and the University of Manchester. The project, like *PostgreSQL*, is of course open source.

With "PostgreSQL Back Up and Recovery: Best practices and tools" by Devrim Gunduz, we discovered solutions for common issues along with pros and cons for each of them. There are of course many backup solutions – both open and closed source.

Gianni Ciolli gave us some "PostgreSQL Administration Recipes". These recipes should enhance the user's experience of *PostgreSQL* by making it speedier and more effective.

Marco Slot discussed the internals and performance of **pg_shard** and some of its latest features. **pg_shard** is a free, open source extension for scaling out *PostgreSQL* across a cluster of commodity servers.

"The audience was very diverse, coming from universities, companies and governments."

Magnus Hagander, president of PostgreSQL Europe, unveiling some of the new features coming soon.

In his "Fun Things to Do with Logical Decoding" Mike Fowler looked at trigger-less auditing, partial replication and full statement replication. The next talk had a great title – "The Elephant and the Snake" – could it be a story by Rudyard Kipling or one of Aesop's fables? No, but Tony Locke told us how to connect from Python to *PostgreSQL*, including tips and tricks.

The day ended with a plea from Simon Riggs for users to upgrade and test the new 9.5 release to ensure the version is bug-free, helping the *PostgreSQL* community currently working on the 9.6 release and beyond.

With around 100 people attending the conference, PGDay UK 2015 was one of the most dynamic *PosgreSQL* meetings that I have ever attended. The audience was very diverse, coming from universities, big corporations, small companies and government institutions.

Thanks to the sponsors 2ndQuadrant, EDB, Brandwatch and CitusData, we were provided with good food and a drink reception at the end of the day.

Interested in *PostgreSQL*? Why not come to the PostgreSQL conference Europe 2015, which will take place in Vienna on 27–30 October? Hope to see you there!

Four categories of new features:

SQL-related features

- BRIN indexes
- UPDATE tab SET
- SKIP LOCKED for row-level locks
- generate_series(numeric)

Administration and DBA

- Row-level security
- REINDEX SCHEMA
- Event triggers – table_rewrite
- Parallel VACUUM with vacuumdb
- ALTER SYSTEM RESET

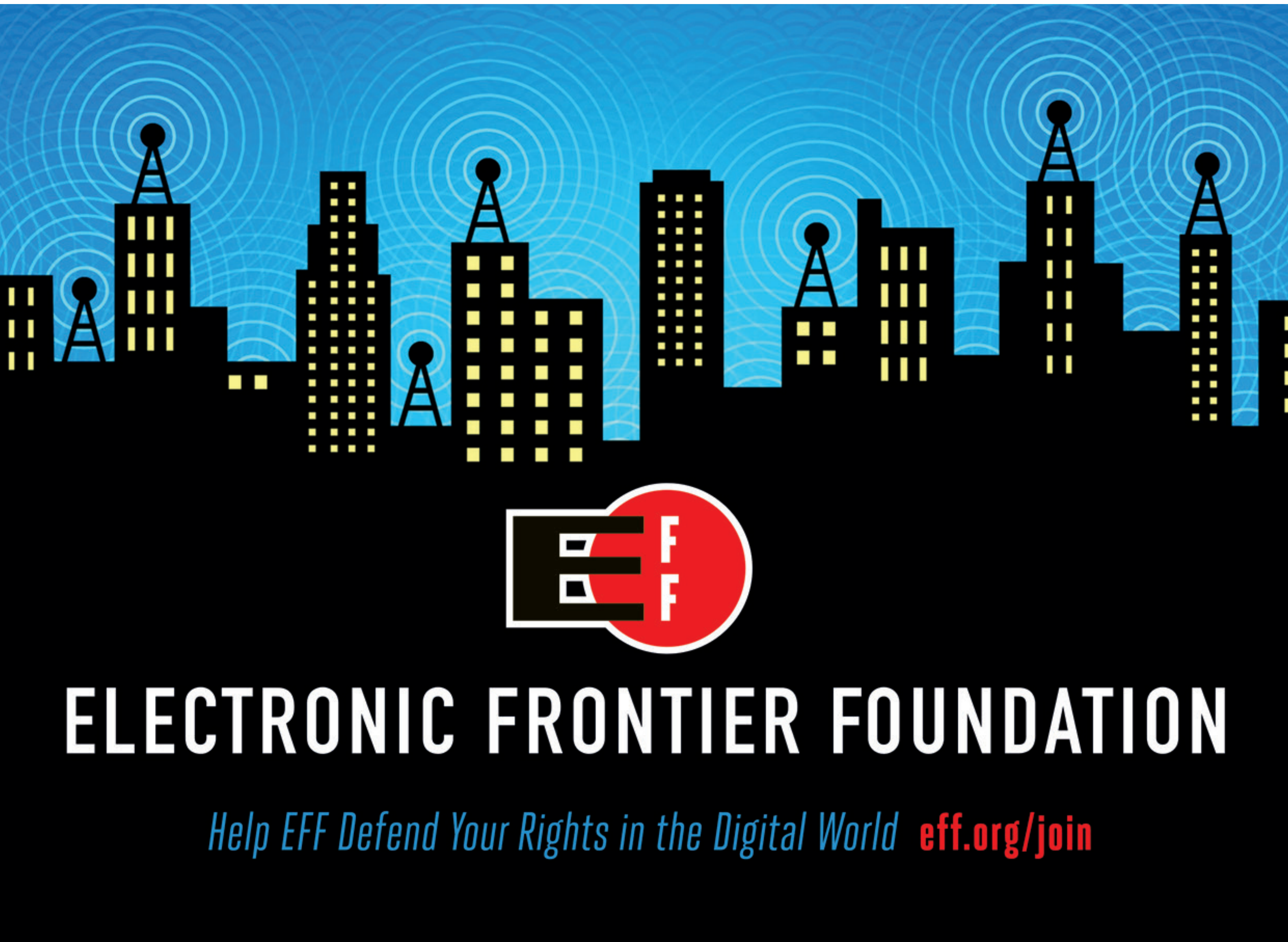
Infrastructure and replication

- Abbreviated keys
- New WAL format
- Actions at the end of recovery
- Improvements of Windows build
- Logging of replication commands
- cluster_name
- Exported snapshots with pg_dump

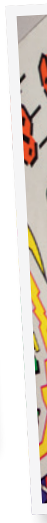
To-be-committed features?

- UPSERT
- WAL compression
- ALTER TABLE log_min_vacuum_duraQon
- File-level incremental backup
- New types for abbreviated keys
- CRC improvements
- Redesign of checkpoint_segments
- Pgaudit

Simon Riggs, founder of 2ndQuadrant.


EFF
ELECTRONIC FRONTIER FOUNDATION
Help EFF Defend Your Rights in the Digital World eff.org/join

We counted far fewer Linux desktops than in previous years, when even the Macs would be running Linux.



Allison Randal was one of the few keynote speakers whose talk looked at the future of Free Software and open source.

OSCON 2015

Graham Morrison visits Portland in the United States for ~~the Oregon Brewers Festival~~ O'Reilly's Open Source Conference

There's change in the air at OSCON. Having found its spiritual home among the micro-brewers and bridge lovers of Portland, Oregon, and after 12 years of (almost sequential) events here, OSCON is moving. First to Amsterdam at the end of October in a new attempt to bring what's been traditionally a US-centric conference to Europe. And more dramatically, in May next year, to Austin, Texas. This wind of change could be felt throughout the week, from the Kids Day held on the Sunday, through the two tutorial days of Monday and Tuesday, to the conference sessions proper, held over the Wednesday, Thursday and Friday. But most of all it was felt in the absence of Larry Wall, whose 'State of the Onion' talk was a long tradition at a conference that began as The Perl Conference. When we enquired further, we were told Larry's talk had been declined by the OSCON program committee. There was certainly a more corporate feeling at the event than usual, especially in the exposition hall, where the main areas were

The loneliness of the costume mascot is understood only by Leonard Cohen and lighthouse keepers.

Portland has more breweries than any other city on the planet.

dominated by PayPal, GitHub, Microsoft and Google. The 'Non-profit Pavilion,' including the FSF, the EFF, Software Conservancy, the Perl Foundation and OSI, was separated from the main conference area by a black boundary, which felt a little uncharitable.

Session management

The sessions themselves, however, maintained the exceptionally eclectic quality we've come to expect from OSCON, with many being videoed for later viewing. The Go, Python and even PHP languages (PHP's creator, Rasmus Lerdorf, was in attendance) were popular, as were many of the testaments given on open source strategy within big business.

Despite the corporate gloss, the majority of sessions were still suited to open source hackers of all abilities. One session delved into embedded processors; another looked at what happens after you've typed "google.com" into your browser, while Matthew Garrett, currently at CoreOS, ran an excellent session on building a trustworthy computer. On the community tracks, Simon Phipps talked about using resources at the OSI to help open source, while Bradley Kuhn, president of Software Freedom Conservancy, gave a great case study where GPL enforcement and compliance created a better community at Kallithea.

"When we enquired further, we were told Larry's talk had been declined by the OSCON program committee."

Outside of the tracks, there were two big announcements. First was the creation of the Cloud Native Computing Foundation (CNCF) by the Linux Foundation, which aims to advance the state-of-the-art for building open source cloud native applications and services. Second was the 1.0 release of Google's Kubernetes orchestration platform. We've seen plenty of announcements like these over the years, but we have to admit that the growing culture behind containers seems like a genuine revolution, from package management on phones to massively scalable data networks.

LINUX IN ~~ANARCHY~~ THE UK

Free (as in speech) software is a worldwide movement, but the UK has been a leading player in its development. Mike Saunders and Ben Everard investigate.

Linux Voice has subscribers all around the world, in the Americas, Europe, Asia, Africa and Australia. Our writers are scattered across the planet too. But the core team – Graham, Andrew, Ben and Mike – are British born and bred, and have followed the UK's involvement in the computing industry since the classic 8-bit plug-into-your-TV devices of the early '80s. The UK has been a pioneer in many aspects of home computing, and its legacy won't be forgotten.

Today, it's easy to think of the IT industry as being dominated by the USA, but there's plenty of innovation happening elsewhere. Open source software helps to make this possible, as it's free to obtain, modify and share, so you can be at the end of a dial-up modem connection in a remote Cambodian village and still contribute to your favourite project. (OK, some *Git* pull requests will take a while like that, but you get the idea...)

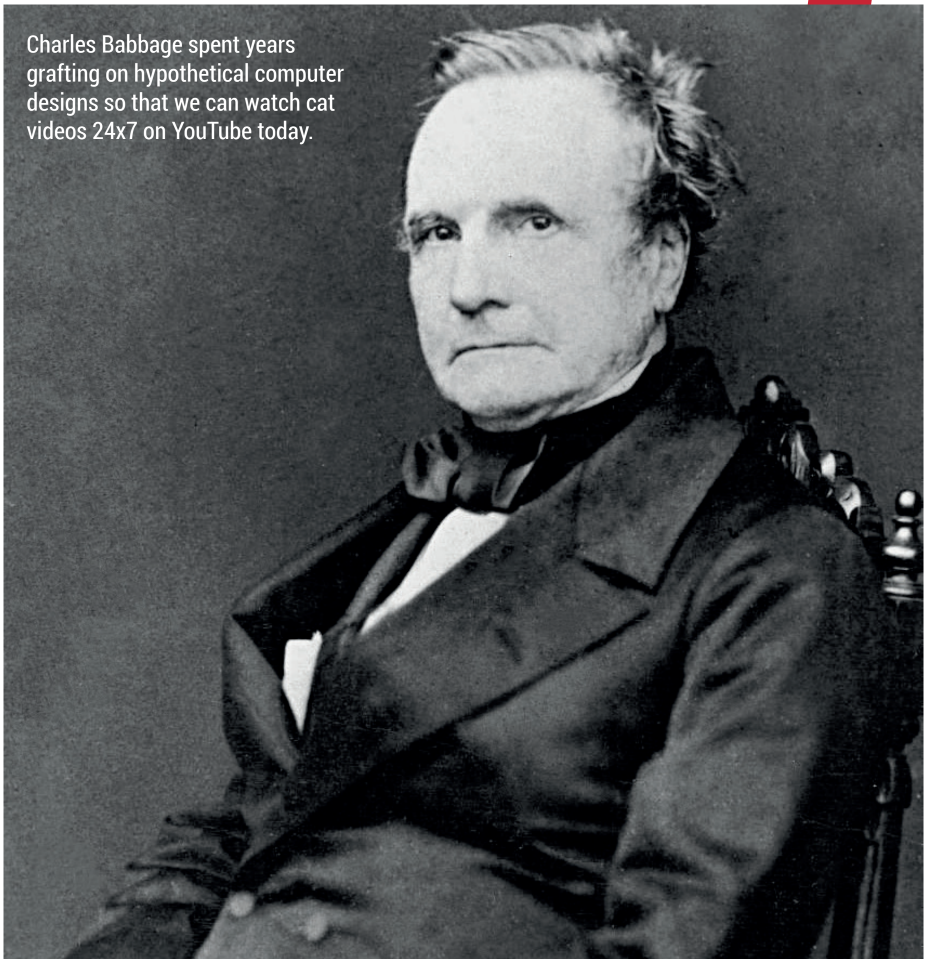
Britain is playing its part as well. Prime Minister David Cameron is trying to promote the country as a "global hub of technology excellence", and cities like Bristol are emerging as top locations for game development studios. ARM chips, which power virtually everything outside of desktop PCs and tiny embedded devices, are a British creation and are dominating the world.

And then there are British Linux companies helping to develop and promote free software. So this issue, we thought we'd celebrate everything that's jolly good from our green and pleasant isle, seeing how the UK shaped the computing industry right from its earliest day, and looking ahead to what UK-based Linux companies have planned for the future. So grab a cup of tea, turn down The Archers, and read on...

"It's easy to think of the IT industry as being dominated by the USA, but there's plenty of innovation happening elsewhere."

From tiny acorns

British scientists were inventing computers before they even existed.


Charles Babbage spent years grafting on hypothetical computer designs so that we can watch cat videos 24x7 on YouTube today.

Pinpointing the invention of the "computer" is tricky. Designs for calculating machines date back 2000 years, such as the Greek Antikythera mechanism which was used to predict astronomical events. In the 17th century, French mathematician Blaise Pascal created a mechanical adding machine, while his German contemporary Gottfried Wilhelm Leibniz produced a more complicated calculator that was plagued with problems due to the intricacy of its design.

The first machine that could be described as a general-purpose computer, however, was the Analytical Engine, described by English mathematician Charles Babbage in 1837. This went far beyond performing simple mathematical operations: it had its own memory, it could perform branches and loops, and could be fed "programs" on punch cards. At least, that was the idea. Babbage worked extensively on the design of the Analytical Engine, but it was never completed in his lifetime.

Many computing milestones were reached in the 1940s with the German Z3 and US ENIAC, but the first ever electronic stored-programmable computer, the Manchester Small-Scale Experimental Machine (nicknamed "Baby"), was brought into operation in England in 1948. This machine was designed for testing purposes, and only three programs were ever written for it, but it led to the development of the Manchester Mark 1, which in turn developed into the Ferranti Mark 1, the first general-purpose electronic computer that could be bought commercially.

In 1957, the British LEO II became the first computer to be sold and used for commercial business applications, sporting a blazingly fast 500kHz operation speed and almost 9KB of RAM to play with. The LEO II was used in various business tasks relating to payroll and inventory management, and was followed up by more powerful models with multi-tasking operating systems.

PETs at Home

While the USA produced late-70s breakthroughs in home computing with the Commodore PET and Apple II, Britain wasn't sitting still and companies started producing very cheap (in comparison) kit models for assembling at home and plugging in to a regular TV. The Sinclair ZX80 and Acorn Atom are two of the most notable of these – and the companies behind them would go on to snap up chunks of the home computer market in the 1980s with the ZX Spectrum, Acorn Electron and BBC computers.

The Spectrum ended up dominating the UK market, although in 1986 Sinclair sold it to Amstrad, another British company.

Many of Acorn's inventions – such as the ARM chip – went on to influence the future of computing, as we'll see in the next few pages. The 1990s were largely the domain of American companies such as Commodore, Atari, Dell (and countless other white-box PC providers), but today the British hardware industry is going through something of a resurgence thanks to a little credit-card shaped computer you may have heard of...

Spinning a web

Before this turns into a massive celebratory fest of Union Flags and 4pm tea breaks, however, we'd like to give some kudos to our American friends across the pond for creating the internet. Their work on ARPANET and similar technologies helped to develop the sprawling network that we have today, so we'll raise a glass of Miller Lite to them.

Arguably the killer feature of the internet, though, is the World Wide Web. This was invented in 1989 by Tim Berners-Lee, a Brit working at CERN in Switzerland. While his work had a huge impact on the way we access and share information, we should also give credit to the many technologies he built upon to create it, such as hypertext and SGML. A relatively obscure fact is that Berners-Lee wrote the first web browser on a NeXT computer – one of the machines developed by Steve Jobs and his team after he left Apple in 1985.

Hardware

Chips and devices designed in Britain are taking over the world.

Here's a mind-boggling statistic: for every human being on this planet, there are seven ARM chips. That's over 50 billion in total. Most people tend to think of Intel or AMD when it comes to CPUs, and it's true that those companies totally rule the roost in the desktop and server markets. But in the rapidly growing mobile space, where everyone is buying smartphones, tablets and low-end netbooks (and upgrading them every six months), ARM chips are dominating.

But why is this? Aren't x86 chips from Intel and AMD much better when it comes to performance? Well, yes, but that's only part of the story. ARM processors are simpler, easier to make, cheaper and have very low power requirements. They are RISC designs (Reduced Instruction Set Computing), which means that they have a relatively small set of instructions compared to x86 CPUs. (Modern x86 chips are actually more RISC internally, but still have to deal with the large range of instructions from the previous CISC designs due to backwards compatibility.)

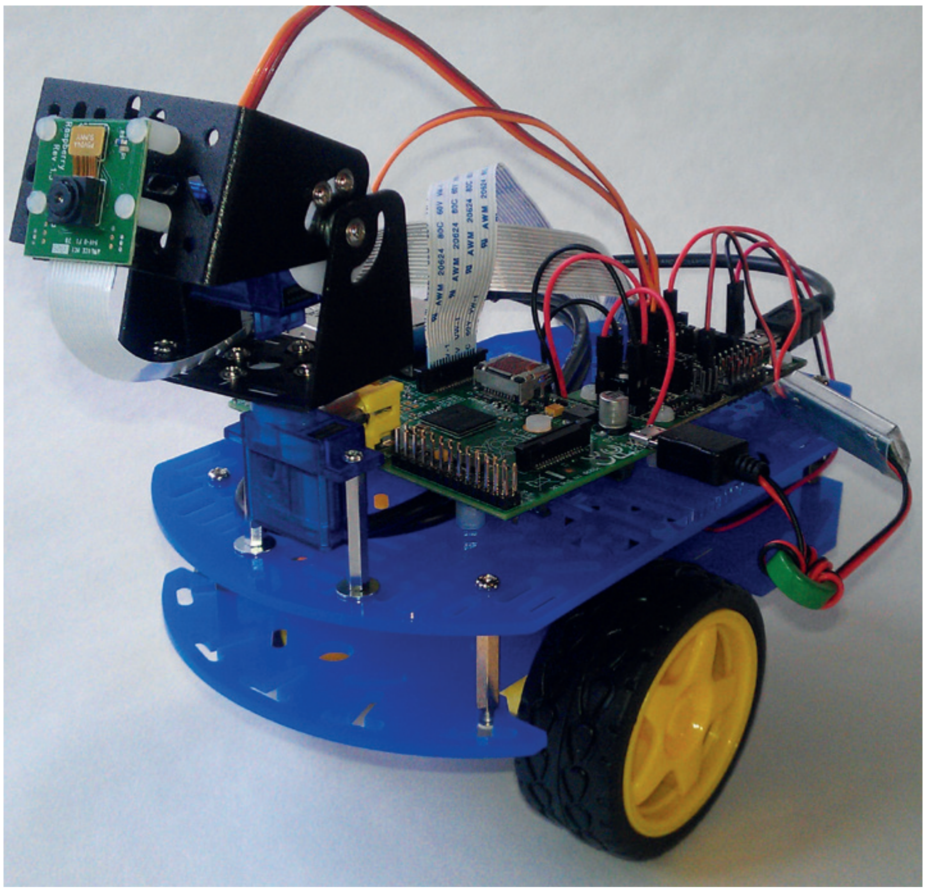
ARM's enormous success is more a product of chance rather than intention, however. Back in the 1980s, the UK company Acorn Computers started work on a next-generation chip to replace the 8-bit 6502 it had been using in the Acorn Electron and BBC range of machines. Acorn wanted to create a computer with a graphical user interface, to compete with other home computers from Apple, Commodore and Atari, and tried various in-production CPUs such as the Motorola 68000 (which was used in the Amiga, Atari ST and Sega Mega Drive/Genesis).

Floats like a butterfly...

Ultimately, Acorn wasn't happy with any of the available chips, so one of its employees, Sophie Wilson, designed a new one from scratch: the Acorn RISC Machine, aka ARM. Little did she know that billions of them would be made. ARM became the chip that powered the Acorn Archimedes range of computers, which were especially popular in UK schools, and in the early 90s other companies started to show interest in the chip. Apple, for instance, worked on newer ARM designs for use in its ill-fated Newton PDA, and Acorn spun off its ARM team into a separate company: Advanced RISC Machines Ltd.

Sophie Wilson designed the original ARM chip. Since then, 50 billion have been made.

Today, this company is part of ARM Holdings plc, a Cambridge-based firm that designs ARM chips and licenses them out to companies around the world. (It doesn't actually manufacture any itself.) However, its list of customers is impressive: IBM, Microsoft, Samsung, Broadcom, AMD, Intel and others have all bought ARM designs and manufactured chips based on them. ARM chips power over 95% of smartphones – although Intel is eager to get more established in this market as well. Sophie Wilson now works for Broadcom, which is well known for its networking and Wi-Fi chips.


The Raspberry Pi's success has bolstered new companies making add-on hardware and robots.

Pi in the sky

The Raspberry Pi needs no introduction here, but it's worth investigating its origins and the effect that it has had on British industry. Eben Upton, founder of the

Raspberry Pi Foundation, was inspired to create the device by the success of the BBC Micro in the 1980s. That computer wasn't just designed for playing a few games at home – it was part of the Computer Literacy Project from the UK's national broadcaster. In other words, it was designed to get a whole generation interested in computing and programming.

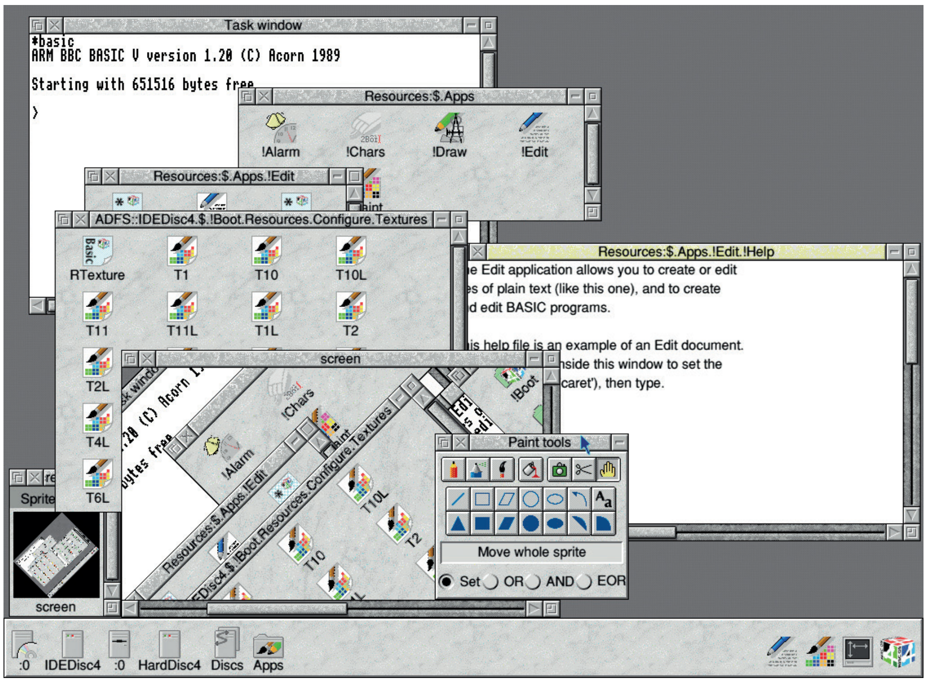
Upton and his team designed the Raspberry Pi to be a functional equivalent in the modern age. Anyone can plug it into a TV and start programming straight away with Scratch. Today, over 5 million Raspberry Pis have been solid, making it one of the most successful computers in British history, right up there with the ZX Spectrum range of machines.

Made in Britain

But the Pi has done a lot more for the UK. In a world where virtually every low-end electronic device is being produced in giant city-sized factories in China, the Raspberry Pi is something of an anomaly, as it's manufactured in Wales. In the early days of the Pi, many observers were sceptical that the machine could retain a low price while being manufactured in the British Isles, but the Pi team has achieved that.

And it goes even further: the success of the Pi has spurred on whole other industries around it. Many small companies have cropped up offering add-on hardware for the Pi, such as cases, GPIO interfaces and displays. The Pi's success has generated more widespread interest in Linux, which in turn made it easier for four crazy people to launch a new Linux magazine...

Of course, as well as being produced in the UK, the Raspberry Pi also houses an ARM chip – a British invention as mentioned before. RISC OS, Acorn's operating system that ran on its Archimedes range of computers, can now run on the Raspberry Pi and is being open sourced. So it's possible to run a computer made in the UK with a chip designed in the UK, with an operating system written in the UK on top. Not bad.



```
*basic
ARM BBC BASIC V version 1.20 (C) Acorn 1989
Starting with 651516 bytes free
>
```

e Edit application allows you to create or edit
es of plain text (like this one), and to create
d edit BASIC programs.

is help file is an example of an Edit document.
side this window to set the
caret'), then type.

With RISC OS on a Raspberry Pi, you can run a UK-made operating system on a UK-made computer with a UK-designed CPU – and it'll be super fast and energy efficient.

"It's possible to run a computer made in the UK with a chip designed in the UK, with an OS designed in the UK."

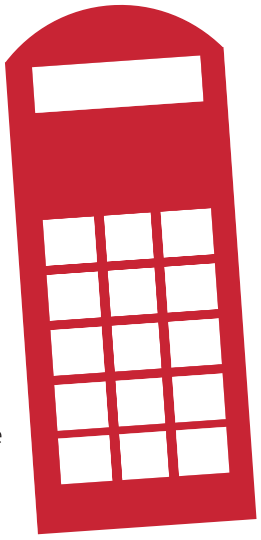
Something similar could be said about GNU/Linux. Sure, it's a worldwide collaborative effort and only a small portion of the code comes from UK-based developers, but Canonical is a British company and doing a lot of work on the OS. Canonical has developers in many countries, and a good chunk of them are based in the United Kingdom.

Linux shops

The UK also has a handful of small businesses dedicated to selling Linux-related software and hardware. The Linux CD Shop (**www.thelinuxshop.co.uk**), for instance, sells DVDs with all major distros along with some merchandise such as keyrings and case badges. The company also offers desktop PCs with Linux pre-installed. For users new to Linux, there's a Starter Pack – a bundle of the latest releases from Ubuntu, Mint, Fedora and OpenSUSE. While most of us simply download the latest distro releases when we want them, if you want to get a friend or colleague into Linux, it helps if you can give them an attractive, pre-pressed DVD.

And then there are UK-based hardware vendors selling Linux-powered machines. Entroware (**www.entroware.com**) is one of our favourites, offering a variety of desktops and laptops with Ubuntu pre-installed. The computers themselves are from ODMs (original design manufacturers) in China, but Entroware does the testing, sales and support in the UK.

We've reviewed two Entroware laptops in Linux Voice before: the Proteus in issue 11, and the Apollo in issue 15. We found the Proteus to be a well-built (albeit chunky) machine with a great keyboard, while the Apollo ultrabook was sturdy and attractive, but had fan and trackpad issues. Still, the Entroware team has been responsive and very open to feedback in our experience, so if you're in the UK and shopping around for a new Linux machine, it's worth giving them a look.



Silicon Roundabout

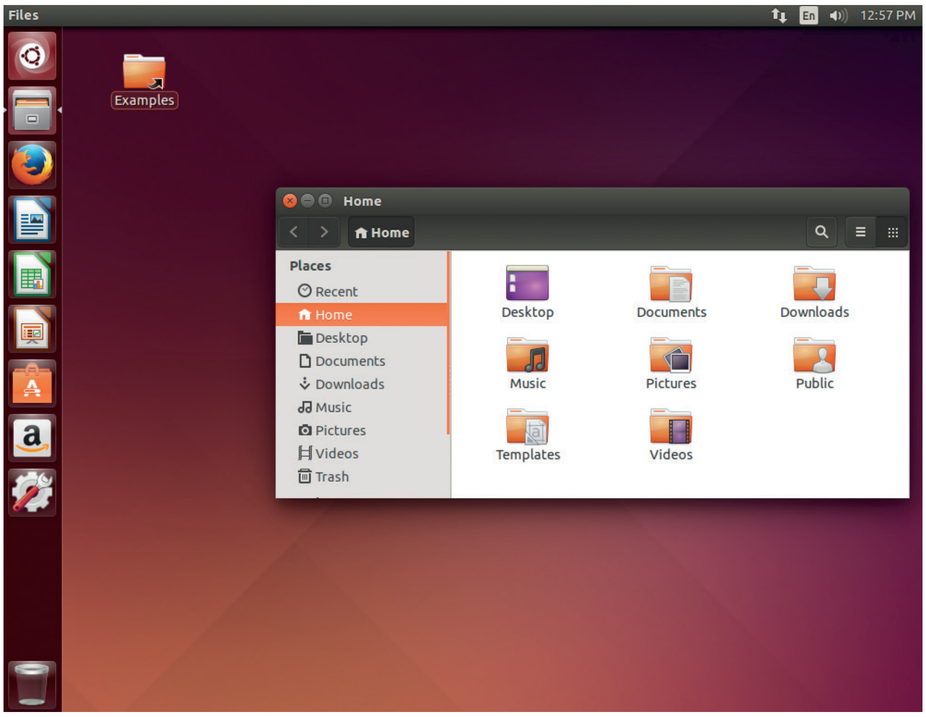
Forget Silicon Valley – the place to be now is East London Tech City, more commonly known as Silicon Roundabout. Formerly a rather run-down area of the UK's capital city, Silicon Roundabout is now home to hundreds of tech companies, from big names to tiny startups. Some of the more notable players include Amazon, Google Cisco and Facebook, while smaller startups include TransferWise (a peer-to-peer money transfer service) and PaveGen (which makes paving slabs that convert footsteps into electricity).

Software

The United Kingdom of code.

Software development is a truly global process, and almost all software is developed by (or brings together work that is developed by) people across the globe. Because of its collaborative nature, open source software is even more global than usual.

Almost all open source projects include some contributions from Britons, but then, by the same measure, almost all FOSS contains contributions from Americans, Germans, and just about every other nationality. In order to get a better picture of what's going on, let's take a look at the most prestigious open source project, the Linux Kernel. Since the shift to *Git* in 2005 made the gathering of statistics more straightforward, Britain has been the fourth top contributor to the Linux Kernel in terms of total number of patches submitted (after the USA, Germany and China in that order). However, development of the kernel is so distributed that the USA – the top contributing country – contributed just 15% of the patches.



Ubuntu's Unity desktop is hated by some and loved by others, but it has brought some innovation to the desktop environment metaphor.

Many hands make light work

If the software itself is created around the world, we can at least look for British organisations that bring it together. Linux usage statistics are notoriously unreliable. The fact that anyone can download and share any software is fantastic news for user freedom, but makes it really hard to work out how many people are using what versions of which software.

Among visitors to **LinuxVoice.com**, Britain's Ubuntu is 10 times more popular than the next most popular distro (Fedora), but not all distros identify themselves. Arch, for example, is one of many distros that just identifies itself to web servers as generic Linux. These unidentifiable Linuxes outnumber Ubuntu, so it could mean that there's another distro that's more popular, or just a lot of less popular distros. What's more, many Ubuntu derivatives will also identify themselves as vanilla ubuntu.

The numbers may be fuzzy, but it's clear that if Ubuntu isn't the most popular desktop Linux, it's certainly very close. This doesn't just go for people installing their own distros, it's also by far the most popular distro for computer makers to preinstall on machines, and you can buy machines from HP and Dell as well as specialist Linux companies.

When Ubuntu first came out in 2004, the tagline 'Linux For Human Beings' was actually quite revolutionary. Canonical and the Ubuntu developers hadn't focused on technical supremacy, but on

The government's failings Where Whitehall is holding us back

Not everything is looking up for the British tech scene. David Cameron, the Prime minister, has announced plans for a sweeping surveillance law dubbed the Snooper's Charter. In January 2015, David Cameron came out as appearing to want to ban all encryption with the statement:

"In extremis, it has been possible to read someone's letter, to listen to someone's call, to mobile communications… The question remains: are we going to allow a means of communications where it simply is not possible to do that? My answer to that question is: no, we must not"

That appears quite straightforward. He's proposing banning any form of data transfer that the intelligence services can't eavesdrop on, and that includes anything that's encrypted. Naturally, this caused some consternation in the tech world as this would mean fundamentally weakening things like online banking.

After six months of standing by this statement, Baroness Shields, the minister for Internet Safety and Security said in a letter to *Business Insider*:

"This government supports encryption, which helps keep people's personal data and intellectual property safe from theft by cyber means. It is fundamental to our everyday use of the internet. Without the development of strong encryption allowing the secure transfer of banking details there would be no online commerce."

At first this seems like the government is flip-flopping on the issue, but then Baroness Shields went on to say:

"However, the prime minister has been clear that there cannot be areas of the internet which are off limits to the rule of law – and this has to include, where necessary and proportionate, individuals' private communications"

This appears to be a complete contradiction. The government isn't seeking to ban encryption, it's just seeking to ensure there's no way for people to communicate that it can't spy on. This appears contradictory. Until the Government reveals the exact wording of the proposed law, it's impossible to know what's being banned and what isn't.

making a distribution of Linux that ordinary people could use and enjoy. This in turn forced other distros to focus on their ease of use, and the general standard of Linux distros improved significantly following Ubuntu's entry into the market.

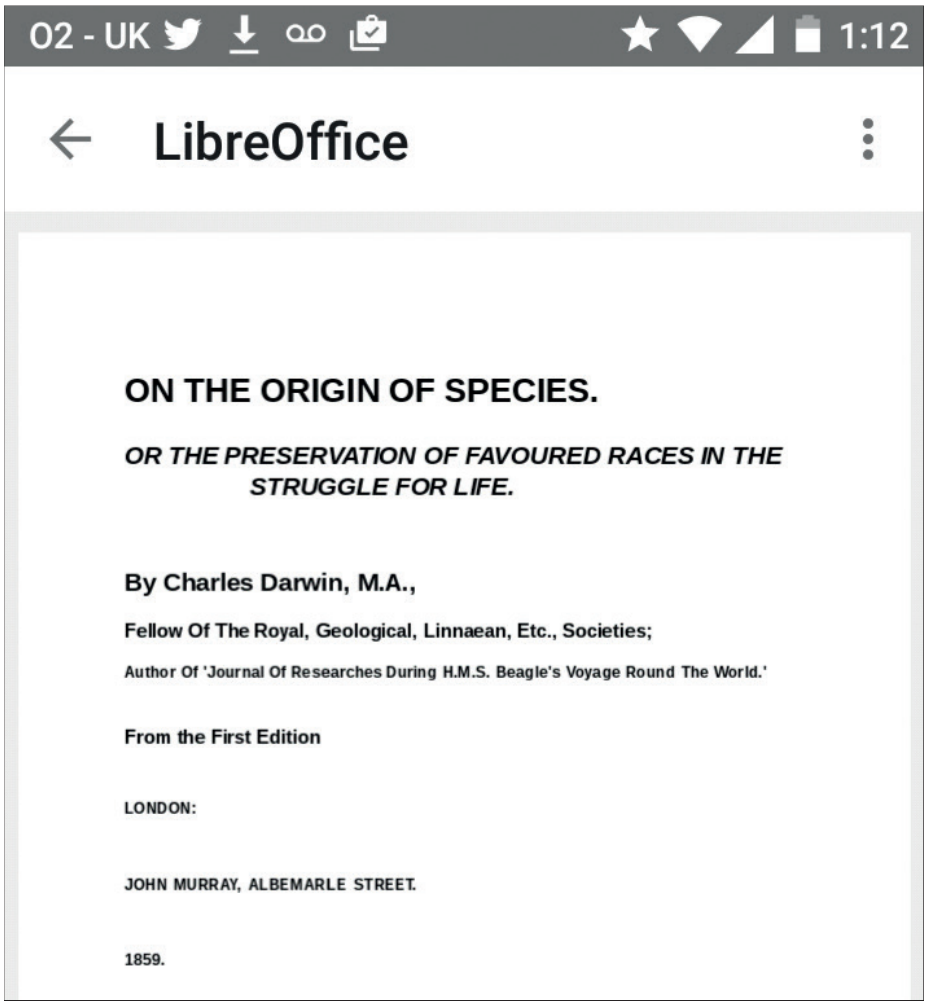
Canonical isn't just a company making a distro though: it's pushing the frontiers of Linux. This includes everything from a new paradigm for mobile phone interaction to a new approach to package management, to desktop interfaces and a controversial display server. It may not always be the most popular Linux company, but it has certainly been one of the most innovative over the last 10 or so years.

Beyond operating systems

British technology has always been at its best when it's efficient and innovative rather than brash and overly complex. The Raspberry Pi is the perfect hardware example of this., But what doesn't get noticed that often is the work that the Raspberry Pi Foundation puts into improving the performance of existing software. According to Eben Upton, it's put in: "Two man years on Squeak and Scratch. Two man years on Pixman and libav. Four Man years on the web browser." That's a significant contribution to the free software world from the small Cambridge company. It's also a significant contributor to the Wayland display server, though that is yet to become release-ready.

LibreOffice is another global software engineering effort. It started life as *StarOffice* in Sun Microsystems in the heart of the USA's Silicon Valley, but is now under the leadership of The Document Foundation, which is headquartered in Germany. Along the way, the code has been written, re-written, modified and adapted by programmers in almost every country around the world. However, despite its global nature, it's easy to see Britain's impact, particularly in newer aspects of the code. Collabora Productivity (based in Cambridge, UK) employs some of the most active developers. The company accounted for 28% of *LibreOffice* development in the first half of 2015, and it's also Collabora that's pushing the project into new areas. *LibreOffice* for Android and LibreOffice Online (a web-based version of the office suite) are both developed by engineers from this British company.

Open source software development is global in its nature, but British companies, organisations and people are at the very front pushing forwards. Whatever the future holds for the open source world, we can be sure that some of it will hail from this fair isle.



ON THE ORIGIN OF SPECIES.

OR THE PRESERVATION OF FAVOURED RACES IN THE STRUGGLE FOR LIFE.

By Charles Darwin, M.A.,

Fellow Of The Royal, Geological, Linnaean, Etc., Societies;

Author Of 'Journal Of Researches During H.M.S. Beagle's Voyage Round The World.'

From the First Edition

LONDON:

JOHN MURRAY, ALBEMARLE STREET.

1859.

LibreOffice for Android is currently only capable of viewing files, but editing is coming soon thanks to the superb work being done by Collabora.

"Cambridge's Collabora Productivity accounted for 28% of LibreOffice development in the first half of 2015."

The government's successes Where Whitehall is pushing us forward

The UK Government has a chequered relationship with the technology industry. Some proposed legislation is, justifiably, vilified by the tech world, but it's not just making enemies. In some areas, Her Majesty's Government is doing an excellent job at using and promoting open source software.

The Government's GitHub page (**https://github.com/alphagov**) lists 396 project that are all being developed in the open. Some of them are small projects only really relevant to a single purpose, but it also includes large projects such as the government's e-petitions platform (**https://github.com/alphagov/e-petitions**). Most of the government's software is provided under a very liberal BSD-like licence.

The government's push for open standards has perhaps been stronger than its push for open source software. This is ostensibly in an effort to avoid vendor lock-in, to ensure that it can always switch to more cost effective software, rather than because of any philosophical commitment to open standards.

The most famous decision in this area was the move to ensure all office documents were stored in Open Document Format (ODF) files. This is the standard format of *LibreOffice*, *OpenOffice* and several other open source office suites. However, it's not only supported by open source tools. Since 2013, *Microsoft Office* supports the standard, and the government isn't yet showing any sign of switching away from this proprietary office suite.

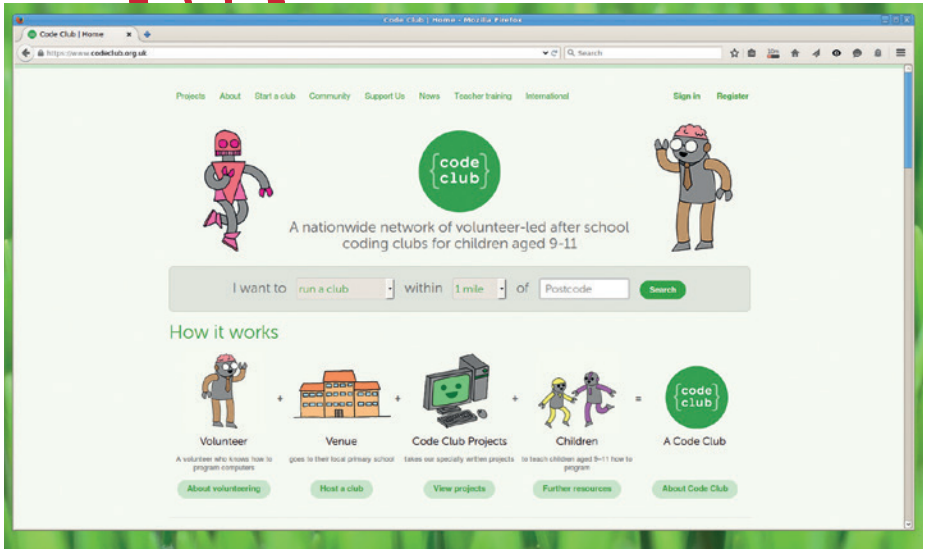
Into the future

What next for this scepter'd isle?

British technology hasn't always done well. After starting the computing revolution, things went quiet for a while. Our nation's technology rose again in the 1980s, but was eventually crushed under the weight of the Microsoft and Intel pairing in the 1990s. Now, though, we think we're rising with a new British technological renaissance with open source technology at its heart.

The standard bearer for this new movement is the Raspberry Pi and its push to improve the standard of computing literacy across the country. It's this new generation, who are just now learning about computing using Raspberry Pis and Linux in classrooms up and down the UK, that will drive the next generation of British tech.

The enthusiasm for the Raspberry Pi showed that there was a significant latent demand in the UK for homegrown technology. Soon after the Pi Foundation released the first board, a small industry sprung up making and selling add-ons. In the intervening three years, this industry has continued to grow and expand, and shows no sign of stopping. This could be the foundation of a resurgence in UK tech manufacturing that died out significantly at the end of the last century.



A nationwide network of volunteer-led after school coding clubs for children aged 9-11

A volunteer who knows how to program computers goes to their local primary school takes our specially written projects to teach children aged 9–11 how to program

As well as schools, there are a growing number of organisations to help kids learn computing skills, such as Young Rewired State and Code Club.

Planting seeds

Of course, if there's going to be a resurgence of UK Tech in either hardware or software, the best place to start is where the engineers of tomorrow are currently learning their skills: schools. This is a place where Britain has turned itself around in the past two years with the release of a new computing curriculum. The old curriculum based on using IT has been thrown out, and in 2014 a new curriculum came into force that emphasised creating and moulding technology with a strong emphasis on programming.

"The best place to start is where the engineers of tomorrow are currently learning their skills: schools."

Michael Gove (the education minister at the time) said at the launch: "ICT used to focus purely on computer literacy – teaching pupils, over and over again, how to word process, how to work a spreadsheet, how to use programs already creaking into obsolescence; about as much use as teaching children to send a telex or travel in a zeppelin."

"Now, our new curriculum teaches children computer science, information technology, and digital literacy: teaching them how to code, and how to create their own programs; not just how to work a computer; but how a computer works, and how to make it work for you."

The most significant part of the new curriculum is focussed on teaching children to program. Michael Gove went on to explain this in a little more detail:

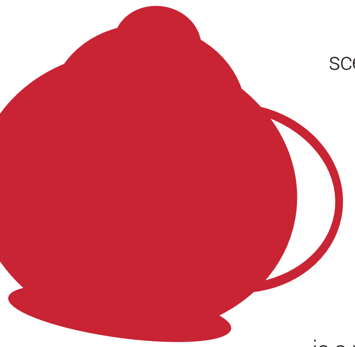
"From 5 [years old], children will learn to code and program, with algorithms, sequencing, selection and repetition; from 11, how to use at least two programming languages to solve computational problems; to design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems; and how instructions are stored and executed within a computer system. "

It's the children of today that will be the programmers of tomorrow. By making sure that every child is exposed to coding, hopefully, the new education system is ensuring everyone who has a talent for programming learns how to utilise that talent. If this goes to plan, the Britain of the future will have the skills to continue to push technology forward.

The BBC is hoping to repeat the coding success of the 1980s with its Micro Bit board.

Enter Auntie

Perhaps, though, the best sign that Britain's tech future looks bright is the return of the BBC to the hardware scene. Although this institution is best-known for its TV and radio stations, in the 80s the BBC released a series of computers that drove the tech



scene of the time. In the intervening decades, it remained quiet on the hardware front, until the release of the Micro Bit.

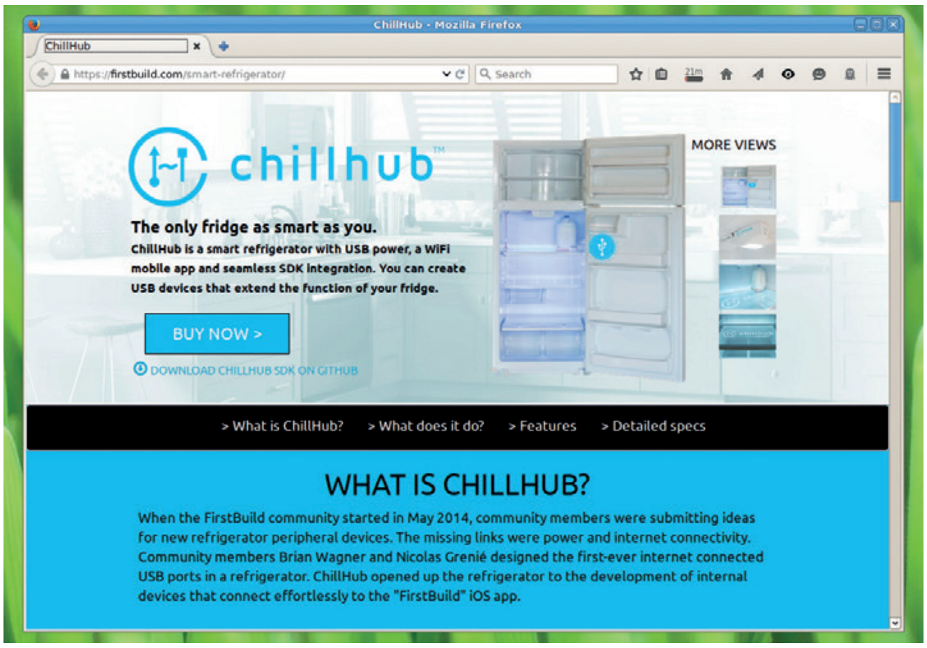
The Micro Bit is a microcontroller designed specifically for teaching children. A microcontroller is a stripped-down processing unit with a little memory and a little storage. It doesn't have enough power to run an operating system. Instead, programs are written on other computers and uploaded onto the device, which can just store and execute a single program at a time.

The Micro Bit can be programmed via a Microsoft-developed web app known as Touch Develop, which enables programs can be uploaded through a USB or Bluetooth. The web-based software should work on Linux, though as we go to press, we don't have the hardware to test this with. This environment is based on ARM's Mbed platform but it's written specifically for the Micro Bit by Microsoft. We haven't been able to confirm whether the system will work with other mbed systems.

The BBC has made some promises about opening up the hardware, but when we pushed for details, its press contact simply replied 'details will follow in due course'.

The finalised design of the board is well suited to an educational microcontroller. It includes a motion sensor and a magnetometer to detect the environment around it, two push buttons for input and an array of LEDs for output. All this will be powered by an ARM Cortex-M0 processor (the Cortex-M processors are very different from the Cortex-A series chips that commonly power mobile phones).

At 5 x 4 cm, the Micro Bit is about half the size of a credit card. Unlike an early prototype, the final version is rectangular, not bug-shaped. It has five easily accessible pins (three input/output pins and power outputs) that can be accessed with crocodile clips, banana plugs and conductive thread. There are also an additional 17 GPIOs (including I2C and SPI buses) accessible via the edge connector, but this will require a special clip to access. The board will be given free to 1 million UK schoolchildren, and will also be for sale both in the UK and internationally. This means that it will rival the Arduino to be the most popular microcontroller board from release, and with this number of boards in the wild. If the response to the Raspberry Pi is anything to go by, we can expect to see a strong level of community support, and a range of expansions soon after launch.



ChillHub - Mozilla Firefox

ChillHub

https://firstbuild.com/smart-refrigerator/

chillhub™

The only fridge as smart as you.

ChillHub is a smart refrigerator with USB power, a WiFi mobile app and seamless SDK integration. You can create USB devices that extend the function of your fridge.

BUY NOW >

DOWNLOAD CHILLHUB SDK ON GITHUB

MORE VIEWS

> What is ChillHub? > What does it do? > Features > Detailed specs

WHAT IS CHILLHUB?

When the FirstBuild community started in May 2014, community members were submitting ideas for new refrigerator peripheral devices. The missing links were power and internet connectivity. Community members Brian Wagner and Nicolas Grenié designed the first-ever internet connected USB ports in a refrigerator. ChillHub opened up the refrigerator to the development of internal devices that connect effortlessly to the "FirstBuild" iOS app.

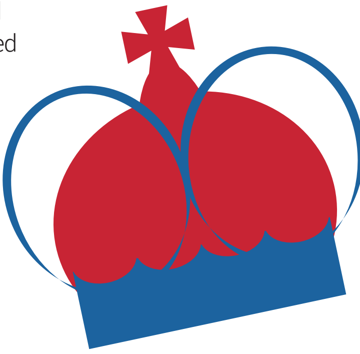
ChillHub's smart refrigerator runs Ubuntu: the future of technology or pointless frippery?

"The UK is surging forward with technological development based on open source software."

The future's bright

The UK's relationship with technology has varied over the past 200 years. At its best, the UK has lead the world. At its worst, the UK has been little more than a consumer of hardware and software that's both designed and made abroad. When it's at its best, our small island becomes a bountiful source of technology that the whole world benefits from. That benefit can come in the form of theoretical advances, nicely packaged software or low-cost boards that open up the technological world to a new generation.

At the moment, the UK is surging forward with technological development based on open source software. There's a huge wave of fantastic creativity just around the corner, and we can't wait to be part of it. LV



Internet of Things It's everywhere!

If there's one phrase that's bound to cause an argument among a group of engineers it's 'The Internet of Things'. It's the buzzphrase *du jour*, but some people actually think it will amount to more than a lot of hot air from advertisers. The basic idea of the internet of things is that as internet connections become ubiquitous, and processors become small, cheap and low-power enough, we'll start connecting everything to the internet.

We're already seeing this to some extent with IP-enabled webcams and even a fridge that runs Ubuntu (**https://firstbuild.co m/smart-refrigerator**), but proponents of the idea tell us that this is just the beginning and soon everything will link together to form the aforementioned internet of things. Other people claim that there's little to be gained from your desk lamp hosting a web page so the fad is sure to die out (anyone who's interested in making a web-based light a reality using a Linux stack should note that **LAMPlamp.com** is available for sale).

One person who believes in this technology is George Osborne, the Chancellor of the Exchequer. In the March 2015 budget he announced a £40m fund to help startups create new devices and applications for the Internet of Things.

WRITE FOR LINUX VOICE

Linux Voice wants your ideas for tutorials, guides, how-tos and insights from the hacker world. If you've found something you want to tell the world about, let us know

What material is Linux Voice interested in?
Most of the time we're more interested in what you can do with software X, rather than singing the praises of software X itself. Clever software is good but useful software is better. Proprietary software that works on Linux is acceptable, but what we're most interested in is Free Software.

What don't you want?
We sometime get submissions that go like "I've been using Linux for X years; can I write for you?". This isn't very helpful, to us, because what we want to see is that you:

- Have an idea
- Can explain it clearly

If you can point us to examples of something you've written, please do – we're not looking for Shakespeare; we value clear communication and enthusiasm above all else.

What do you want?
Tutorials. We want tutorials, of around 3,300 words in length usually. We pay money! All tutorials should have a clearly stated aim, so readers know at first glance why they should follow it. "Get started with XX software" doesn't tell you anything; "Build a weather tracker with Python" is much more active and informative.

These are common reasons why we reject ideas:

- Something which has been covered repeatedly on Linux Voice and/or elsewhere
- Material not obviously related to Free Software
- Incoherent writing

Email ben@linuxvoice.com to write for Linux Voice

ownCloud
CONTRIBUTOR
CONFERENCE

August 28 - Sept 3 2015
Berlin

Bringing ownCloud Contributors from around the world together for a week of coding, design, discussion, talks and fun

Hackathon

August 28-sept 3

Writing code & sharing inspiration:

- Coding (PHP, JS, CSS, HTML, C++)
- Design & Front-end
- Testing
- Translation & Documentation

Talks & workshops

August 29

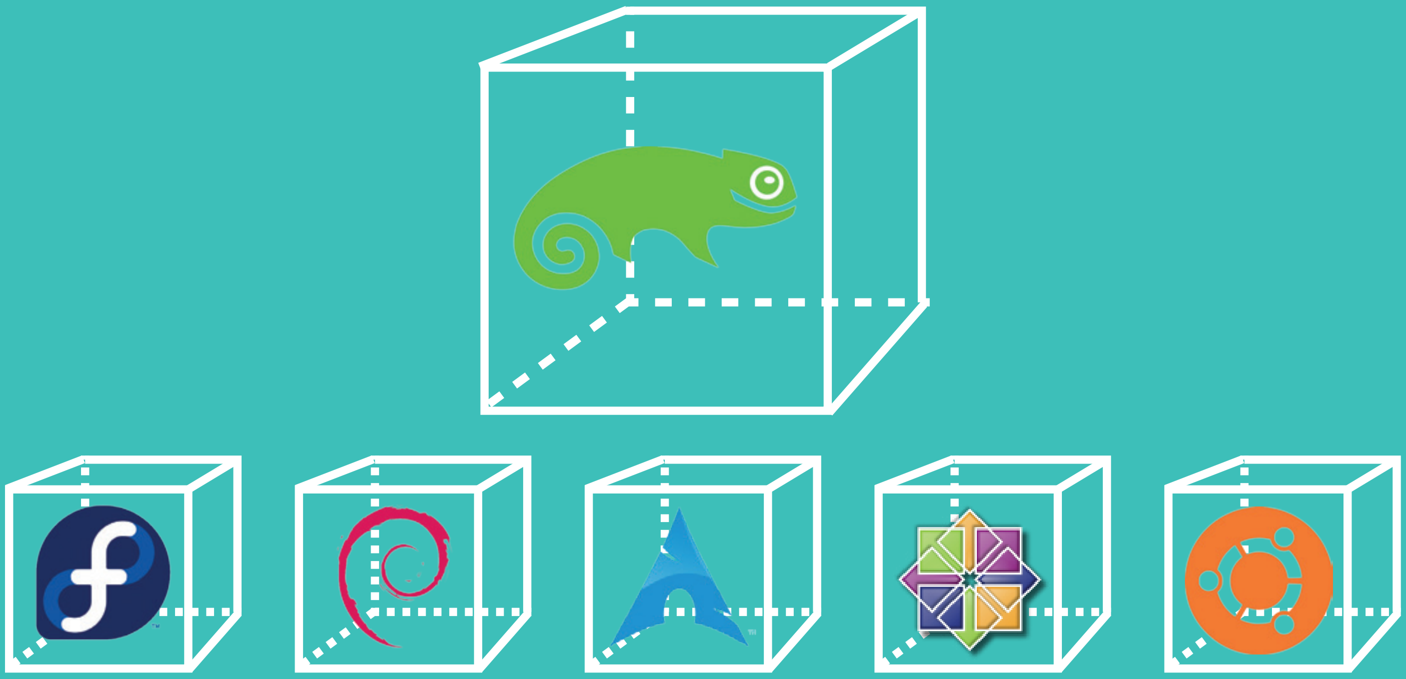
Keynotes, talks, workshops:

- Write your first ownCloud App
- Secure your PHP application
- Connect your JS app to ownCloud
- State of Music, Notes, Contacts...

Join us

And build your own Cloud!

owncloud.org/conf



SUSE'S OPEN BUILD SERVICE

Want to build packages for a wide range of distros with just a few clicks? **Henne Vogelsang** and **Markus Feilner** of SUSE explain their distro's Open Build Service.

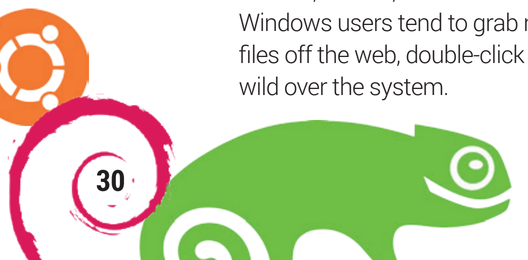
We at Linux Voice dread having to administer Windows machines. Fortunately, we don't have to do it very often, and we get pretty much all of our work done inside our comfy and familiar Linux distributions. Every now and then, however, some friend or relative will get on the phone and ask us to fix something with their Windows box. With heavy hearts, we trudge over to their places to poke around in registries, System32, and other horrors.

But the worst part is dealing with software. Compared with Linux, software installation in Windows is a nightmare, as you probably know. Instead of using a consistent, secure and well-checked repository of software, as we have in Debian, Ubuntu, Fedora, SUSE and many other distros, Windows users tend to grab random EXE and MSI files off the web, double-click them, and let them run wild over the system.

But enough about Microsoft. The point we want to make is: software installation on Linux is something we can all be proud about. But! It's by no means perfect. Due to all of the niggling differences between distros, there's no guarantee that a package made for distro A will work on distro B. Sometimes there will be no compatibility issues – but in other cases little glitches can creep in, which are often hard to spot.

"Software installation on Linux is something we can all be proud about, but it's not perfect."

Various solutions to this problem have been proposed over the years, such as Autopackage, but arguably the best is SUSE's Open Build Service. This takes a lot of the effort out of packaging software for different distros, so we thought we'd ask the SUSE developers to explain how it works. If you're a software developer this is vital reading – or if you're just a regular end user, it's worth knowing what goes on in the background. So, take it away, Team SUSE...



A bit of background

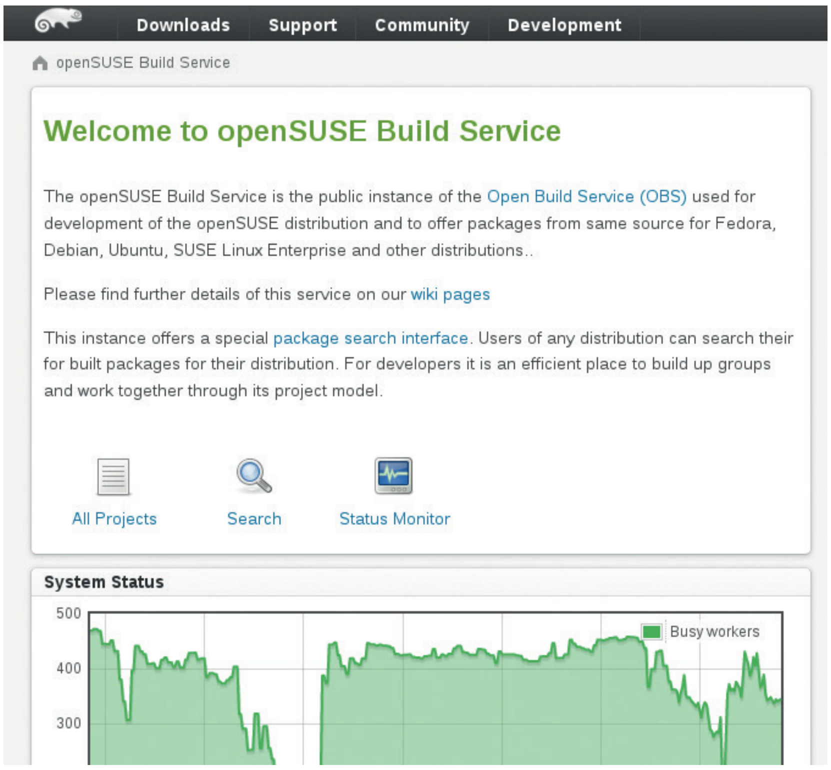
SUSE, OpenSUSE, Red Hat, Fedora, Mandriva, Ubuntu, Debian, Arch – there's a long list of distributions that are supported by SUSE's development platform Open Build Service (OBS, **http://openbuildservice.org**). The OpenSUSE Build service (**https://build.opensuse.org**), a public instance of OBS, has been around since about 2006 when SUSE's Adrian Schröter first announced it. Since then the tool has grown, sponsored by SUSE and contributors like the German admin company B1 Systems. As of July 2015, more than 500 CPU cores host some 39,000 software projects with almost 300,000 packages in 52,000 repositories, contributed to by almost 42,000 developers.

The basic challenge of distributing free software is compiling that awesome open source code into machine code for different Linux distributions in a way that makes it easily consumable for users. That may sound simple, but once you get started, you'll find out it very often isn't simple at all.

If you look at the dependencies of a typical free software project you will find thousands of other projects in the stack. They build on top of each other and have functional dependencies; sometimes they are interdependent – they conflict with each other and what not. In short: building free software is like herding cats. But it's even more than that: the users of the software will expect a steady, well behaved, streamlined herd of cats!

The Open Build Service (OBS) is the tool that makes this possible. It helps free software distributors to automatically and consistently build binary packages from free and open source code. Engineers upload source code and build instructions to the OBS via its APIs either through the web interface seen in the OpenSUSE build service or its command line client **osc** (**https://en.opensuse.org/Build_Service/CLI**). Then the system compiles that into binary packages for different distributions or versions of these and different CPU architectures.

Build instructions describe how to compile the source code into machine code, defines the dependencies and the conflicting capabilities the software has in relation to other software, itemise which files are needed to run it, and provide a whole lot of other metadata.



Welcome to openSUSE Build Service

The openSUSE Build Service is the public instance of the Open Build Service (OBS) used for development of the openSUSE distribution and to offer packages from same source for Fedora, Debian, Ubuntu, SUSE Linux Enterprise and other distributions..

Please find further details of this service on our wiki pages

This instance offers a special package search interface. Users of any distribution can search their for built packages for their distribution. For developers it is an efficient place to build up groups and work together through its project model.

The OpenSUSE build service is a public offering that shows the web interface to the Open Build Service.

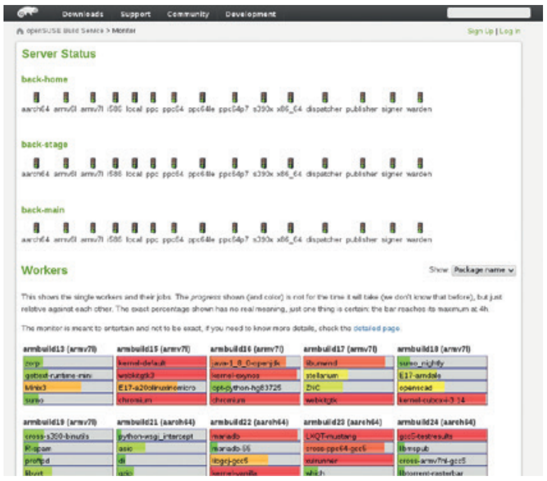
A unique problem solver

The job of the OBS is to interpret all this information, to provide an up-to-date build environment matching the requirements, the execution of the build, and of course the reporting of the outcome.

The end result is a bunch of binary packages out of one single source. The twist is this: if other software depends on the package in some way, the OBS will trigger a rebuild of the depending package afterwards. This ensures that changes propagate through the whole stack and that the user gets a consistent product to install.

Sadly, software is sometimes defective and people make mistakes. Nobody is perfect. That's why the second basic service that a free software distributor delivers to its users is the art of exchanging pieces without breaking the whole: Updates. Your distributor does not want to interrupt you in going about your business, and to be able to do this they need to reproduce the software builds at any given time in the long lifetime of the software.

The OBS helps distributors to achieve this by tracking who has made changes, when changes are



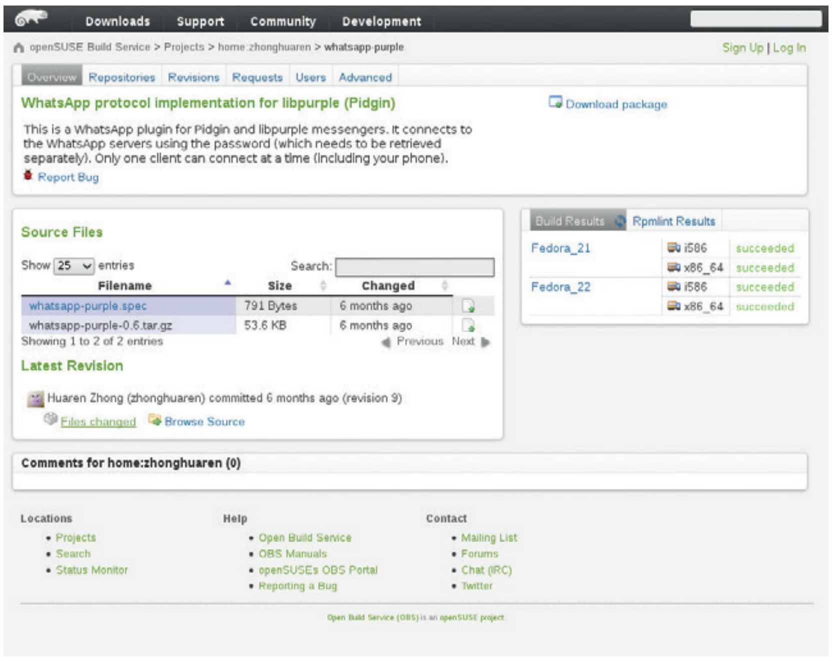
Downloads
Support
Community
Development
openSUSE Build Service > Monitor
Sign Up | Log In
Server Status
back-home
aarch64 armv6l armv7l i586 local ppc ppc64 ppc64le ppc64p7 s390x x86_64 dispatcher publisher signer warden
back-stage
aarch64 armv6l armv7l i586 local ppc ppc64 ppc64le ppc64p7 s390x x86_64 dispatcher publisher signer warden
back-main
aarch64 armv6l armv7l i586 local ppc ppc64 ppc64le ppc64p7 s390x x86_64 dispatcher publisher signer warden
Workers
Show Package name

This shows the single workers and their jobs. The *progress* shown (and color) is not for the time it will take (we don't know that before), but just relative against each other. The exact percentage shown has no real meaning, just one thing is certain: the bar reaches its maximum at 4h.

The monitor is meant to entertain and not to be exact, if you need to know more details, check the detailed page.

armbuild13 (armv7l)	armbuild15 (armv7l)	armbuild16 (armv7l)	armbuild17 (armv7l)	armbuild18 (armv7l)
zorp	kernel-default	java-1_8_0-openjdk	liburnerd	sumo_nightly
gobject-runtime-mini	webkitgtk3	kernel-exynos	stellarium	E17-amdale
Minix3	E17-a20olinuxinomicro	opt-python-hg83725	ZNC	openscad
sumo	chromium	chromium	webkitgtk	kernel-cubox-i-3.14
armbuild19 (armv7l)	armbuild21 (aarch64)	armbuild22 (aarch64)	armbuild23 (aarch64)	armbuild24 (aarch64)
cross-s390-binutils	python-wsgi_intercept	mariadb	LXQT-mustang	gcc5-testresults
R-spam	asio	mariadb-55	cross-ppc64-gcc5	libmspub
proftpd	di	libgcj-gcc5	outrunner	cross-armv7hl-gcc5
libvirt	asio	kernel-vanilla	which	libtorrent-rasterbar

Who is doing what? Where are the current problems, which platforms are we building? Find out, with OBS's extensive reporting service.



openSUSE Build Service > Projects > home:zhonghuaren > whatsapp-purple

WhatsApp protocol implementation for libpurple (Pidgin)

This is a WhatsApp plugin for Pidgin and libpurple messengers. It connects to the WhatsApp servers using the password (which needs to be retrieved separately). Only one client can connect at a time (including your phone).

Source Files

Show 25 entries Search:

Filename	Size	Changed
whatsapp-purple.spec	791 Bytes	6 months ago
whatsapp-purple-0.6.tar.gz	53.6 KB	6 months ago

Showing 1 to 2 of 2 entries

Latest Revision

Huaren Zhong (zhonghuaren) committed 6 months ago (revision 9)

Fedora_21	i586	succeeded
	x86_64	succeeded
Fedora_22	i586	succeeded
	x86_64	succeeded

Comments for home:zhonghuaren (0)

Locations
- Projects
- Search
- Status Monitor

Help
- Open Build Service
- OBS Manuals
- openSUSEs OBS Portal
- Reporting a Bug

Contact
- Mailing List
- Forums
- Chat (IRC)
- Twitter

Open Build Service (OBS) is an openSUSE project.

The search for a single package reveals the almost distribution-agnostic approach of OBS. In this case, the developer of the Whatsapp Plugin for the chat client *Pidgin* obviously favours Fedora over SUSE.

made, and what has changed. The OBS also helps by using a clean, virtualised build environment. This is how it goes: When an engineer triggers a build in the OBS by changing the software in it, the system saves the current state, gathers all the dependencies and kicks off a virtualised environment to execute the build. As the information on how to assemble the build environment is contained in the software, and as all dependent software gets rebuilt too, it makes every build reproducible.

If something is reproducible it is also predictable, and that's what distributors aim for. If you can predict how all of the software projects in your stack influence one another, you can make sure that a change to a single piece can be managed through all of its dependencies, ensuring that the whole system of software continues to work after a change. This is how the OBS helps you as a user, because updates from the OBS don't get in the way of your business.

"OBS supports more than 20 distros, half a dozen architectures and three package formats."

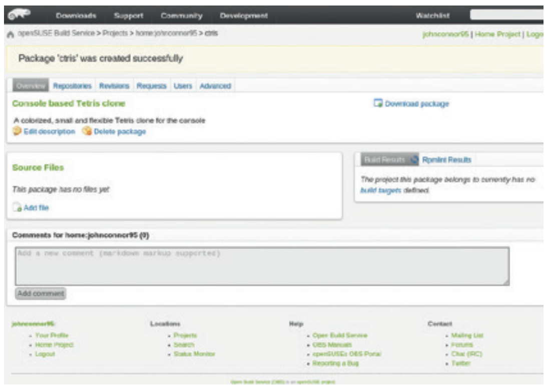
The software isn't the only aspect that needs some work. OBS also helps free software engineers to harness the power of the open source development model – which also helps users because they get a tightly integrated software solution. OBS brings together people who collaborate on the integration of free and open source software; each of the thousands of software projects is maintained by one or more developers that collaborate with each other. This is the third basic challenge a free software distributor solves for its users.

Learning the specifics for each distro

For those who want to dive into the vast realm of software packaging, all major Linux distributions have great introductions to their package formats and the best practice processes that we suggest beginner packagers should read:

- **RPM** http://fedoraproject.org/wiki/How_to_create_an_RPM_package
- **Deb** https:/wiki.debian.org/IntroDebianPackaging
- **Pkgbuild**: https://wiki.archlinux.org/index.php/Creating_packages

The GitHub repository for this tutorial is at **https://github.com/hennevogel/obs-tutorial**, and contains build instructions for all kinds of different distributions.



openSUSE Build Service > Projects > home:johnconnor95 > ctris

Package 'ctris' was created successfully

Console based Tetris clone

A colorized, small and flexible Tetris clone for the console

Source Files

This package has no files yet

The project this package belongs to currently has no build targets defined.

Comments for home:johnconnor95 (0)

Add a new comment (markdown markup supported)

johnconnor95:
- Your Profile
- Home Project
- Logout

Locations
- Projects
- Search
- Status Monitor

Help
- Open Build Service
- OBS Manuals
- openSUSEs OBS Portal
- Reporting a Bug

Contact
- Mailing List
- Forums
- Chat (IRC)
- Twitter

Open Build Service (OBS) is an openSUSE project.

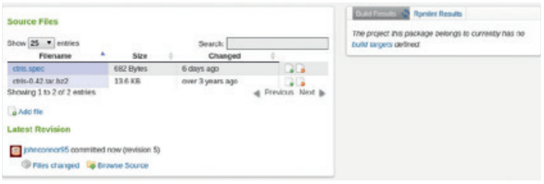
Your OBS home project is ready to start.

Integration

The OBS formalises this collaboration into workflows that all engineers use. Everybody uses the same way to submit new software, to update existing software to a new version, to submit bugfixes and features. Everyone uses the same means to branch, study, change, and improve the software. Thus many different distributors utilise the Open Build Service for the benefit of the users: build binary packages for a wide range of distributions and architectures from source code in an automatic, consistent, reproducible and collaborative way.

But there's more: OBS itself is free software. You can run, copy, distribute, study, change and improve it. The source code and the developers are on GitHub (**https://github.com/OpenSUSE/open-build-service**). As free software, OBS can keep up with the ever-evolving ecosystem, which constantly produces new distributions, new package formats, new architectures, software, standards and tools.

At the current time, OBS supports more than 20 different distributions, half a dozen architectures and three different package formats. It can cryptographically sign your software and products; different instances of OBS can connect to each other; and OBS can be used in conjunction with source code revision systems like *Git*/GitHub in continuous integration workflows.



Source Files

Show 25 entries Search:

Filename	Size	Changed
ctris.spec	682 Bytes	6 days ago
ctris-0.42.tar.bz2	13.6 KB	over 3 years ago

Showing 1 to 2 of 2 entries

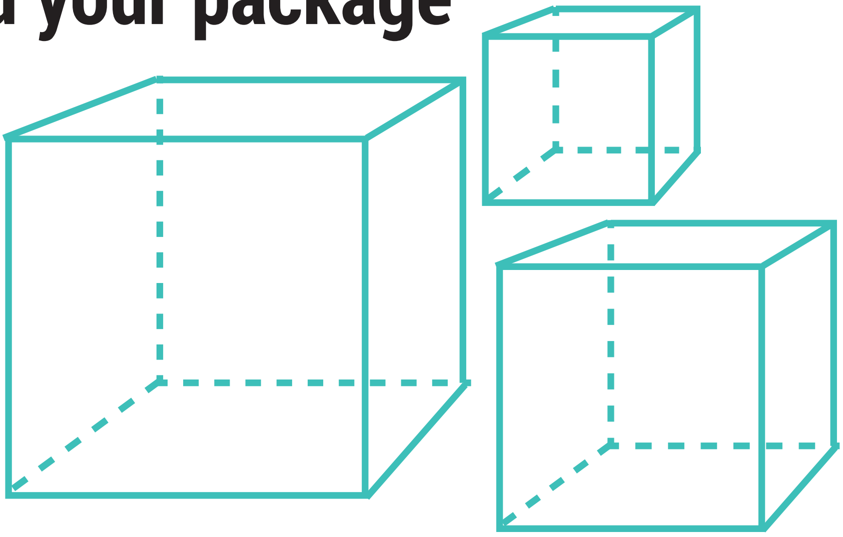
Latest Revision

johnconnor95 committed now (revision 5)

The project this package belongs to currently has no build targets defined.

The new package has been filled with source code files.

Step by step: Build your package



Enough talk, let's get practical! The following directions will show you how to make available your own free software to a bunch of distributions via the build service. It will use OpenSUSE's reference server, which is freely available for developers to build packages for the most popular Linux distributions including OpenSUSE, Debian, Fedora, Ubuntu, Arch, Red Hat Enterprise Linux and SUSE Linux Enterprise.

Building packages is the most basic feature of the OBS. All you need to for that is a modern web browser (we recommend *Firefox*) and an account on the OBS reference server.

1 Create an account and log in

Point your browser to **https://build.opensuse.org** and click the "Sign Up" button in the upper-right corner. Fill out the required information (please take note of SUSE's Privacy Statement at **www.suse.com/company/policies/privacy**) and create your account. Once finished, log into the OBS (the "Log In" link is also in the upper-right corner).

2 Create your home project

Every user in the OBS has their own little space for building packages, called "home project", it always has the name of **home:USERNAME**, for example **home:hennevogel**. You can also navigate to your home project via the link in the upper-right corner.

After first login you'll have to create it, and that's why you are greeted by a form. You can give your home project a nice title and a description of its content, and once you're done, click on "Create Project". In the next step you're greeted by your own shiny little space on the OBS reference server to build your packages. Hooray!

An OBS project can contain zero or more OBS packages. For now your home project is empty, but that's boring so let's create your first package!

3 Create a package

Find the "Create package" link and click on it. Another form creeps up on you. Fill in the information like this: "Name": ctris. "Title": Console-based Tetris clone. "Description": A colorised, small and flexible Tetris clone for the console. Once you're done press "Save changes". This will lead you to your new package.

An OBS package can contain multiple files that are needed to build it. But for now your **ctris** package is empty. Let's change this by uploading some files.

4 Upload files

Typically an OBS package consists of two parts: the source code archive and the build description. As this is a tutorial on how to use the OBS and not about getting you started with packaging we have prepared some files to exercise with. Please download the source code archive (**www.hackl.dhs.org/data/download/download.php?file=ctris-0.42.tar.bz2**) and the RPM build description (**https://raw.githubusercontent.com/hennevogel/obs-tutorial/master/ctris.spec**) onto your computer. Then use the "Add file" dialog to upload them to your OBS package.

Now your OBS package contains everything it needs to produce RPM packages that your users can install; however, nothing has happened yet! Did you notice the "Build Results" box next to your files? It tells you the last missing piece you need: the project that this package belongs to currently has no "build targets" defined. Alright, let's change that – click on the "build targets" link in that box.

5 Add build targets

Build targets are basically the Linux distributions that you want the package to be available for, such as Debian or Arch Linux. On the OBS reference server you're greeted by an impressive collection of build targets to choose from. For this tutorial let's pick some RPM-based distributions: OpenSUSE 13.2 and Fedora 22. After you have added the build targets you are redirected to the configuration for them; don't bother with that (yet) and click straight on the "Overview" link in the upper-left corner.

6 Building

Now the magic happens! Do you notice the "Build Results" box? If you did everything right it will have kicked of your first OBS build already. Your **ctris** package will be in one of the following states: "scheduled", "building" or "succeeded". If you click on those links the OBS will take you to the log file of this build that you can inspect, follow and download.

7 Download binary packages

Once all four build targets are in the state "succeeded" you can click on the "Download Package" link, which will lead you to a page that includes instructions on how to download and install the **ctris** package for Fedora and OpenSUSE.

9 THINGS WRONG WITH WINDOWS 10

We banish **Graham Morrison** to the isolation chamber to test out the latest from Microsoft and report objectively on his findings.

Three years after the release of Windows 8, more people are still using Windows 7 by a huge margin: 60% vs 13.12%, according to Net Applications statistics from June 2015.

Thanks to tablets and smartphones, Linux is now the dominant operating system, at least in Android form. 48.61% of devices shipped with Android in 2014, compared to 14% for Windows, according to a Gartner report from January 2015. We won't even mention that 97% of the top 500 fastest supercomputers use the Linux kernel, the others mostly running UNIX. Oh, sorry, we just did. But like most old brands, Microsoft Windows and its office suite enjoys plenty of momentum. As if to put an impermeable force field between its past and the future, Microsoft has skipped Windows 9 entirely, passing through an event horizon to release Windows 10, which we've been playing with occasionally since the Technical Preview.

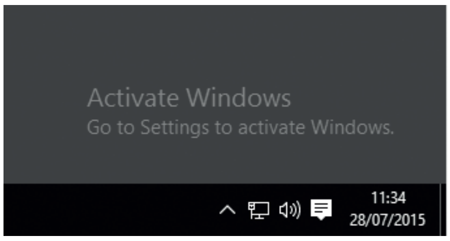
"A major release of a competing operating system is a great time for a little proselytising."

We know we're preaching to the converted. But as the Free Software Foundation's executive director, John Sullivan, says in this month's interview, a major release of a competing operating system is a great time for a little proselytising. Or as Sun Tzu wrote, "If you know the enemy and know yourself, you need not fear the result of a hundred battles." So that's what we're going to do.

1 Product activation

This one's first because it's one of the first things you have to do when you install any Microsoft operating system. Product activation takes your product serial number, generates a unique hash from your system's hardware configuration, and makes sure the serial is being used according to the licence Microsoft is granting you. We've had many conversations with Microsoft in the past when activation has been declined, probably due a hardware change, and each time we've needed to beg someone on the end of a support line to re-instate our system. With Linux, you can install the same copy on 1 machine or 1,000 machines. You don't need anyone's permission, whether that's a Raspberry Pi, your router, a laptop, multiple virtual machines or cloud instances.

As Windows 7 and 8 users are getting a free upgrade, it's more likely you'll need to upgrade or replace your machine at some point, and it's unlikely you'll be able to move your Windows 10 licence across. You will be able to perform a clean install on your new machine, even if you upgraded from a previous version of Windows, but you'll still need to be careful with upgrades. It's unclear what will happen if you don't activate your machine. You get a constant reminder, and the system remains 'non-genuine', but it's too early to say whether features will be removed and updates held back unless you activate your system.


Activate Windows
Go to Settings to activate Windows.
11:34
28/07/2015

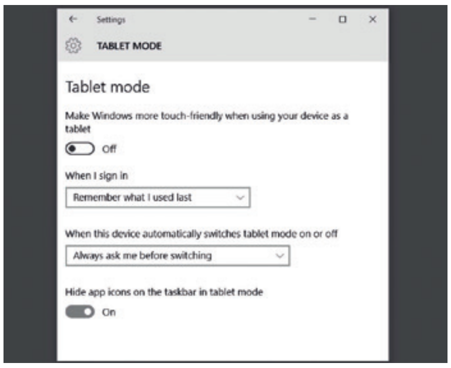
It may be a free upgrade, but you're still not free to install Windows wherever and whenever you like – unlike Linux.

2 Desktop

In some ways, Windows has been going through a similar design transition to Gnome and Unity. This was most evident in Windows 8, where Microsoft seemed hellbent on making everything touchable, draggable and full-screen, despite very few of its users wanting to interface with their computers via a touch display.

The biggest change was the disappearance of the old Start menu. Windows 8 replaced this with a full-screen launcher, where the search field required a mouse drag to the top-right and the power controls were hidden beneath an arrow pointer and page scroll. We reckon this is one of the main reasons Windows 8 has had such poor adoption.

This confusion continued to applications, where some were redesigned for this new full-screen world, and some weren't. If you hacked your way back to the old Start menu, Windows 8 started to feel broken, like running *GTK 3* apps on KDE. Things have improved with Windows 10 (there's a setting that switches 'Tablet Mode' off), but it's a paradigm shift away from what Linux offers, where your desktop can be exactly what you want it to be, and this is unlikely to change regardless of where Microsoft takes Windows.

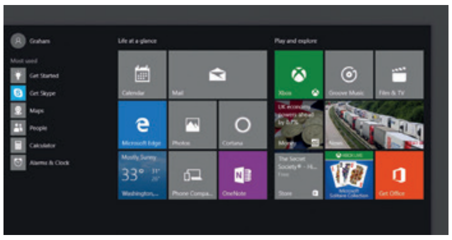

Settings
TABLET MODE
Tablet mode
Make Windows more touch-friendly when using your device as a tablet
Off
When I sign in
Remember what I used last
When this device automatically switches tablet mode on or off
Always ask me before switching
Hide app icons on the taskbar in tablet mode
On

It's now possible to switch between tablet and desktop modes, but there's little other control over the visuals.

3 Launcher

Don't tell your friends, but the launcher in Windows 10 is vastly improved over Windows 8, and Microsoft has done a great job at taking a step back from the previous version and stealing what worked while dropping what didn't. The new launcher is one-third 'Start' menu, one-third 'Metro' styled tablet interface and one-third context search. All this fits into the lower-left quarter of the screen, and feels much more in-keeping with old versions of Windows. It's probably our favourite thing about the new version of Windows, and it's something we think is going to be integral to people migrating from Windows 7 or 8.

As such, the new launcher could be Windows 10's biggest threat to Linux adoption, because there isn't anything quite as powerful while remaining old-school familiar. Gnome 3.x is more capable, in terms of what you can do from the launch input – from virtual desktops and context switching, to customising your distribution. Unity too is doing remarkable things with Scopes, adding shopping, search news and email in a way that makes sense on both mobile and desktop. KDE has a few menu systems that are augmented in a similar way to Windows 10, but there's nothing that embeds the social side, the live tiles, the design success of what's going on in Windows 10. Which is why we're listing the launcher here. It's actually good!

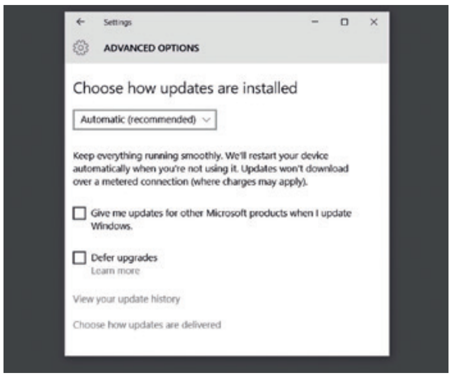


We dislike the Windows 10 launcher because it's good, and it's going to keep people using Windows.

4 Forced updates

Of course, updates are essential. But updates for Windows are typically larger and more frequent than Linux updates, and they can take a long time to install. When the frustration gets too great, it has always been possible to delay updates or to disable the feature entirely, but not any more. From Windows 10, updates are going to be mandatory, at least for Home users. Pro and Enterprise users can defer updates.

This is important for a couple of reasons. Firstly, these enforced security updates are going to reduce the number of Windows machines running malware, which make up, by far, the majority of compromised computers running on the internet. But there's a more draconian side to updates too, and that's the likelihood that features will be removed that you don't want to be removed. It's the lack of user control that worries us, especially as the idea you no longer have control over what you own is growing – whether that's a Kindle removing *1984*, or Apple removing Google Maps. Linux distributions will never have these problems. You can choose not to update your machine, or select only those updates that are relevant to you. We've also got more trust in almost any Linux distribution to only roll out security patches.


Settings

ADVANCED OPTIONS

Choose how updates are installed

Automatic (recommended)

Keep everything running smoothly. We'll restart your device automatically when you're not using it. Updates won't download over a metered connection (where charges may apply).

☐ Give me updates for other Microsoft products when I update Windows.

☐ Defer upgrades
Learn more

View your update history

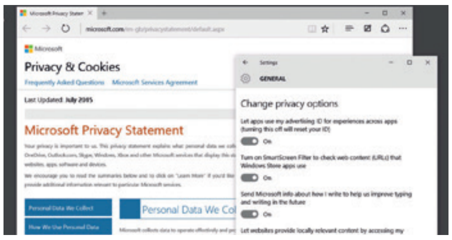
Choose how updates are delivered

Windows 10 Home users will be forced to update their machines, whereas Pro and Enterprise users will be able to defer an update.

5 Privacy

With Linux, it's clear when your data is leaving the system, or when you're using services that store your data on external services. You don't have to agree to intrusive processes running through your photos or email unless you want them to, and you don't have to agree to privacy decimating end-user agreements. The same isn't true of Windows 10. Microsoft's new Services agreement, effective from 1 August 2015, is 12,000 words long. Microsoft's Privacy agreement, fully expanded, is 17,000 words. These agreements are so ambiguous that they seem to grant Microsoft access to every facet of your computer use.

A big part of the new agreement seems to be designed to build a profile much like Facebook's user profile, "We collect data about your interests and favorites, such as the teams you follow in a sports app, the stocks you track in a finance app, or the favorite cities you add to a weather app," is a good example from the Privacy statement. Microsoft also creates an 'Advertising ID' for each device you activate. "Your advertising ID can be used by app developers and advertising networks to provide more relevant advertising, " according to Microsoft, and major new features such as the Cortana 'personal assistant', which takes voice input and generates sensible output like Google Now or Siri, is another worrying addition for privacy advocates.


Microsoft Privacy Statem

microsoft.com/en-gb/privacystatement/default.aspx

Microsoft

Privacy & Cookies

Frequently Asked Questions Microsoft Services Agreement

Last Updated: July 2015

Microsoft Privacy Statement

Personal Data We Collect

How We Use Personal Data

Personal Data We Col

Settings

GENERAL

Change privacy options

Let apps use my advertising ID for experiences across apps (turning this off will reset your ID)

On

Turn on SmartScreen Filter to check web content (URLs) that Windows Store apps use

On

Send Microsoft info about how I write to help us improve typing and writing in the future

On

Let websites provide locally relevant content by accessing my

The privacy panel in Windows 10 puts all your privacy options in one place, but it can't stop your data being shared.

6 Usability

Subtle use of animations and a new, thinner window manager helps Windows 10 feel like a modern operating system. We like it. But Windows 10 has also taken some serious inspiration from Linux with its new Task View. This is essentially a re-implementation of virtual desktops, and while old versions of Windows had this to some extent (or via proprietary graphics drivers), it's being touted as a major feature. For those of us who rely on multiple virtual desktops, it's certainly a step in the right direction. But it's a long way from the facilities currently offered by the average Linux desktop, and a whole dimension short of KDE's Activities, where a set of virtual desktops can be pre-configured per-application or use case.

The new notifications system has taken inspiration from OS X, which took inspiration from Linux. They appear in their own area, now called the Action Centre, which slides in from the right of the desktop. It's a big improvement over the chaos of notifications in Windows 8, and we'd like to see something like this on a Linux desktop. Linux has become very good at dealing with notifications. Gnome's notification system is improving with each release, and we currently prefer the way it displays messages. Ubuntu's Scopes also work well here, adding real interaction to updates that are usually passive. But we'd like to see Windows 10-like integration, even if as Linux users, we still have more control.

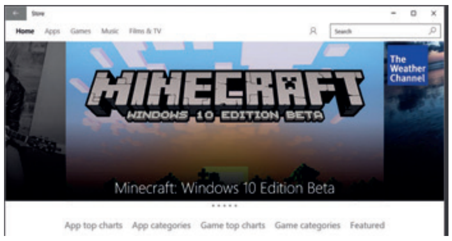
LINUXVOICE
FEATURES
TUTORIALS
PODCASTS
FORUMS
SHOP
ABOUT
Linux Voice is a different kind of Linux magazine
SUBSCRIBE TODAY! from just £38
LINUXVOICE CONTENTS
LATEST
DOWNLOAD LINUX VOICE ISSUE 8

Virtual desktops make a welcome return as first-class citizens in Windows 10.

7 App store

This is a big category because of the way both operating systems handle the installation of new software. Like Apple, Microsoft is using security as an excuse for making it harder to install downloadable **.exe** files. And it has a point, as random executables downloaded from the internet are probably responsible for more viruses than any other feature. But its replacement, Microsoft's Store, is a conduit designed for sucking your privacy and your money. This is a huge difference between Linux and Windows, of course, and it has more to do with the culture behind both operating systems. Under Windows, it's typical even for small utilities to be crippled, or run as shareware. Whereas, as Linux users, we sometimes take for granted the incredible selection of software we have available, freely installable through your distribution's package manager without anyone making notes on what you're installing.

Microsoft Office is also a huge part of Microsoft's strategy, and while it's not yet part of Windows, its download and management has been part of the Windows eco-system for some time. However, with the release of *LibreOffice 5.0* in early August 2015, with its open source and open standard credentials, we think switching from proprietary office software to an open source alternative is imperative, even when people still need to use Windows.



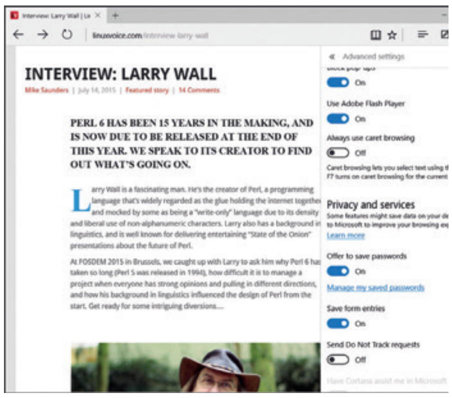
App stores, such as the one in Windows 10, are really package managers with built-in DRM, and must be resisted.

8 Web browsing

Once upon a time, the thought that *Internet Explorer* would no longer be a dominant force in web browsing seemed like a fairytale. But the rise of Google, rapidly accelerated JavaScript engines and serious security flaws have slowly eroded *Internet Explorer*'s viability.

So Microsoft has started again, bundling a new browser that it's calling *Microsoft Edge* (previously Project Spartan). *Edge* is a huge improvement over the bloated anachronism of IE. Its sharp, borderless UI is the best example of Windows 10 design. The URL doesn't even appear until you click on the top. It's also fast, finally competing with the JavaScript engines of the other browsers. But it's also new, and its rendering engine – *EdgeHTML* – is entirely proprietary, rather than the open source *KHTML*/*WebKit*/*Gecko* derivations of *Safari*, *Chrome*, *Opera* and *Firefox*. As a result, we've still far more confidence in the future of these than *EdgeHTML*.

The other big omission is that *Microsoft Edge* has no support for extensions, although Microsoft has said in the past it should support *Chrome*'s and *Firefox*'s. This means no ad blocking, or any of the other extensions that lots of us rely on as part of our web experience.


Interview: Larry Wall | Li
linuxvoice.com/interview-larry-wall
Advanced settings

INTERVIEW: LARRY WALL

Mike Saunders | July 14, 2015 | Featured story | 14 Comments

PERL 6 HAS BEEN 15 YEARS IN THE MAKING, AND IS NOW DUE TO BE RELEASED AT THE END OF THIS YEAR. WE SPEAK TO ITS CREATOR TO FIND OUT WHAT'S GOING ON.

Larry Wall is a fascinating man. He's the creator of Perl, a programming language that's widely regarded as the glue holding the internet together and mocked by some as being a "write-only" language due to its density and liberal use of non-alphanumeric characters. Larry also has a background in linguistics, and is well known for delivering entertaining "State of the Onion" presentations about the future of Perl.

At FOSDEM 2015 in Brussels, we caught up with Larry to ask him why Perl 6 has taken so long (Perl 5 was released in 1994), how difficult it is to manage a project when everyone has strong opinions and pulling in different directions, and how his background in linguistics influenced the design of Perl from the start. Get ready for some intriguing diversions....

Block pop-ups: On
Use Adobe Flash Player: On
Always use caret browsing: Off
Caret browsing lets you select text using t… F7 turns on caret browsing for the current…

Privacy and services
Some features might save data on your de… to Microsoft to improve your browsing ex…
Learn more

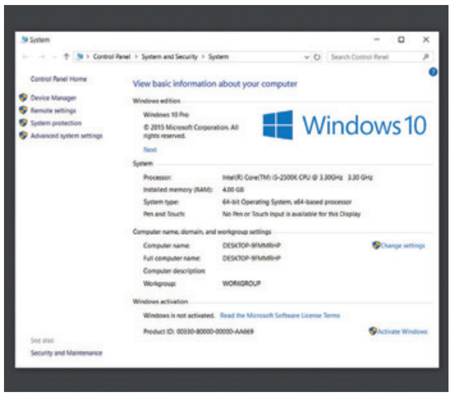
Offer to save passwords: On
Manage my saved passwords
Save form entries: On
Send Do Not Track requests: Off
Have Cortana assist me in Microsoft

If *Edge* doesn't work, you can still use *Internet Explorer*, which isn't dead yet.

9 Closed source

Our last gripe is the most fundamental, and while it's not something new to the release of Windows 10, the fact that Windows is proprietary is still its biggest hindrance. Open source is better for so many reasons. For security, Microsoft is usually very quick to patch problems when they occur. But there's no transparency about how long an exploit has been exploited, or what was needed for the fix, or even how major one vulnerability might be in comparison to another. Even after several open source projects have had their own vulnerabilities exposed, the transparency and self-regulation that accompanies open source is a far healthier, and we think, far more secure system. The same is true for privacy. It's almost a given that there are backdoors into Windows, sending data back.

While the same could be said for Linux, if there are backdoors, they're from a lack of eyeballs rather than machiavellian intrigue. We're far more likely to trust a messaging application when the source code is available, than promises of good faith from a proprietary offering (cough, *Skype*). But open source is also more innovative. Many of the ideas taken and integrated into both Windows and OS X started life as ideas on the Linux desktop. LV



View basic information about your computer

Windows edition
Windows 10 Pro
© 2015 Microsoft Corporation. All rights reserved.
Next

System
Processor: Intel(R) Core(TM) i5-2500K CPU @ 3.30GHz 3.30 GHz
Installed memory (RAM): 4.00 GB
System type: 64-bit Operating System, x64-based processor
Pen and Touch: No Pen or Touch input is available for this Display

Computer name, domain, and workgroup settings
Computer name: DESKTOP-9FMMRHP
Full computer name: DESKTOP-9FMMRHP
Computer description:
Workgroup: WORKGROUP

Windows activation
Windows is not activated. Read the Microsoft Software License Terms
Product ID: 00330-80000-00000-AA669

Imagine not being able to check or share the code running through your CPU....

UNPROTECTED COMMUNICATION

Mass surveillance violates our fundamental rights and is a menace to the freedom of speech! But: we can defend ourselves.

The password that protects your email is not sufficient to protect your mails against the mass surveillance technologies used by secret services.
Each email sent over the internet passes through many computer systems on the way to its destination. Secret services and surveillance agencies take advantage of this to read millions and millions of emails each day.
Even if you think you have nothing to hide: Everyone else that you communicate with via unprotected emails is being exposed as well.
Take back your privacy by using GnuPG! It encrypts your emails before they are sent, so only the recipients of your choice can read them.

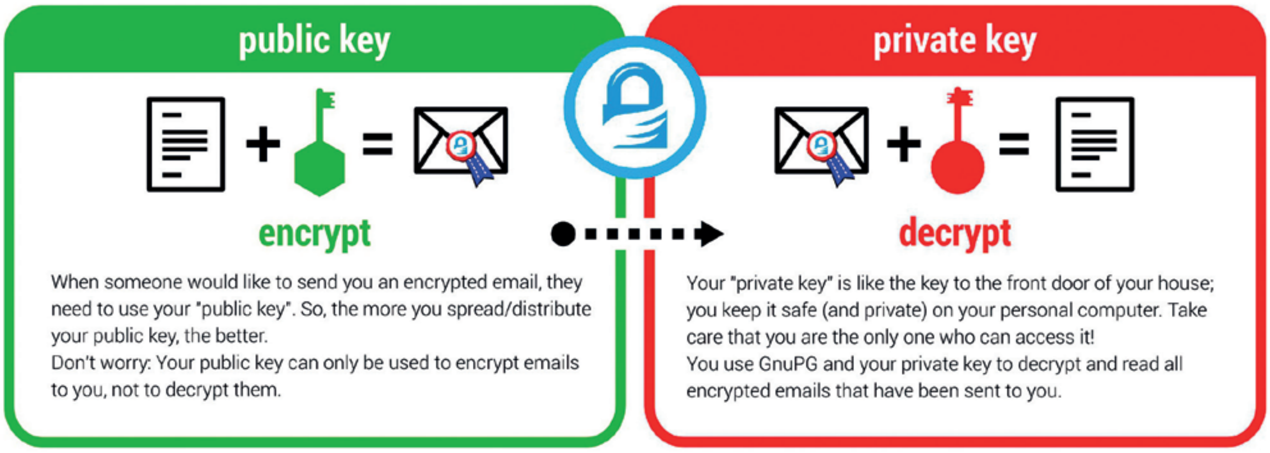
GnuPG is platform independent. That means it works with every email address and runs on pretty much any computer or recent mobile phone. GnuPG is free and available at no charge.

About GnuPG

Thousands of people already use GnuPG, for professional and private use. Come and join us! Each person makes our community stronger and proves that we are ready to fight back.
Whenever an email that is encrypted with GnuPG is intercepted or ends up in the wrong hands, it is useless: Without the appropriate private key it cannot be read by anyone. But, for the intended recipient - and only for her - it opens like a totally normal email.
Sender and recipient are both safer now. Even if some of your emails contain no private information, consistent use of encryption protects us all from unjustified mass surveillance.

What makes GnuPG secure?

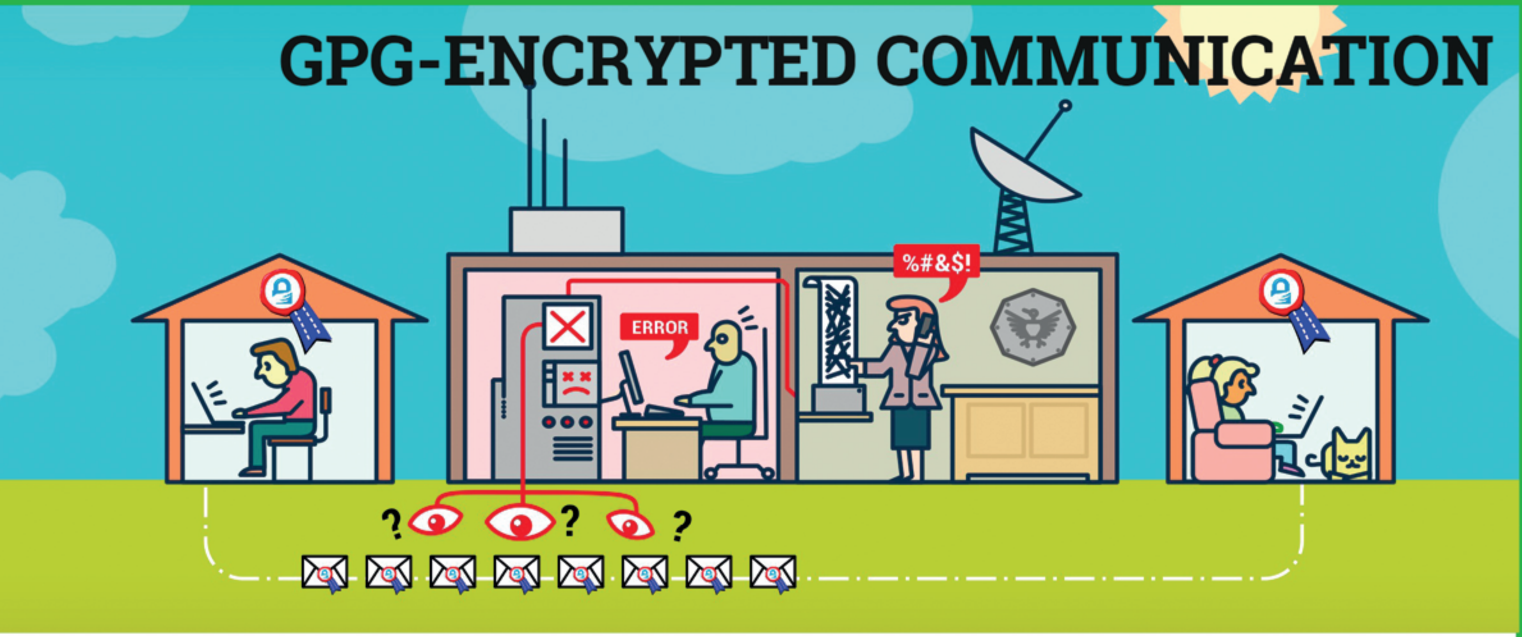
GnuPG is Free Software and uses Open Standards. That is essential to be sure that software can really protect us from surveillance. Because in proprietary software and formats, things might happen beyond your control.
If no one is allowed to see the source code of a program, nobody can be sure that it does not contain undesirable spy programs - so-called "backdoors". If software does not reveal how it works, we can merely trust it blindly.



When someone would like to send you an encrypted email, they need to use your "public key". So, the more you spread/distribute your public key, the better.
Don't worry: Your public key can only be used to encrypt emails to you, not to decrypt them.

Your "private key" is like the key to the front door of your house; you keep it safe (and private) on your personal computer. Take care that you are the only one who can access it!
You use GnuPG and your private key to decrypt and read all encrypted emails that have been sent to you.

GPG-ENCRYPTED COMMUNICATION



In contrast, a fundamental condition of Free Software is to publish its source code: Free Software allows and supports independent checking and public review of the applied source code by everyone. Given this transparency, backdoors can be detected and removed.
Most Free Software lies in the hands of a community that works together to build secure software for everyone. If you want to protect yourself from surveillance you can only trust Free Software.

Practical advice

The technology behind GnuPG provides first-class protection. To ensure that your encrypted communication is not compromised for other reasons, use a strong passphrase and backup your private key. Encrypt as much as you can! By doing so, you prevent others from realising when and with whom you exchange sensitive information. Thus, the more often you encrypt your messages, the less suspicious encrypted messages will be. Be aware that the subject is transmitted unencrypted!

You can find a simple tutorial for email self-defense with GnuPG encryption here: EmailSelfDefense.FSF.org

Watch out for so-called "Cryptoparties" in your area! These are events where you can meet people that would be happy to help you in setting up and using GnuPG as well as other encryption tools at no charge.

About the fsfe

This article was created by the Free Software Foundation Europe (FSFE), a non-profit organisation dedicated to empower people in Europe in their use of technology by promoting software freedom.

Access to software determines how we can take part in our society. Therefore, FSFE strives for fair access and participation for everyone in the information age by fighting for digital freedom. Nobody should ever be forced to use software that does not grant the freedoms to use, study, share and improve the software. We need the right to shape technology to fit our needs.

The work of FSFE is backed by a community of people committed to these goals. If you would like to join us and/or help us to reach our goals, there are many ways to contribute. No matter what your background is. You can learn more about this under: **fsfe.org/contribute**

Donations are critical for us to continue our work and to guarantee our independence. You can support our work best by becoming a sustaining member of the FSFE, a "Fellow". By doing so, you directly help us to continue the fight for Free Software wherever needed: **fsfe.org/join**

What is Free Software?

Free Software can be used by everyone for any purpose. That includes free copying, reading the source code and the possibility to improve or adapt it to your own needs (the so-called "four freedoms").

Even if you "only want to use" the program, you still benefit from these freedoms. Because they guarantee that Free Software remains in the hands of our society and that its further development is not controlled by the interests of private companies or governments.

Find out more about this and how Free Software can lead us into a Free Society: **fsfe.org/freesoftware**

If you like to help spreading the word, you can order this and other leaflets under: **l.fsfe.org/promo**

fellowship of fsfe

FAQ WebAssembly

Does the web need another binary format, and can a new one avoid the pitfalls of the past?

BEN EVERARD

Q WebAssembly? I've never heard of it, but I guess it's something to do with the internet.

A Yep. It's a new binary format for code that runs in the web browser. Basically, it will enable programmers to compile languages to create executable files that will run inside web browsers on any platform. For example, the same executable should run in *Firefox* in Linux and *Internet Explorer* on Windows. This binary format will also be processor agnostic, so will run on ARM mobile chips, x86 desktops and any other processors that you may find.

Q What is this binary format you speak of? A bytecode similar to Java or Flash?

A As far as the user is concerned, it's probably best to think of WebAssembly as a binary executable similar to Java or Flash, but one that runs directly in the browser, so can change the whole page in the same way as you can with JavaScript, rather than being restricted to a particular object on the page in the same way that Java and Flash are.

If you really get down to the details of WebAssembly, it's actually implemented in quite a different way to bytecode. Bytecode is a machine code for a virtual machine that runs in software and can be implemented on many different platforms. WebAssembly, on the other hand, is a binary representation of the abstract syntax tree.

A simple way of thinking of this is that it's the midway point in compiling code. So, rather than raw code (like JavaScript), or pseudo-machine code (like Java), WebAssembly is half-compiled code. Much of the most-time consuming part of compiling has already been done, but the platform-dependent part of is still to be done by the browser. None of this should matter to the users, but technically, WebAssembly isn't a bytecode.

"WebAssembly will give programmers a language other than JavaScript for the web."

Q So if it has the same capabilities as JavaScript, and it runs in the same model as JavaScript, why don't we just use JavaScript? It's been around for ages and seems to be working OK.

A WebAssembly is designed to be like JavaScript but faster. Every time your browser visits a website that uses JavaScript, it has to download and parse the language, and this takes time and memory. For simple scripts, this is negligible; however, web designers are putting more and more functionality into web apps, and the number of scripts web browsers have to parse is getting larger and larger.

This isn't usually a major problem for desktop machines, but on lower-power devices such as phones, some larger scripts can take tens of seconds just to parse. If that has to happen on every page, it's a significant slowdown in browsing speed.

Q Isn't a better solution just to consider excessive use of JavaScript a bug, and just start shouting at web designers until they use less of it?

A There are certainly sites that use too much JavaScript, in fact, there are quite a lot of sites that use too much JavaScript. However, there are also some really good uses of large amounts of JavaScript. Take, for example, online document editing, browser games or even video codecs. These are always likely to be JavaScript (or WebAssembly) -heavy, but they're also really good uses of technology.

Even if you don't want to move all your programs to web apps in the cloud (and there are very good reasons why you shouldn't), having the option to use web apps when it is appropriate is very useful. In other words, good web tech is a bit like freedom of speech. Lots of people abuse freedom of speech to spout silly or offensive views, however it's still vital because without it, important things can't be said. In the same way, lots of programmers abuse web tech to make silly interactive graphics that just convolute the interface and slow down browsers, but it's still important, because without it, there are whole classes of web app that can't be made.

Another advantage of WebAssembly is that it will enable programmers to use languages other than JavaScript to program for the web. There are also quite a few languages that compile to JavaScript already, but compiling to WebAssembly should make them run faster and easier to debug.

Q I run NoScript to stop websites running code in my browser. Is this just another attempt by advertising companies to get inside my browser? How can I stop it?

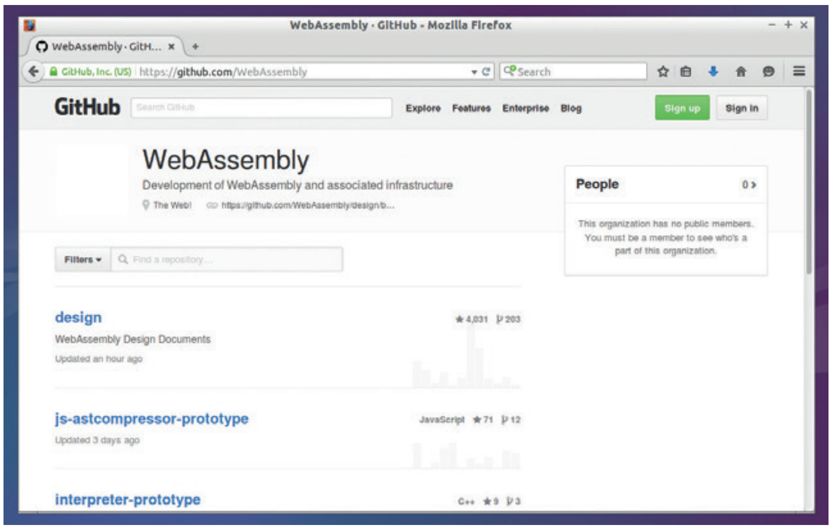
A WebAssembly will run in the JavaScript engine, so won't be another way in. From a privacy and security perspective it should be no different from running JavaScript. At the same time, it should be possible to block it in the same way. We expect NoScript and other extensions to be updated in due course to protect those who don't wish to run WebAssembly.

Q So what's assembly got to do with it?

A Actually very little. The name is a little bit of a misnomer. Assembly language is a text language that directly maps to binary machine code. WebAssembly, on the other hand, is itself binary. There is a text version of WebAssembly that programmers can write in directly, but this is known as the WebAssembly textual representation rather than WebAssembly assembly (or WebAssembly^2). Even the binary format isn't machine code, so a textual version of it isn't assembly anyway.

Q Ah, that brings me onto my next question: How will people program WebAssembly? Using this textual representation?

A While it will be possible to code directly in WebAssembly textual representation, it's unlikely that many people actually will for the same reason that not many normal people program directly in regular assembly language



WebAssembly

Development of WebAssembly and associated infrastructure

The Web! https://github.com/WebAssembly/design/b...

This organization has no public members. You must be a member to see who's a part of this organization.

design

WebAssembly Design Documents

Updated an hour ago

js-astcompressor-prototype

Updated 3 days ago

interpreter-prototype

The final version of WebAssembly isn't quite ready yet, but development is happening on GitHub, where you'll find the design documents and prototypes.

(Mike Saunders, who wrote an entire operating system in assembly, cannot be considered normal by any reasonable metric).

Initially, the aim is to make it easy to compile C and C++ code to web assembly. The reason for this is that there are currently a lot of large performance-critical codebases in C and C++ that would benefit from a fast web-based version. Games and Video Codecs are obvious examples, but anything that is too big or slow to run on the web would be a candidate.

In time, it's likely that compilers will expand to support more languages compiling to WebAssembly. There are no fundamental restrictions on what languages can be compiled; it will simply depend on what languages people are willing to take the time to support. It's even possible that JavaScript will be compiled to WebAssembly.

Q Ah, that brings us back to JavaScript. How will WebAssembly work with JavaScript?

A There are so many existing skills in JavaScript that it would be foolish to expect it to go away anytime soon. In cases where you just need a little code, JavaScript currently runs fast enough that there won't be any significant speed increase from WebAssembly, so the simplicity of JavaScript may outweigh the negatives – like the way people still write in Python despite C code running faster.

Since WebAssembly runs in the same engine as JavaScript, the two will be able to talk to each other. We expect to start seeing libraries in WebAssembly that are designed to be used from JavaScript. This should enable people to gain the load and performance of WebAssembly, but still use an interpreted language that they're familiar with.

Q The history of the web is littered with the corpses of binary formats: ActiveX, Flash and Java to name but a few. One factor in the demise of all of them was that not all browsers supported them, and if you needed users to install a plugin to work with your site, then you'd lose visitors. How is WebAssembly going to avoid this?

A WebAssembly has a real advantage here because it can run quite efficiently in a JavaScript polyfill. That means that if someone with an older browser visits your site, and it relies on WebAssembly, you can load a JavaScript library that will convert the WebAssembly into JavaScript and run that. Obviously this has a performance impact, but early indications are that the performance of the polyfill is quite good.

Not only that, but this is the first binary format to have support from all major browser developers. Mozilla, Google, Apple and Microsoft all have people working on WebAssembly, so it should be widely supported in new web browsers. LV

"Use the GPL if you want to make sure that nobody turns your software into proprietary software."

JOHN SULLIVAN

FSF EXECUTIVE DIRECTOR

Graham Morrison meets one of the only people in the world of Free Software who Richard Stallman can legitimately call 'boss'.

Founded by Richard Stallman in 1985, the Free Software Foundation is now 30 years old. It predates the world wide web, Linux, laptops, smartphones, social media and the Amiga 500. As the custodian of the GPL, the Free Software Foundation puts the GNU in GNU/Linux, and it's vital for the future and well-being of our favourite operating system, and a future where we can all request and change the source code. As its Executive Director, John Sullivan has a tough job making sure the FSF sticks to its principle while adapting to an ever-changing world of technological development. The GPL is being used more than ever before, and Free Software now requires little introduction, as he explained when we met him for a chat.

LV The GPL has been vital for the growth of Linux, but is it becoming marginalised by permissive licences such as Apache and MIT?

John Sullivan: I gave a talk at FOSDEM a few years ago about the what had been written about the decline of copyleft licences, and I do have a lot of issues with that premise. I'm planning to write an article to lay them all out because these studies keep getting published periodically like the GitHub one that was published a little while ago, showing licence preferences on GitHub.

LV But there's also a feeling, especially around big open source companies like Facebook, Twitter, that permissive licenses are becoming a developer's first-choice.

JS: One thing to keep in mind is that it's not just a question of whether you're choosing copyleft or choosing permissive licences; we've added another element to this, which is that we now have companies that would normally either not distribute their software or at all, or have been distributing it as proprietary software. [These companies] are a lot of the ones doing a lot of the permissive distribution. We have to keep in mind that it might not be trading off with GPL adoption, as much as trading off with proprietary software licences. We're not just counting two categories here, you have to also count the third category. The number of programs that are released under a Free licence is way larger than it ever has been.

LV That's important to remember.

JS: There's definitely a lot of talk, again, about why younger developers are not choosing copyleft. One specific case for that seems to be codified in the GitHub 'Choose a licence' tool. We would really like them to change the language that they use to describe the GPL, to make it parallel with the other licences – MIT, BSD-style, as they call it, and Apache. The GPL is described in a way that would make people not want to choose it. It focuses on restrictions. We would like the language to focus on protections, so, "Use the GPL if you want to make sure that nobody turns your software into proprietary software", it prevents that from happening, as opposed to "choose the GPL if you want requirements like this."

I think it would probably have some influence, on choices and somebody else, if maybe someone else were to provide a different tool that would accomplish the same ends. But really I'd just like to see them improve their wording, make it a bit more clear and less judgemental.

Google not releasing the source code to Android is a prime example of why we need the FSF to preserve Free Software for everyone.

What can you do to help spread the word?
JS: I think that one thing we could do to advocate copyleft better is to compile the cases that give us concern with permissive, lax licences. I don't think we've done that work yet, just as other people maybe aren't giving the consideration they should. We also haven't done the work to present the information as well as we could. Particularly in the spaces where code is being made proprietary in a way that disrupts other development. People depend on Android. People depend on the compiler. And to have bits and pieces of that taken away screws up other people's work – that's something I want to avoid.

The challenge of all this is that a lot of people who would prefer that software not be copyleft are businesses – like Facebook, as they said in the keynote the other morning [at OSCON], "We do it in the end because it's good for the company." And we have no argument with that. We still welcome their participation in Free Software, but we need people to recognise that it's contingent that way because a lot of this attitude comes from larger companies, with lots of resources, up against a bit of a marketing.

Another example of why GPL is so important is in the revelations that The Hacker Group had used GPL software to build its tools, much to the initial chagrin of Free Software developers. But because they took GPL'd software, it also means those developers have some course of action against those hackers, should they want to take it.
JS: Also, in that case in particular, the only way to ensure that your programs aren't being used against you is to be able to see what they're doing, and GPL is the only licence that guarantees that.

We need to do more for just providing assistance. One activity we've done for a very long time is just to provide, free of charge, help to developers to understand the licence and use it properly. And it's not just the GPL – it's also interactions with other Free Software licences. I'd like us to be doing more in that area. We need to be doing more to push back against this idea that GPL is somehow too complicated. It's really not. The GPL functions as a licence that can be used effectively in jurisdictions all around the world to distribute any kind of software – even things that aren't software – and that's how many words it takes. A single website has the same number of words in governance, so it's really not that complicated. Complications are over stated.

Yes, and it's easier for development teams to publish stuff within a company without consulting their legal department.
JS: Yeah, the problem is that patents are still an issue for that. Companies wanting to hold them back as a weapon they can use.

So, perhaps more optimistically, what are your feelings on free software at the moment? Do you feel good about it, and take any pride at the success of conferences like this?

"The GPL functions as a licence that can be used effectively in jurisdictions around the world."

JS: Yeah. There are plenty of people here who support us, who do not fully agree with the idea that there should be no proprietary software, they just think free software is really good. So, even though we may agree that we're not as close to the goal, there are a lot of people getting together talking about sharing code both commercially and non-commercially and the benefits of doing that.

One of the dangers that we're seeing is free platforms being used just as ways to run proprietary software. Like, Netflix has several talks here and they're doing a lot of work in free software and sharing that, and using a lot of free software internally, but in the end they're using it to deliver DRMed media that requires proprietary software on the user side. So that's taking us farther away from some important goals. And I think that's one of our major threats right now and it's connected to the copyleft question, which is: how do we can stop literally every exciting popular technological development in the last several years that has had free software involved in it somewhere? You know, Netflix, iPhone, iOS, Android, Amazon Kindle, they all have free software in them.

LV And it leads us to the difficulty of dealing with proprietary software used in the cloud.
SF: With network services that take the programs out of the user's hands we can see how powerful free software is, and how wrong the argument that if software is free then nobody will develop software anymore. Obviously false, but it's not getting us closer to the goal of users actually being respected. And that's where you would find the differences between us and some of the companies represented at this conference. We care about why people are using free software more than what they're using it for.

LV Are there companies that are particularly responsive to the message of the FSF? Are there companies that embrace the obligations of the GPL and GNU and make it part of their ethos?
SF: Yeah, for one thing there are some smaller companies that are doing really amazing things. We have a hardware endorsement program, called the "Respect Your Freedom" program, and the companies that we're working with under that banner like Aleph Objects, ThinkPenguin, Gluglug in the UK. They are making products that are completely free, so I guess that's the pinnacle of what we have right now.

And I think there's a lot of room in that space. I'm really excited about that program, because I think one of the most important things we need to be doing is make it easier for regular people to buy a free system or device without having to do so much research. Lots of things work with free software, but for years Wi-Fi and Bluetooth required proprietary blobs, and laptops have proprietary Wi-Fi and proprietary graphics acceleration a lot of times. But if you were to line up all the different devices, you could take from each of them and put them into one package, you'd get pretty close to having a fully free option. So I think it's just a case of single businesses not making the right decisions. The components are there, and I think if more companies started doing that they would find a market for that kind of product, especially now with the privacy concerns.

Being able to sell a product and say you can see everything that it does is potentially powerful if somebody with a marketing budget were to pick it up and run with it. But there are plenty of companies sharing the source code properly under the GPL, there have been forever, but unfortunately, especially in the mobile space, there are lots of companies that need to be doing more about, otherwise companies that are following the terms have a legitimate gripe about playing by the rules...

Contributing to Free Software isn't just about being a good citizen – for companies, it can also be good marketing material.

LV But then resources are really limited – perhaps 40 people in

total including WordPress, Software Conservancy and the FSF. It must be hard to pick the fights!
JS: Well that's part of the reason I'm excited about the endorsement program. If we can just get people to make fully free products then there are no GPL issues, because everything is already free software. That's the easiest way to resolve that. We announced the **copyleft.org** project a while back… the whole idea is that individuals are supposed to be able to request source code for a device and see the source code for a device to be able to build it, and while we're doing the enforcement work ourselves, we really ought to be doing more to help other people exercise those freedoms. Plus I think that it will have an impact on the space if we get more people just trusting the source code.

Part of the compliance issues with some if these companies is that they don't get requests frequently enough to have a system, and so they don't treat them the right way. That's not an excuse, but I think it would probably improve things if more people were actually requesting the source code.

John told us that the IT infrastructure at the FSF runs on Trisquel, the totally Free Software Linux distribution.

That's a great point. Do you think attitudes towards Free Software have changed?
JS: We've definitely seen much bigger interest outside the hacker/developer core of supporters that got the whole thing started. We had a member write to me once and ask "How do I install GNU/Linux?", for example, and it was kind if a wake up call for me because I knew that our membership had a variety of skill levels and interests and in the most parts, they actually care about working with computers in terms of developing software. But I'd always thought that everybody had at least reached the stage of being able to try installing GNU/Linux. And it's really encouraging actually, because it means that this person joins because he likes the ideas. So that was really cool. Since any kind of advocacy now requires using computers, I think we're seeing a lot more interest from other politically oriented advocacy groups.

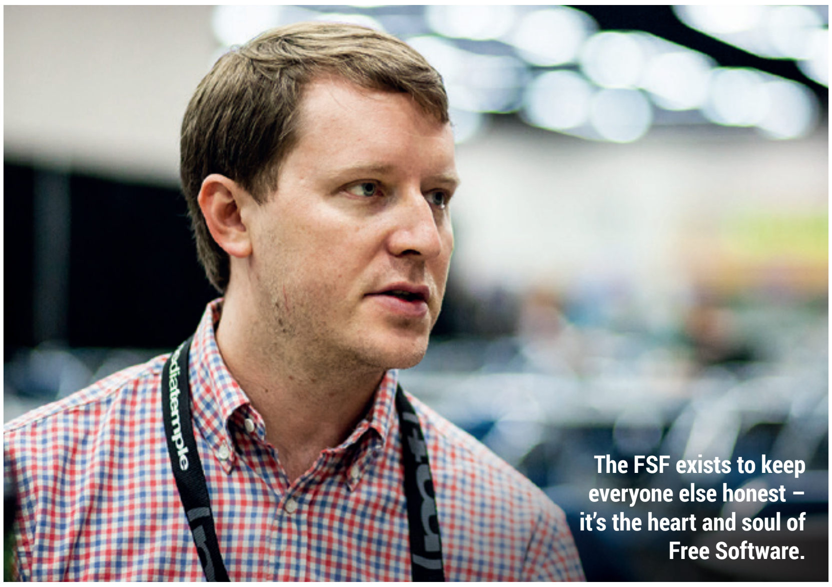
Has this been after the Edward Snowden revelations?
JS: Not entirely. I think even before then, when (Microsoft) Vista came out we had Greenpeace to support our campaign against Vista, because they understood our relationship between proprietary software and forced obsolescence. There's been, since I started in 2003, a rise of interest from other political groups. The idea of being in control of your own communications, your own tools that you use to connect to your supporters, so that no company can just yank that out from underneath

> "We're seeing a lot more interest from other politically oriented advocacy groups."

you. Organisations still have a time getting started with Free Software but they're more interested in it than they were 12 years ago, for sure.

Are you doing anything for the Windows 10 launch?
JS: Definitely. We have an on-going 'Upgrade from Windows' campaign, and we'll update when we know about Windows 10. One thing we know is that updates are now mandatory, so users can't disable the software updates any more, which has great potential for abuse, since updates have been known to be used to take features away that users like, or install anti-features.

We also have an update in our Restricted Boot campaign because there may be some changes with Windows 10 around SecureBoot, and we'll definitely be doing more because those launches are good opportunities – Free Software people tend to not like to be too critical. Folks would rather


The FSF exists to keep everyone else honest – it's the heart and soul of Free Software.

In his non-FSF time John's a Debian developer (even though it's not completely non-free).

send the positive message of Free Software rather than 'Windows is bad', or 'Mac OS' is bad. At the same time those launches are great moments for reaching new people. People who have just got Windows 8, they don't want Windows 10, so they start searching for alternatives and our campaign that mentions Windows 10 and GNU/Linux together is a way for people to discover what they're searching for.

Searching for information on Windows 10 and finding a different OS sentirely, that's a gateway – it's not just about bashing, it's about linking our option, alternative or our replacement, whatever people are looking for because that's what they know already.

LV How did you get involved with Free Software Foundation?
JS: When I was a teenager I was very active. I was a wannabe hacker, I ran a bulletin board system. I tried to learn to do some programming. And then I got away from that completely in college because of the math class requirements, and studied philosophy and creative writing instead. I ended up applying for a job at the Free Software Foundation to work on the manuals, because I wanted a part time job that would be related to publishing. Then I worked part time on my own writing, but then when I got working at the FSF, all of my computer geek stuff came back again. I went from part time to full time and got back to learning a little bit about programming again. It was really consistent with other political values that I had, and I hadn't had that really strong political components to my early experience with computers. And bringing those things together was powerful for me.

LV How many full time staff are there at the FSF in the US?
JS: We have 12 right now. We've been growing, so we've added new positions in the last few years. I would like to see us be much larger. Part of my goal as the Executive Director is to try and make that happen.

LV Is funding the main issue?
JS: We're about 85% funded by individuals and then the rest, there's a little bit from sales of manuals and T-shirts and then the rest from companies and other large non-profits that contribute funds. I really like that the vast majority of our funding comes from individuals, because that's who ultimately we want to be accountable to. But I also think we could be receiving a lot more funding from businesses that are involved in Free Software.

LV With so many companies reliant on what the FSF has achieved, it's perhaps surprising they haven't made more of a financial contribution.
JS: And prospective employees like to see it. They like to see the place that they're going to work be a contributor to groups like the FSF, because it's a great statement about that company's culture and can make the difference in someone's decision between wanting to work for one company versus another company. There's a lot of willingness out there in the business world to support us. But we do have to ask, and we do have to make the case.

LV Is RMS funded from the FSF?
JS: We don't fund his travel or give him a salary. Events normally cover expenses to get him to different places and of course he prefers to stay with local people. But we do have the position of his assistant, and some of the other roles help to support his travel and work.

LISTEN TO THE PODCAST

LINUX VOICE

WWW.LINUXVOICE.COM

BUY LINUXVOICE MUGS AND T-SHIRTS!

LINUXVOICE
FREE SOFTWARE | FREE SPEECH
"IT'S EASIER TO ASK FORGIVENESS THAN IT IS TO GET PERMISSION"
Rear Admiral Dr Grace Murray Hopper

shop.linuxvoice.com

LINUX VOICE REVIEWS

The latest software and hardware for your Linux box, reviewed and rated by the most experienced writers in the business

Andrew Gregory
No Google, I'm not 10 minutes away from the office; that's the pub.

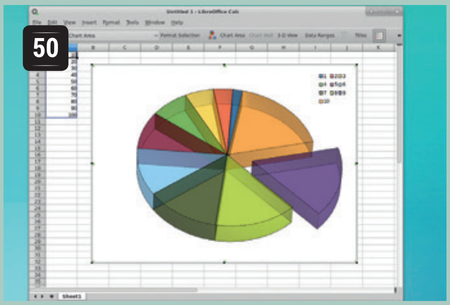
Windows 10 is out, and already the usual suspects are falling over themselves to tell us how amazing it is. What seems to have gone unnoticed in the mainstream media, however, is the intrusive amount of data that Windows 10 reports back to Microsoft and its 'trusted partners.'

By default (you can change the default settings, but the option to do this is hidden in blue text on a blue background) Windows 10 reports your location back to Microsoft; it sends details of your calendar, and your typing. The Edge browser sends your browsing data to Microsoft, which is egregiously bad. But possibly the worst of the default settings is that Windows 10 automatically connect to open wireless hotspots.

Seriously, just use Linux

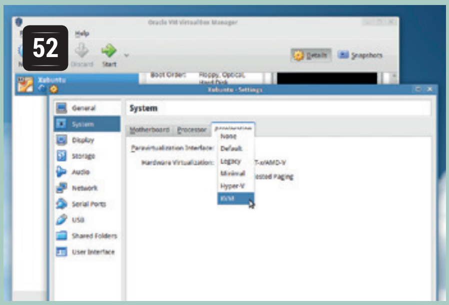
Most people accept the defaults; that's just what we do as humans. Which means that pretty soon there will be millions of devices vulnerable to dodgy routers running malicious software. So there are lots of opportunities here: identity theft is going to be a growth industry, which should spur a new wave of no-win, no-fee legal cases. And maybe, just maybe, we'll see some disgruntled users switching to Linux.
andrew@linuxvoice.com

On test this issue...


50

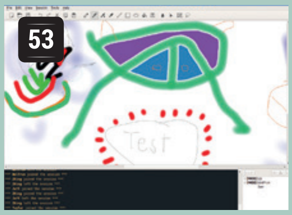
LibreOffice 5

Graham Morrison formats some paragraphs in one of Free Software's flagship applications. Today, open documents, tomorrow, the world!


52

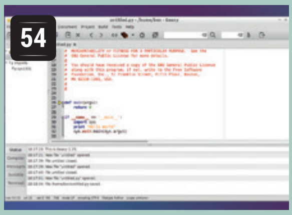
VirtualBox 5.0

More speed and some extra compatibility options for this stalwart virtualisation platform make **Mike Saunders** a very happy distro hopper.


53

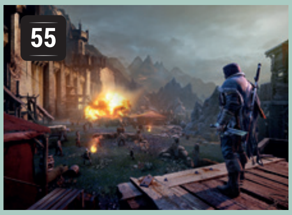
Drawpile 1.0

Collaborate on drawings with teams halfway round the world – **Ben Everard** tests an application eight years in the making.


54

Geany 1.25

Ben Everard turns his back on the Vim vs Emacs editor wars to explore a clean GTK 2- based development environment.


55

Shadow of Mordor

Michel Loubet-Jambert has gazed into the palantír and seen the future of gaming on Linux, and it's based on a book published in 1954.

BOOKS AND GROUP TEST

As we know, Google and Microsoft are trying to take every bit of your personal data and feed it into the monetising machine. But there are other ways to interact with the world. In this issue's Group Test we're exploring search engines for the privacy conscious, including one fascinating option that uses distributed, shared processing rather than sending data through a central server.

And if you're sick of living in Jeremy Bentham's panopticon, you might be interested in a little book by Cory Doctorow – it's fiction, but only just.

LibreOffice 5

Graham Morrison needs to find a collective noun for people who love word processors.

DATA

Web
http://libreoffice.org
Developer
The Document Foundation
Licence
LGPLv3 & MPLv.20

We must admit. Despite their prosaic nature, and a tendency to be associated with open plan offices, we quite like office suites. This may be something to do with the Amiga, and the emergence of proper graphical word processors like *Wordsworth* and *Final Writer*. Even to our younger selves, opening the physical packaging around those (costly) products and patiently installing a single application off several floppy discs was exciting. To be then presented with a WYSIWYG view of your writing, as you typed, seemed revolutionary.

We had the same warm and fuzzy feeling when Sun Microsystems bought *StarOffice* in 1999 and then open sourced the project in 2000. We had the same feeling when The Documentation Foundation was formed to help *LibreOffice* fork itself from the then incumbent *OpenOffice.org*, which remained under the control of Oracle Corporation after its acquisition of Sun. Even now, when installing the latest version of *LibreOffice*, it feels wonderful that we have such a powerful suite of office applications, free in both cost and principle. That there's a graphical database, a spreadsheet, a presentation creator and a drawing tool alongside the wordsmith feels like a free pass to an all-you-can-eat buffet. It's a good job major releases like this don't come along too often.

High five

LibreOffice 5 comes three years after 4.0, but it's also a little unexpected. It was due to be version 4.5, but perhaps because the 4.x release cycle has consistently delivered great upgrades, the project has 'done a Linus' and upped the major number as a demarcation of everything that's been achieved. Notably, we loved the new icon theme that came with 4.2, and the OpenGL and UI overhaul of 4.4, which was only released in January 2015. Our huge datasets have also been grateful for the re-written computation engine in the spreadsheet, along with many other small updates and refinements along the way.

The *LibreOffice* report for 2014, released by The Document Foundation in June 2015, for instance, is worth a perusal just to see how much has been achieved. It's great to know, for example, there were a total of 67,500 donations throughout 2014, raising nearly €595,000.

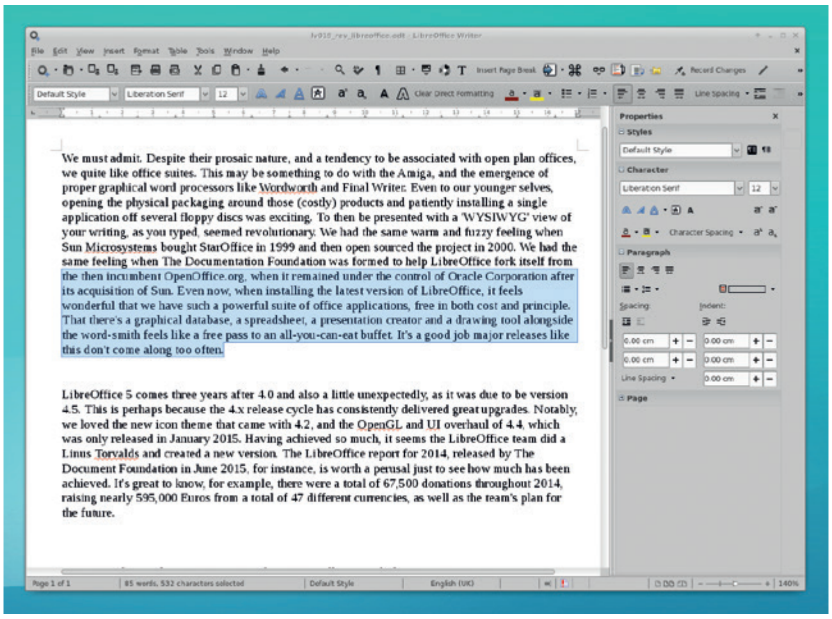
As usual, the release notes that accompany *LibreOffice 5* are longer than a Microsoft EULA. It's good that they're comprehensive, because so many projects fail to document what they've been working on. But they're also too long for immediate gratification. It would be great if alongside the release notes there were a visual overview of what was new and what had changed.

Close to the edit

We'll start with the word processor, *Writer*, as this is likely to be the most commonly used component of the entire suite. There are small graphical refinements everywhere. Some of the menus have changed, and there's a new icon set based on Breeze from KDE. Whether this looks good is subjective, but as this review is written by a KDE user using exactly that icon set on the desktop, it's a very pleasing update, especially when KDE integration always feels a few steps behind *GTK* and what the Ubuntu team does themselves for better unification.

Styles now have a rendered preview, just like the font selector does, which is a great help if you use more than a few styles. If you want to flag your friend's spelling mistakes, text highlighting is now compatible with *Microsoft Word* formats, and images can be cropped with your mouse. We'd like to see the integration of word definitions, perhaps using an offline database, so we can look up a word while writing. We'd also love to see a more configurable writing environment where we can remove all distractions and have an on-screen word count for the total document, paragraph and selection.

Editing and creating text documents, especially if you need *Word* compatibility or if you're working with downloaded templates, works brilliantly. We use it almost every day for label printing, for example, and we'd be lost without *LibreOffice* and its exceptional format support. One of the best new features is the addition of the timestamp protocol to PDF exports.



We must admit. Despite their prosaic nature, and a tendency to be associated with open plan offices, we quite like office suites. This may be something to do with the Amiga, and the emergence of proper graphical word processors like Wordworth and Final Writer. Even to our younger selves, opening the physical packaging around those (costly) products and patiently installing a single application off several floppy discs was exciting. To then be presented with a 'WYSIWYG' view of your writing, as you typed, seemed revolutionary. We had the same warm and fuzzy feeling when Sun Microsystems bought StarOffice in 1999 and then open sourced the project in 2000. We had the same feeling when The Documentation Foundation was formed to help LibreOffice fork itself from the then incumbent OpenOffice.org, when it remained under the control of Oracle Corporation after its acquisition of Sun. Even now, when installing the latest version of LibreOffice, it feels wonderful that we have such a powerful suite of office applications, free in both cost and principle. That there's a graphical database, a spreadsheet, a presentation creator and a drawing tool alongside the word-smith feels like a free pass to an all-you-can-eat buffet. It's a good job major releases like this don't come along too often.

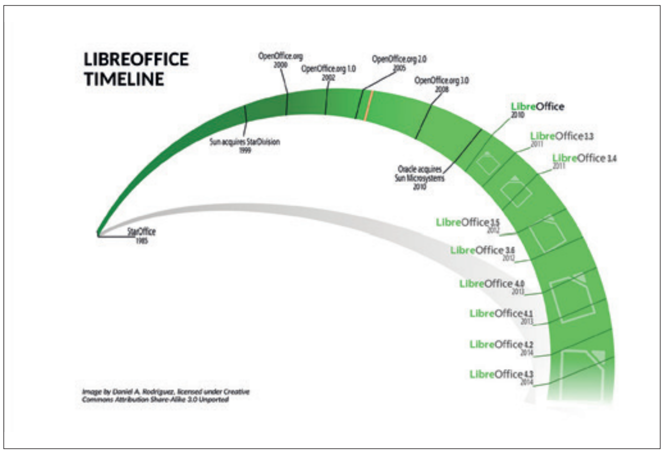
LibreOffice 5 comes three years after 4.0 and also a little unexpectedly, as it was due to be version 4.5. This is perhaps because the 4.x release cycle has consistently delivered great upgrades. Notably, we loved the new icon theme that came with 4.2, and the OpenGL and UI overhaul of 4.4, which was only released in January 2015. Having achieved so much, it seems the LibreOffice team did a Linus Torvalds and created a new version. The LibreOffice report for 2014, released by The Document Foundation in June 2015, for instance, is worth a perusal just to see how much has been achieved. It's great to know, for example, there were a total of 67,500 donations throughout 2014, raising nearly 595,000 Euros from a total of 47 different currencies, as well as the team's plan for the future.

New style previews, more icons and auto-correction are the highlights of this release, and we found the entire suite a step-up in stability.

The Document Foundation

One of the best things about *LibreOffice* is The Document Foundation. This is the charity responsible for its development, and it's a remarkably transparent and accountable organisation. It was built from the *OpenOffice* community after it started to become obvious that a fork was the best direction to take the project.

The Foundation has a board of directors, a membership committee and a board of trustees. Anyone can become a member and get their voice heard, and many companies are doing just that. An additional advisory board includes members from Red Hat, Google, AMD, Intel and even the city of Munich. It's rare in open source that a Foundation like this exists for the betterment of a single project. And like The Mozilla Foundation, the importance of its custodianship cannot be underestimated. That we now have the ability to edit and email documents to people regardless of their operating system or office suite is partly to do with the legacy of *OpenOffice* and *LibreOffice*, and the Foundation is determined to keep pressure on standards commitees to ensure this momentum continues. If you'd like to find out more, we spoke to its Executive Director (he was Chairman at the time), Florian Effenberger in our very first issue, and we chatted with Michael Meeks, VP of Productivity, in issue 05, both of which are now free under the Creative Commons BY-SA licence.



Only a very small percentages of the updates added to *LibreOffice* get merged into the old *OpenOffice*.

This enables you to sign a document with an external authority that guarantees the authenticity of the timestamp. It's easier than getting an image of yourself with today's newspaper, and is a serious requirement for all kinds of legal and archival uses.

Almost too good?

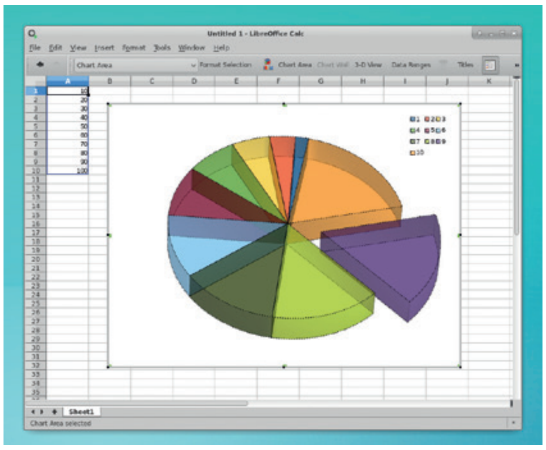
We love new features, but we still think that *LibreOffice* could do with a feature purge, or at the very least, some menu and option pruning. The suite seems to have continued getting more complex for over a decade, despite there being a minor revolution in distraction-free editors and online suites. We know *LibreOffice* needs to compete with *Microsoft Office* and match as many features as it can, but we'd love to see a genuine overhaul of the interface, and we don't mean by adding the abysmal ribbons.

The spreadsheet has had some cosmetic changes made to the data bars, plus conditional formatting, which can now be exported to XSLX. There are also a handful of new spreadsheet functions to improve compatibility with *Excel*. We still experienced stability issues when using large spreadsheets. This may be because *Calc* has received a lot of internal work for this update, as part of the computational engine rewrite. Similar updates have been made to the other applications in the suite. *Draw* and *Impress* both look smarter, and the entire suite works well with our High DPI display now.

LibreOffice is undoubtedly looking to the future, with the new editable Android app and the soon-to-be-released cloud version. This is particularly clever, because it encapsulates the real code and functionality of the native application, rather than being a web interface connected to an API. We know of no other online office suite that's as comprehensive, and *LibreOffice* could genuinely find success with cloud/online users who want something more comprehensive than the low-fat office suites currently available.

"We can't imagine Linux being a viable alternative to OS X or Windows without LibreOffice."

LibreOffice 5.0 is another strong release, tempered perhaps by its own successes and the expectations that come with a major version number. We may be beholden to *LibreOffice* for its cross-platform compatibility, but development has never stalled. Each release adds hundreds of features, fixes and improvements. It still has rough edges – especially around the GUI – but we can't imagine Linux being a viable alternative to OS X or Windows without it.


Untitled 1 - LibreOffice Calc
File Edit View Insert Format Tools Window Help
Chart Area
Format Selection
Sheet1
Chart Area selected

Version 5 could have been version 4.5, as it contains mostly cumulative updates, but the spreadsheet does get faster and more stable.

LINUX VOICE VERDICT

We'd like to see some UI rationalisation, but version 5.0 is a strong update to this cornerstone of FOSS success.

★★★★☆

VirtualBox 5.0

Mike Saunders spends half of his life inside VirtualBox, and finds out what's new in version 5.0.

DATA

Web
www.virtualbox.org
Developer
Oracle
Licence
GPL/CDDL (base)

If we had to name one piece of software that has been absolutely instrumental in the making of Linux Voice, it has to be *VirtualBox*. Sure, there are plenty of other PC emulation and virtual machine options out there, but we've always had a soft spot for *VirtualBox*: it's very easy to use, it's a good performer, and it has all the features we need for testing and reviewing Linux distros (and indeed other operating systems). For instance, it'd be hard to imagine life without snapshots, which let us test new distros or poke around inside them, and revert to the previous state quickly if/when something goes wrong.

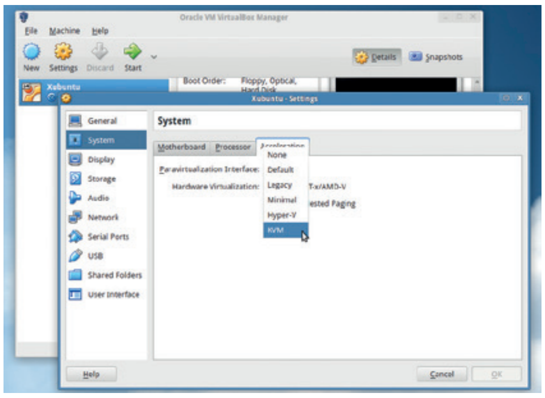
Chances are that you've used *VirtualBox* at some point as well, probably for the same purposes or perhaps for running the odd Windows program on your Linux desktop without having to reboot. Originally a product of German company Innotek GmbH, it was acquired by Sun in 2008. Just two years later, Oracle snapped up Sun, which led to nail-biting times for *VirtualBox* fans. Oracle's relationship with the free software community has been topsy-turvy at best, and many Linux users doubted that *VirtualBox* would remain open source. But luckily, it still is – although there are some closed-source extensions providing better USB support along with RDP and PXE booting.

Distro-Inception!

VirtualBox 5.0 is available as a 60MB download, and thankfully, the developers have created packages for all the major Linux distributions including Ubuntu, Debian, Fedora and OpenSUSE. If you're using a different distro, however, you can try the **.run** version (available in 32-bit and 64-bit editions) which you just need to make executable and run. It's also worth noting that *VirtualBox* runs on other OSes as well, such as Windows and Mac OS X, so if you're forced to use one of those proprietary OSes in your workplace, maybe you're still allowed to install *VirtualBox* and get your Linux fix...

"The biggest new feature in VirtualBox 5.0 is support for paravirtualisation."

The biggest new feature in *VirtualBox 5.0* is support for paravirtualisation. Essentially, this enables guest machines (ie the OS you're running inside *VirtualBox*) to work more closely with the real hardware of the host. With fewer layers of virtualisation and emulation between the guest OS and the host machine, performance should be improved. *VirtualBox 5.0* can use *KVM* on Linux for this purpose, or if it's running on Windows, Microsoft's *Hyper-V*. Of course, if you'd rather stick with the normal *VirtualBox* virtualisation you can disable this – go to Settings > System and you'll find the option under the Acceleration tab.



Boot Order: Floppy, Optical, Hard Disk

Paravirtualization Interface: KVM

Hardware Virtualization: VT-x/AMD-V, Nested Paging

With paravirtualisation, *VirtualBox 5.0* can now use Linux's *KVM* for an extra speed boost.

Additionally, more CPU instructions are now available to guest OSes, such as SSE 4.1/4.2, AVX, AES-NI and RDRAND. A virtual xHCI controller has been added, which provides USB3 device support in guests, and bi-directional drag-and-drop support has been implemented so you can move files around more easily. It's now possible to scale the output of the virtual display, so you can shrink it down and leave it in the corner of your screen if you're waiting for a process to finish and a dialog box to pop up. Or you can start VMs headless, and attach and detach virtual displays to them later on.

Another major update is encryption for the virtual hard drive images. You'll find this under Settings > General > Encryption, and two AES ciphers are supported. This is very handy if you're working on a confidential project inside a VM but don't have full disk encryption on your host OS enabled. Then there's HiDPI support, hotplugging for SATA disks, a new modular audio architecture, and minor updates and bugfixes all over the code.

So in all, *VirtualBox 5.0* is a mightily impressive release and genuinely worthy of the major version number bump. We've never had big issues with *VirtualBox*'s performance (at least, not for the sort of distro testing work we do), but the speed increases, scalable display, USB 3 support and disk encryption keep *VirtualBox* competitive with the other options on Linux. All we can hope now is that Oracle remains a good custodian of the project and it will have a healthy future for the next few years. LV

LINUX VOICE VERDICT

Bags of new features and performance improvements mean this stays our favourite VM.

★★★★½

Drawpile 1.0

Ben Everard can't draw, but he does enjoy scribbling on the internet.

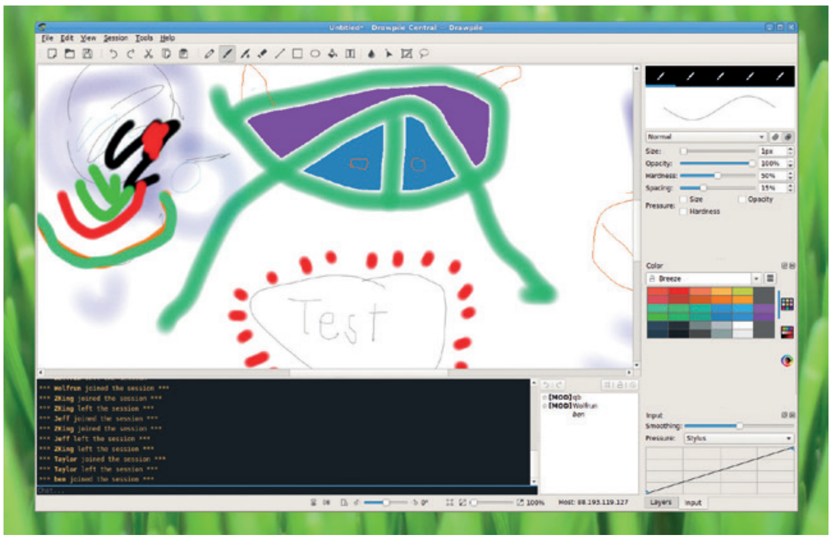
Collaborative working via cloud applications is becoming an increasingly important part of many organisations' workflow. There are plenty of ways to work together to produce text or code even when you're thousands of miles away from your collaborator. *Drawpile* brings the same sort of instantaneous collaboration as *Etterpad* or Google Docs to artists.

It's not hard to see that collaboration could work the same way in this field as it can in others. Concepts can be sketched out, worked on and discussed by groups without these groups having to be physically together. *Drawpile* supports the Open Raster file format, which is also supported by other free software drawing applications such as *Krita*, so artists can switch between multiple programs for different stages in the work's creation.

Unlike many online collaboration tools, *Drawpile* doesn't run in a web browser. Instead it's a standalone program that acts as both client and server. When you first launch it, you can use *Drawpile* as you would any other raster image editor to open, manipulate and save images. You can then enter a collaboration by either hosting a session or joining one hosted by someone else. In a session, all members can edit the image and see the edits made by other people. There's a chat session built into the *Drawpile* window, but many people may prefer to use this software alongside another collaboration tool such as a WebRTC video chat.

It's not always possible to host your own server because of various restrictions with network setups. *Drawpile* can get around this by letting users host sessions on remote servers. You can also announce public sessions and invite other artists to join you.

Drawpile is easy to use, but there's still a excellent help section on the website to help if you get stuck.



```
*** Wolfrum joined the session ***
*** ZKing joined the session ***
*** ZKing left the session ***
*** Jeff joined the session ***
*** ZKing joined the session ***
*** Jeff left the session ***
*** ZKing left the session ***
*** Taylor joined the session ***
*** Taylor left the session ***
*** ben joined the session ***
```

Of course, more artists doesn't always mean more quality. This is the result of 14 artists over 15 hours.

Currently running public sessions are listed on the project website at **http://drawpile.net/servers**.

The brushes in *Drawpile* aren't as advanced as those in, say, *Krita*, but that's to be expected. This is the first stable release of the software, so naturally the features aren't as complete as more mature drawing tools. It does, however, have all the basic tools artists need including layering and the ability to work with Wacom's stylus devices.

Work in progress

Drawpile enables users to record sessions, so that they not only have the finished image, but a record of how they worked together to create that image.

The documentation of *Drawpile* is particularly impressive given that it's the first stable release. It's not only detailed and accurate, but well presented and easy to navigate, which is something that a lot of more mature open source projects could learn from. Head to **http://drawpile.net/help** to find out how to use the software.

Overall, we really like *Drawpile*. It is a little lacking in features compared to pure drawing tools, but at the other end of the spectrum, it's a far nicer drawing experience than the shared whiteboards that many video conferencing tools have. It's easy to use and a great way of working together with other artists (or clients) to create pictures. LV

LINUX VOICE VERDICT

Drawpile is now our favourite way of scribbling nonsense across the web (it can also be used for serious work).

★★★★☆

DATA

Web
http://drawpile.net
Developer
Calle Laakkonen
Licence
GPL v3

Geany 1.25

In a never-ending quest for a better programming environment, **Ben Everard** and his trusty pet monkey starts rubbing lamps.

DATA

Web
www.geany.org
Developer
Enrico Tröger and Geany Project
Licence
GPLv2 or later

There are almost as many ways of writing code as there are programmers. A few hardy souls still insist that *Ed*, the Unix text editor, is the best way to code. Slightly less masochistic than these are the people who prefer to use a powerful text editor (such as *Vim*) with separate compilers, etc. At the opposite end of the spectrum are those who prefer to use integrated development environments (IDEs) that bring everything together in one massive program.

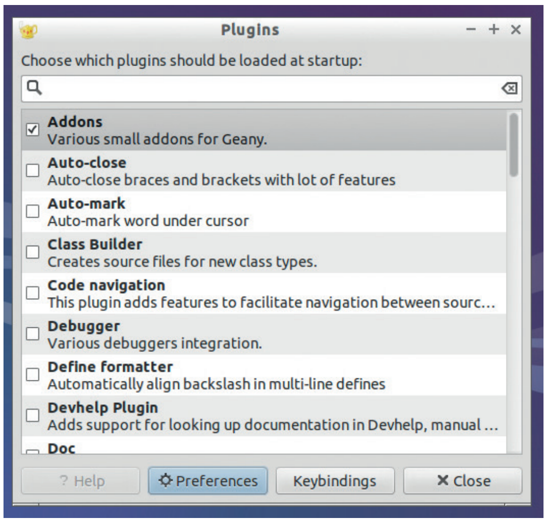
The trick with programming environments isn't waiting around for a perfect one that will be all things to all programmers, but finding one with the right set of features for your programming style. *Geany* occupies a middle ground. It's not a fully featured IDE, yet it's more powerful (from a programmer's perspective) than most text editors.

"Through judicisous use of plugins you can adapt Geany to your way of working."

All the basic programmer's features are present: syntax highlighting, code folding, integration with compilers and interpreters, and terminal support are all present and working for just about every major language. Compared to a serious IDE, there's less support for managing projects and there's no GUI builder, while compared to a *Vi* or *Emacs* setup, there's less flexibility in how everything links together.

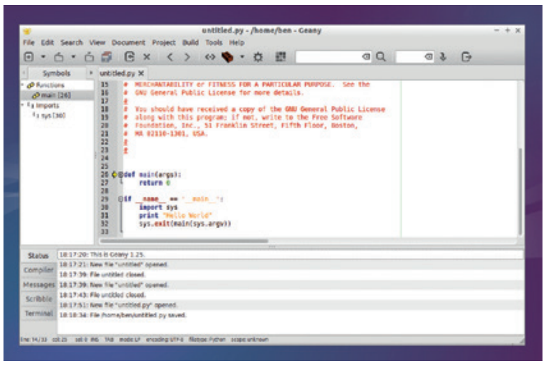
New in version 1.25 is support for *GTK 3*. By default, *Geany* is still shipped with *GTK 2*, but if you're using a desktop environment built on the newer toolkit, you can switch and get an interface that's more in-keeping with the rest of your software.

You can extend the features of *Geany* via plugins, and there's a standard set of plugins though this is often included in a separate package in Linux distros (called **geany-plugins** in many package managers).



[✓] **Addons**
Various small addons for Geany.
[ ] **Auto-close**
Auto-close braces and brackets with lot of features
[ ] **Auto-mark**
Auto-mark word under cursor
[ ] **Class Builder**
Creates source files for new class types.
[ ] **Code navigation**
This plugin adds features to facilitate navigation between sourc...
[ ] **Debugger**
Various debuggers integration.
[ ] **Define formatter**
Automatically align backslash in multi-line defines
[ ] **Devhelp Plugin**
Adds support for looking up documentation in Devhelp, manual ...
[ ] **Doc**

Plugins give *Geany* the power to compete with more complex development environments.


untitled.py - /home/ben - Geany
File Edit Search View Document Project Build Tools Help
Symbols
untitled.py
Functions
main [26]
Imports
sys [30]

```
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to the Free Software
#  Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
#  MA 02110-1301, USA.
#
#


def main(args):
    return 0

if __name__ == '__main__':
    import sys
    print "Hello World"
    sys.exit(main(sys.argv))
```

```
18:17:20: This is Geany 1.25.
18:17:21: New file "untitled" opened.
18:17:39: File untitled closed.
18:17:39: New file "untitled" opened.
18:17:43: File untitled closed.
18:17:51: New file "untitled.py" opened.
18:18:34: File /home/ben/untitled.py saved.
```

Geany is more powerful than its interface suggests.

If you want the full features, you'll need to make sure you install this as well as the standard *Geany* package. Alternatively, you can browse the available plugins and just get the ones you want at **http://plugins.geany.org**. Once they're installed, you can add and remove plugins via the Tools > Plugin Manager. It's these plugins that really lift *Geany* above general text editors and can add features such as a debugger, a pretty printer and more. Through judicious use of plugins, you can adapt *Geany* to your way of working, though the scope for customisation isn't as large as it is in some text editors.

Spread the code

Geany works on Linux, Windows, Mac OS X and several BSDs, so it's a good option if you want to have the same programming environment on more than one OS. Because so much is built-in, you can have almost the same experience on every environment.

Geany does its job well, but not everyone will appreciate it. If you program a lot and are happy with your programming environment, you're probably not going to gain much by switching to *Geany*. The limited customisation means that you may have to adjust the way you work slightly to fit with *Geany* rather than the other way around. However, if you're new to programming, or only program occasionally, then *Geany* is worth a try. The simple interface is ideal for smaller projects, and the features are easy to use without having to learn shortcuts or wade through deep menus. It's a great compromise between the complexity of an IDE and a simple text editor.

LINUX VOICE VERDICT

As a programming environment for intermediate and occasional programmers, *Geany* is hard to beat.
★★★★★

Middle-earth: Shadow of Mordor

Michel Loubet Jambert finds out if one of the biggest games to land on Linux meets all the hype and expectations.

When we're getting critically acclaimed AAA games like *Shadow of Mordor*, based on the huge *Lord of the Rings* franchise, it's easy to forget just how far Linux gaming has come in these last few years. The story of *Shadow of Mordor* takes place between the events of *The Hobbit* and *The Lord of the Rings*, featuring both new and old characters true to the Middle-Earth canon. The game puts the player in the shoes of Talion, a Gondorian ranger out for revenge in a world turned upside-down by Sauron's dark forces.

Ironically, story is perhaps *Mordor*'s weakest area. The player is immediately presented with a few somewhat rushed cutscenes and action sequences where Talion's sole motivation to avenge his family is revealed. The player is taken to the action as quickly as possible with little backstory, as if the developers have a very pessimistic view of the attention spans of the millennial generation. Though the game's plot is coherent and unfolds at a good pace, it continues to feel rushed and condensed at times. That said, the voice acting and music make up for a lot of this, while well-written and plentiful codex entries draw on the world's rich lore, providing a lot of much-needed context to those who want it.

I smell man flesh

The combat and game mechanics are where Mordor really excels. Being able to take on dozens of orcs simultaneously with an impressive fast-paced and responsive combat system is incredibly satisfying. When needing to block an attack, Talion immediately stops any other action and focuses on the task at hand, and the game always seems to do exactly what the player wants throughout combat, making it feel very fluid. There is also a huge variety of abilities and special moves which, along with the three available weapons, provide many different ways of taking on the orcs. The game places an emphasis on player choice, and while all these tools remain at the player's disposal, it makes no attempt to shove them down your throat.

In other areas of gameplay, *Mordor* draws from an array of different genres, incorporating stealth, open world and strategy elements. The game world is not as big as that of other games like *Skyrim*, but it's big enough to get sidetracked in and necessitate fast travel to go to different areas. The stealth is similar to that of the *Assassin's Creed* series, with plenty of climbing, sneaking and use of environmental objects, though *Mordor* perhaps pulls it off better. The strategy elements of the game are the most original and intriguing of its features, showing the hierarchy of Sauron's army evolving as the player kills off captains and warlords, engaging in power struggles and planting sleeper agents. Similarly, each encounter with an orc leader is unique, with enemy-specific dialogue and traits, while encounters make the player feel as if they are having a significant impact on the world.

The port itself is excellent, and at the standard expected from a professional porting house. We found no bugs or visual quirks, and performance is at similar levels to what Windows gamers would get. The visuals are also impressive, and if you're fortunate enough to have a high-end card, it's one of the best-looking games out there. On Nvidia cards, the game recommends the latest drivers, so players may find themselves fiddling with PPAs or compiling drivers from the Nvidia website to get the most out of the game.

"Shadow of Mordor is a great game that you don't need to be a Lord of the Rings fan to enjoy."

Shadow of Mordor is a great game which you don't need to be a *Lord of the Rings* fan to enjoy. It has plenty of innovative features that it pulls off naturally without them coming off as gimmicks, and while the story won't win a Pulitzer prize any time soon, it does its job in tying together the incredibly fun gameplay.

Mordor looks great, even on mid-range graphics cards, though to get insane high-res textures you'll need some very pricy hardware.

DATA

Web
www.shadowofmordor.com
Developer
Monolith Productions, Feral Interactive (Linux port)
Price
£29.99

LINUX VOICE VERDICT

A very fun open world game that provides countless hours of gameplay and plenty of replay value.

★★★★☆

When sysadmins ruled the earth

Ben Everard wishes someone would put a sysadmin in charge of the trains and get five nines of them on time.

Science fiction, at its best, is an exploration of the very nature of humanity. It's not about the science, but by placing humans in new and unusual circumstances. We get to imagine what we would do in a similar setting – the dissonance provides the central conflict. Perhaps the most unusual circumstance possible is a group of geeks inadvertently ending up running the world. *When Sysadmins Ruled The Earth* explores the relationships and power struggles between the people running various datacenters as they debate global leadership.

Google's data centre queen vies with the sysadmin of a bank and hacktivists to decide the fate of the world using the ultimate democracy tool of the future: Usenet. Through online discussion they try to chart a course for humanity. We can't say much more than this without giving away the story, but you should get a good of what this story is about from the title.

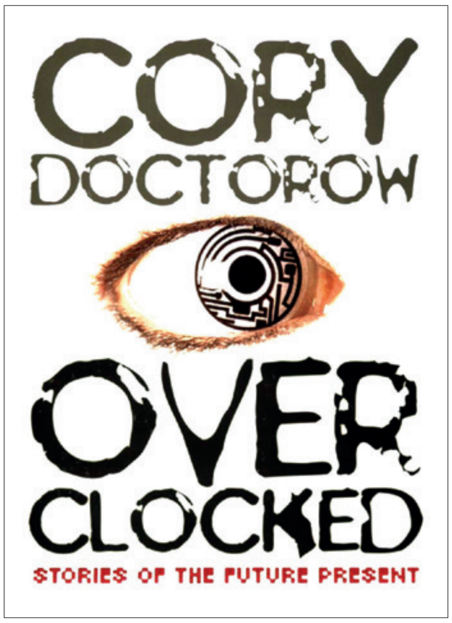
We're cheating a bit by including this story in the book review section, since it's not a book by itself but a short story. You can get it via the anthology *Overclocked: Stories of the Future Present*, or it's also available as a standalone story in various digital forms from Cory's website, **http://craphound.com/overclocked/download**.

LINUX VOICE VERDICT

Author Cory Doctorow
Publisher Thunder's Mouth Press
Price Free
ISBN 978-1560259817

A fast-paced tale that manages to be both sad and uplifting, and leaves us wondering whether the principals of cyberspace should be applied to the real world.

★★★★★



If you like *When Sysadmins Ruled The Earth*, there are five more stories in this anthology.

The Linux Command Line

Ben Everard could get to work quicker if his bike had a command line interface and not pesky handlebars.

When you open a terminal and start to type commands, you're using the most powerful computer control system ever created. I don't actually have any proof of that statement, but I'm pretty sure it's true. Nothing else has the sheer range of features that a good Linux install does, and if you're connected to a deep set of repositories, you can always get any more software you need from the same command line.

There is, however, one flaw in the command line system: it can be hard to learn how to use it. Sure, you'll pick up bits as you go along from websites and magazines, but unless you make a concerted effort to systematically learn, you'll never fully unlock the system's power.

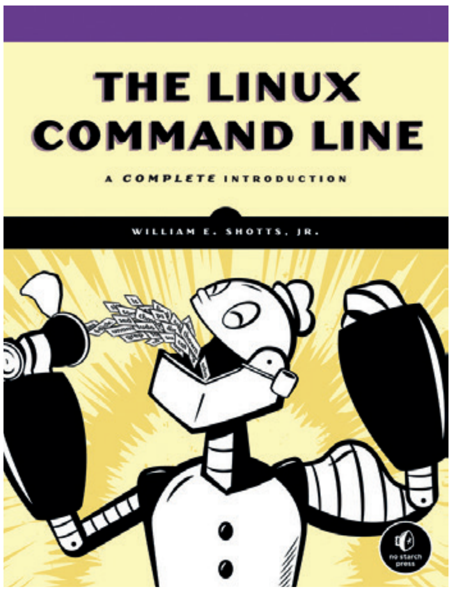
There are many, many books about the Linux command line. In fact, it's such a complex interface that no one book could completely cover it. *The Linux Command Line* is ideally suited to people who have a little experience of Linux and are looking to make the jump from graphical use to command line use. It starts with the complete basics of opening a shell and navigating through the filesystem, but if you're already familiar with this, you can easily skim through the first few chapters. By the end of the book, the reader should be familiar with shell scripting and other more advanced uses of the command line.

LINUX VOICE VERDICT

Author William Shotts
Publisher No Starch Press
Price Free or £26.50
ISBN 978-1593273897

If you struggle with the command line, get this book. It will improve your life.

★★★★★



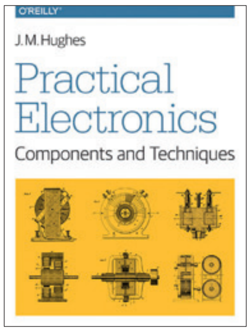
The best book on Linux you don't have to buy (and better than most that you can buy too).

Practical Electronics

Graham Morrison finds a guide for building the ultimate synthesizer.

If there was one thing we wish they'd taught us at school instead of Home Economics, it was some simple electronics theory. We've been playing with circuit boards and components ever since, but our lack of any educational rigour and perspective makes it much harder than it needs to be. Unlike computers, where trial and error seldom leads to smoke, playing with electronics is less intuitive, more costly and slightly more dangerous.

This is why *Practical Electronics* is one of the best books we've read for a while. It fits our collective ignorance perfectly, never patronising or over explaining a subject, and covering everything you need to know to start hacking circuits like you hack Python code. We really like the early parts on background theory, especially as it does away with the water flow/pressure analogy that's getting long in the tooth, and we much preferred the allusion to the real physics of neutrons, electronics and protons. Ultimately, though, it's the size and comprehensive nature of this book that makes it effective, and we have no hesitation recommending this to anyone with a little theory and a soldering iron.


Knowing very little about electronics shouldn't stop you building stuff.

LINUX VOICE VERDICT

Author JM Hughes
Publisher O'Reilly
Price £26.50
ISBN 978-1449373078

Not for experts, but for anyone else this is a comprehensive read and great value.

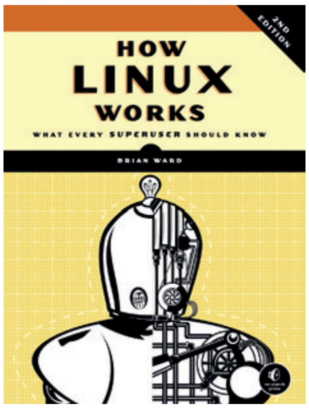
★★★★☆

How Linux Works 2nd Edition

Graham Morrison finally learns the difference between su and sudo.

If there was one book that did more than anything to help us get into Linux, it was *Linux in a Nutshell*, by O'Reilly. In the 1990s, it offered a rare insight into how the whole thing held together, and how you could harness the various parts of Linux to start doing practical things with your computer. The original *Nutshell* is still a classic, but it does read a little like a man page, which is why this is such a good alternative. It's absolutely crammed full of information, building up from the basics of layers and the kernel, through the most important commands, devices, storage and networking.

What's so good about *How Linux Works* is that while it's undoubtedly dense, it's far more readable than the average manual, and there are plenty of examples to help put the information in context. Even those with no prior Linux, if they're patient, will become proficient sysadmins. It's also huge (366 pages) and sensibly priced, making it our go-to recommendation.


The only things missing are robots.

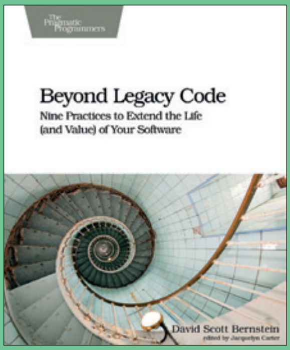
LINUX VOICE VERDICT

Author Brian Ward
Publisher No Starch Press
Price $39.95
ISBN 978-1593275679

It's like having a friendly and experienced sysadmin show you the ropes.

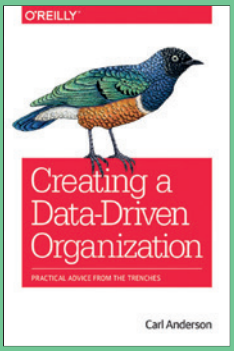
★★★★★

ALSO RELEASED...


The cover image made us initially think of *Bash*.

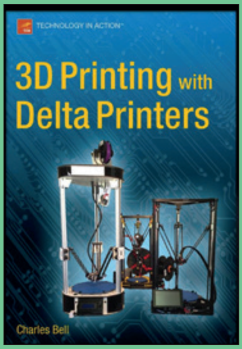
Beyond Legacy Code

Dealing with old code is much more common than starting a fresh project, so it's surprising that there aren't more titles that deal with this thorny subject. The book promises to provide nine technical practices that will help any project head off the problems of old code.


We chose this book purely for the nice bird on the cover.

Creating a Data-Driven Organization

We like to think of ourselves as a data-driven organisation here at Linux Voice. Each month we plough through a *LibreOffice* spreadsheet of stories, sales and re-subscriber rates, print pretty charts and discuss it all in Hangouts. What we really need is a C-level chief data.


What happens when the machines build more machines?

3D Printing with Delta Printers

We all think 3D printing is fascinating, even if it's still slightly costly. But the technology is moving all the time, and it's worth investing some time to understand how things are changing and how things might improve. Delta printing is one such innovation.

LINUX VOICE

SEARCH ENGINES

GROUP TEST

Google may rule the web, but alternatives are worth checking out. **Mike Saunders** explores them and shows you some tricks and tips.

On test

Google

URL www.google.com
COMPANY Google Inc.
LAUNCHED September 1997
The best known, but with growing concerns about privacy.

Bing

URL www.bing.com
COMPANY Microsoft
LAUNCHED June 2009
Microsoft is snapping at Google's heels, with almost 20% market share in the US.

DuckDuckGo

URL www.duckduckgo.com
COMPANY DuckDuckGo Inc.
LAUNCHED September 2008
"The search engine that doesn't track you." But how good are the actual results?

YaCy

URL www.yacy.net/en
COMPANY YaCy developers
LAUNCHED November 2011 (v1.0)
Built on a peer-to-peer network, this is a free and distributed search engine.

Wolfram Alpha

URL www.wolframalpha.com
COMPANY Wolfram Alpha LLC
LAUNCHED May 2009
More than a search engine, this is a computational knowledge engine for answering scientific queries.

Ixquick

URL www.ixquick.com
COMPANY Surfboard Holdings BV
LAUNCHED March 2005
A metasearch engine that aims for maximum privacy.

Search engines

In a departure from the norm, we're not looking at Linux-specific software this month, although many of the search engines on test here use Linux in one way or another. But we feel that it's a subject that's well worth investigating, as virtually every Linux user makes use of a search engine, usually multiple times every day. Google has dominated this market for so long, but times are changing, and alternatives are starting to win attention.

But why are people leaving Google? Doesn't it simply provide the best results out there? Well, historically, that has been the case. For many years, Google provided hands-down the best results – and in a clear and fast way. But the situation is rather different today. Google users often find themselves in the "filter bubble", whereby Google shows results very much tailored to your previous search history and websites you've visited. This can be helpful, but it can make it hard to find alternative opinions or lesser-known sites, so you end up with a lot of confirmation bias.

Similarly, due to Google's domination of search, many companies are doing everything possible to get high up in the list of results. So you'll often click links that contain nothing of use except adverts, or simply ripped-off content from another site that someone else is trying to monetise. This isn't so easy for Google to fix, but it's frustrating nonetheless.

Then you have privacy issues. By and large, Google has been one of the better players in the industry in this regard, and has fought against mass surveillance efforts by western governments. Nonetheless: the company makes money from advertising, and stores a huge amount of data from your search habits, browsing history, and email content (if you use Gmail). So if privacy is your number one concern when searching, you'll want to try other options.

"Google stores a huge amount of data from your search habits."

Different name, same engine

You'll notice that a few famous names in search, such as Yahoo and Ask, aren't included in this group test. The reason for this is simple: they've outsourced their web search facilities to other companies. Yahoo, for instance, now gets its results from Bing, so apart from its website design there's not a lot to talk about. Ask used to have its own search engine, but now gets results from a third-party provider.

Similarly, some readers might be asking why we haven't covered **www.startpage.com**. This is a useful privacy-oriented engine that claims to not store any of your data, but ultimately it's just a front-end to Google Search, so the quality of the results is the same.

Yandex – the next big contender?

Russian search giant is making moves into the English-language market.

What Google is to the huge markets of Europe and the United States of America, Yandex is to Russia. It's by far the biggest search engine in that country, with over 60% of the market, and it also hosts additional services such as email, maps and videos. Yandex has even produced its own web browser, based on Google's *Chromium*.

Although Yandex has been in operation since 1997, it has only recently started targeting the English-language market with a search engine at **www.yandex.com**. The design is clean and tidy, and along with text search the site also offers image searches and a rather effective translation system.

Intriguingly, Yandex also includes buttons for Bing and Google searches at the bottom of its results – in case you're not happy with Yandex's. That's very nice of them, but it's strange for a fledgling company (in the English market at least) to point people at its competitors. Still, Yandex's own results are fairly good, and cached versions are available for many pages, which is useful if the original site is down or has been moved to another location.

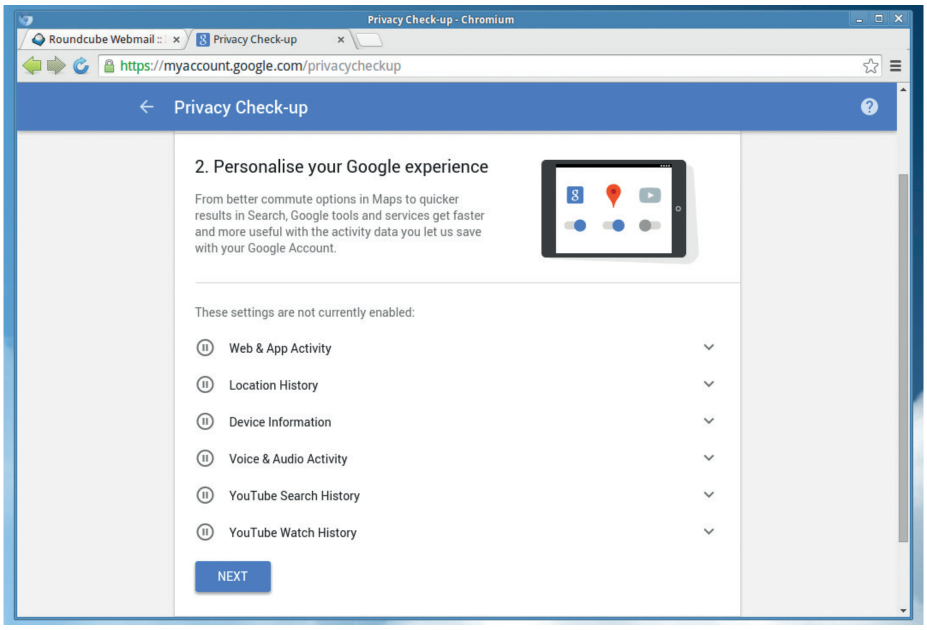
Google

Close to the "creepy line".

In October 2010, Eric Schmidt, the then-CEO of Google, explained his company's goals thusly: "Google policy is to get right up to the creepy line and not cross it." In other words, because of the vast amount of information that the company collects – from your web searches, image searches, calendar, email and Google+ posts – Google should be able to provide you with information so specific and useful, it becomes almost creepy in its accuracy. He added: "We don't need you to type at all. We know where you are. We know where you've been. We can more or less know what you're thinking about".

Now, plenty of people found this useful at the time; if you're willingly giving information to Google, there's nothing wrong with the company parsing it, poking around in it, and trying to make your life better with it, right? Unfortunately, following the Snowden revelations of mass government surveillance, and the use of government force to secretly extract information from companies in the name of "terrorism prevention", many people have become a lot more cynical. Google – along with other companies – has joined campaigns to reduce the levels of mass surveillance, and wants to make it very clear when it is forced to hand data over to the spooks.

One problem Google has is the sheer number of services that it operates. So we'll give the company some kudos for creating a "Privacy Check-up" page at **https://myaccount.google.com/privacycheckup**, which provides an at-a-glance list of things you may want to turn off (web search history, location history, YouTube video history and so forth). It also lets you disable adverts based on your interests – or more specifically, what Google thinks your interests are.


Privacy Check-up - Chromium
Roundcube Webmail ::
Privacy Check-up
https://myaccount.google.com/privacycheckup
Privacy Check-up
NEXT

2. Personalise your Google experience

From better commute options in Maps to quicker results in Search, Google tools and services get faster and more useful with the activity data you let us save with your Google Account.

These settings are not currently enabled:

- Web & App Activity
- Location History
- Device Information
- Voice & Audio Activity
- YouTube Search History
- YouTube Watch History

Google's data-harvesting capabilities sometimes give us the creeps, but the company makes it easy to opt out of them.

Bubble bobble

We have two main beefs with modern Google, the first being the "filter bubble". Just recently, as the Linux Voice team was talking on our IRC channel, we saw this in action: one of us noted that a certain search put Linux Voice right at the top of the results, but on another team member's machine, it was much further down.

Now, it is possible to prevent this from happening, but it's rather fiddly. You have to go to Google's front page for your language (not a search result page), then click Settings > History. From there you disable "Signed-Out Search Activity", which isn't a particularly descriptive term anyway. From here, the bubble shouldn't affect your results.

The second annoyance we have is with unrelated results that don't contain the words we specify, even when we provide them in quote marks to search for an exact phrase. You need to click Search Tools > All Results > Verbatim to actually retrieve results for what you're looking for. This becomes very tedious after a while.

On the whole, Google still offers the most comprehensive set of results when you fiddle with its default settings, and the integration between its services is excellent.

VERDICT
Still the best for results, but by default it's lacking in terms of privacy and power-user options.
★★★★☆

Wolfram Alpha

More than search – it's a computational knowledge engine.

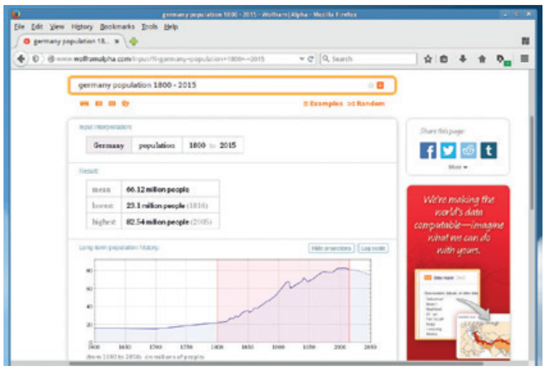
Wolfram Alpha isn't a direct competitor to most of the search engines on test here, in that you wouldn't use it to find a decent curry house in a town you're going to visit. Instead, it's all about performing comparisons between huge sets of data. For instance, if you search Google for "tallest buildings in Japan", Google will try to find a relevant page for these words – in this case, some pages on Wikipedia.

Wolfram Alpha, in contrast, will try to make more sense of the words and look at its own data sources. Its 10,000 CPUs chug away to process statistics stored in its database, and present you with useful results without having to direct you to another page. It can also perform calculations itself, so you can use a query like "days until Christmas 2017", and Wolfram Alpha will show you the result and related queries (number of weeks, weekdays etc).

The site is also excellent for performing comparisons between things. Enter "Paraguay vs Uruguay", for example, and you'll receive a vast amount of data comparing the two countries' geographies, economies, demographics and so forth. You can narrow down to specifics with "Paraguay vs Uruguay GDP" and get more detailed information, often accompanied with graphs generated on the fly. And right at the bottom, you can see the sources for this data: the UN, the WHO, World Bank etc.

Wolfram Alpha can do incredible work, but it can be quite tricky to use properly. Fortunately, the developers have created a broad set of examples showing how to harness all the maths, statistics, history, nutrition and other data that the website provides: **www.wolframalpha.com/examples**.

"Wolfram Alpha is all about performing comparisons between huge sets of data."


Wolfram Alpha taps data from a vast number of sources and is great for making comparisons or charts.

VERDICT
Takes time to master, but fantastic for research and investigation.
★★★★☆

YaCy

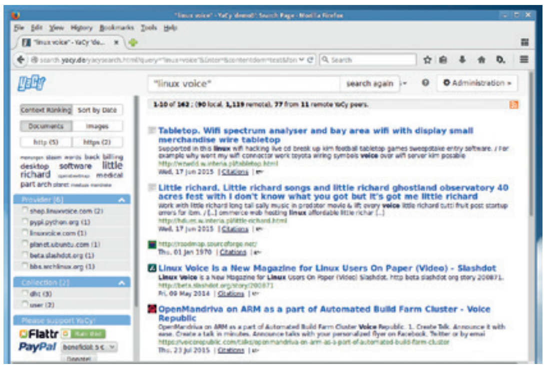
"Web search by the people, for the people."

While Google, Bing and co. are centralised search engines – in other words, they're operated by single companies from a bunch of datacentres – YaCy is a distributed engine, which relies on contributions from users around the world. It's decentralised, so anyone can contribute computing power to it by downloading the software and hooking up with the other 600+ nodes that are crawling the web and generating indexes of content. YaCy is written in Java with no other dependencies, so it's fairly quick and easy to get it up and running: just run the program and access it via the web-based interface at **http://localhost:8090**.

YaCy can be used to provide search facilities for a local network such as an intranet, but here we'll focus on its capabilities as a web-wide search engine. Currently it receives around 130,000 search queries every day, and if you want to try it yourself without booting up the software, a demo search portal is available at **http://search.yacy.de**. And here's the downer: the search results are rather bad. "Linux Voice" brings up several completely irrelevant results, followed by a Slashdot story about the magazine, but our actual main website doesn't appear until the bottom of the page.

Potential

Now, the YaCy community is rather small at this stage and doesn't have the vast resources of Google and Microsoft, so this is to be expected. You wouldn't want to use YaCy for your daily queries, but it's a noble project nonetheless, especially as it promises to be resistant against widespread censorship and centralised advertising campaigns. YaCy could be a major player in search a few years down the line, if more and more people contribute their computing resources to it, but for now we consider it more of an intriguing tech demo rather than something for practical day-to-day usage.


"linux voice"

1-10 of 162 : (90 local, 1,119 remote), 77 from 11 remote YaCy peers.

Tabletop. Wifi spectrum analyser and bay area wifi with display small merchandise wire tabletop
Wed, 17 Jun 2015 | Citations

Little richard. Little richard songs and little richard ghostland observatory 40 acres fest with I don't know what you got but it's got me little richard
Wed, 17 Jun 2015 | Citations

http://roadmap.sourceforge.net/
Thu, 01 Jan 1970 | Citations

Linux Voice is a New Magazine for Linux Users On Paper (Video) - Slashdot
Linux Voice is a New Magazine for Linux Users On Paper (Video) Slashdot.
Fri, 09 May 2014 | Citations

OpenMandriva on ARM as a part of Automated Build Farm Cluster - Voice Republic
Thu, 23 Jul 2015 | Citations

Provider (6)
shop.linuxvoice.com (2)
pypi.python.org (1)
linuxvoice.com (1)
planet.ubuntu.com (1)
beta.slashdot.org (1)
bbs.archlinux.org (1)

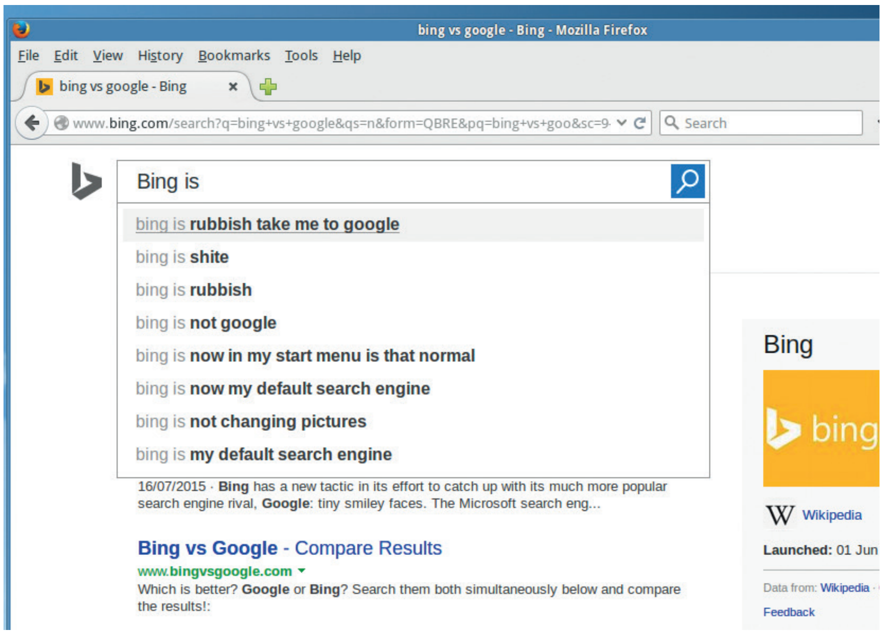
Collection (2)
dht (3)
user (2)

YaCy's results page is a bit cluttered, but the "provider" links on the left can come in useful.

VERDICT
Plenty of promising tech in here, but the results leave a lot to be desired.
★★½☆☆

Bing

Microsoft has its fingers in many pies.


bing vs google - Bing - Mozilla Firefox
File Edit View History Bookmarks Tools Help
bing vs google - Bing
www.bing.com/search?q=bing+vs+google&qs=n&form=QBRE&pq=bing+vs+goo&sc=9-
Search
Bing is
bing is rubbish take me to google
bing is shite
bing is rubbish
bing is not google
bing is now in my start menu is that normal
bing is now my default search engine
bing is not changing pictures
bing is my default search engine
Bing
bing
Wikipedia
Launched: 01 Jun
Data from: Wikipedia
Feedback

16/07/2015 · Bing has a new tactic in its effort to catch up with its much more popular search engine rival, Google: tiny smiley faces. The Microsoft search eng...

Bing vs Google - Compare Results
www.bingvsgoogle.com
Which is better? Google or Bing? Search them both simultaneously below and compare the results!:

It's very clear that Microsoft hasn't tried to game the suggestion box when you type "Bing is"...

Bing may be a relatively new player in the search engine market, having been launched in June 2009, but Microsoft has been involved in this field for many years, starting with MSN Search in 1998. At the start, Bing distinguished itself from Google by having a fancy front page (with large background images showing places around the world) and a slightly cheesy slogan: "The sound of found". Microsoft marketed Bing as a "decision engine", claiming it would provide more relevant results than the competition along with tools to narrow down searches.

Today, Bing has around 20% market share in the US – a respectable figure, although it must be noted that *Internet Explorer*, while not as dominant as it was, is still used on a vast number of Windows installations and has Bing set as the default search engine. Bing also offers image searches and rather good maps.

By default, Bing keeps a history of your web searches, even if you're not signed in; it's possible to disable this by clicking the cog icon in the top-right. One useful feature that Bing lacks and which Google has, however, is the ability to narrow down results to a certain period of time. You can narrow down by language and location, but not to specific dates. Bing did have this facility a few years ago, and it's not clear why Microsoft removed it, but it puts Bing way behind Google for certain types of searches.

"Bing is for doing"

Results-wise, though, Bing is catching up with Google. We tried dozens of searches across all sorts of categories – food, travel, technology etc – and rarely did we see any great differences.

Bing has various advanced features such as the ability to search for pages on a specific URL with "site:", or show results that only match a specific filename extension. Bing also caches many pages, which comes in useful.

For most users, Bing offers everything available in Google, and it even looks remarkably similar. We'll give Microsoft credit for producing good results, but while the company (or at least parts of it) is still hostile to Linux and FOSS, we can't recommend it in good conscience just yet. And while privacy is a hot topic, we would be happier if Bing let new users know that it stores their search history with a big pop-up, along with information on how to disable this "feature".

VERDICT
Approaching Google for results quality, but still with issues about privacy.
★★★½☆

The future of search

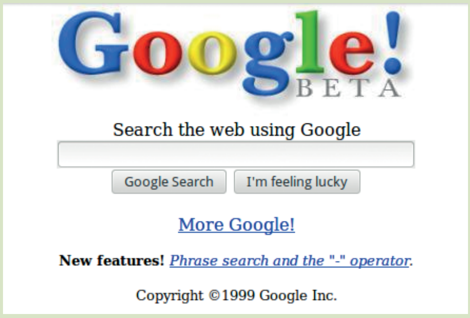
Out with the reactive, in with the proactive.

While the traditional search engine won't go away any time soon, Google, Microsoft and others want to feed you with information you ostensibly need – without you having to explicitly look for it. To some extent this is already available today with Google Now and similar services, which pull together all your contact, calendar and location data, and try to present you with relevant information on the move.

As an example: Google claims that 30% of mobile searches are restaurant-related. If Google knows where you are from GPS coordinates, who you're meeting from your calendar entries (business or personal), and where you've been before, it can suggest places to eat without you having to look them up. And this is a boon for advertisers – on a dull, rainy evening when you have nothing to do, they can suggest events and things to see and do nearby.

Rage against the dying of the search!

But, of course, this is all at the expense of privacy. Some people will love this level of integration between different services, whereas others will run a million miles in the opposite direction. Many of us will just want to use search engines as they always were, anonymously and outside of a filter bubble, and those people will find challenges up ahead. Websites are constantly cropping up supposedly offering useful data (actually copied from elsewhere) and loaded with annoying adverts, and it's a constant battle for Google *et al* to sort the wheat from the chaff.


Google! BETA
Search the web using Google
Google Search
I'm feeling lucky
More Google!
New features! Phrase search and the "-" operator.
Copyright ©1999 Google Inc.

Search companies are trying to be all things to all people; here's hoping they don't forget the basics.

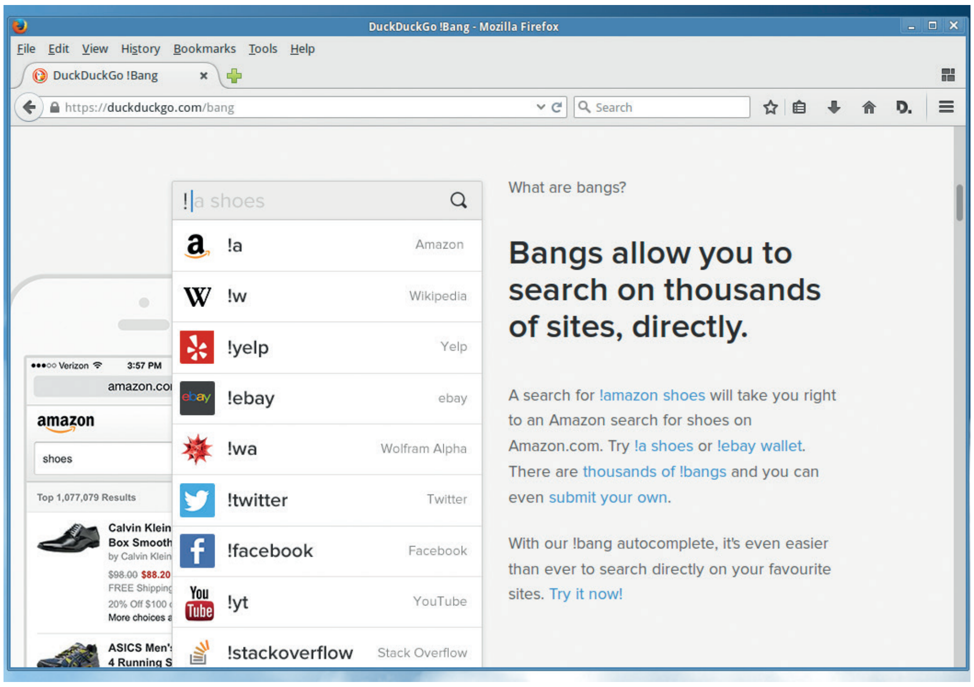
DuckDuckGo vs Ixquick

Two privacy-oriented engines go head-to-head.

DuckDuckGo is the better known of these two engines, and proudly proclaims that it helps you to "take back your privacy", because it doesn't track you. However, that doesn't mean that the search engine doesn't store any data – indeed it does, but not in a "personally identifiable way". So whereas Google stores all search terms along with the IP addresses from which they were sent, DuckDuckGo only stores the former and uses it to improve results. Similarly, DuckDuckGo doesn't save any cookies by default, and only if you change the default settings.

Out of the box, DuckDuckGo's results page is one of those infinitely scrolling beasts, which rather annoys us. Still, you can turn this off by clicking the three-line "hamburger" icon in the top-right, which also lets you disable safe search, narrow results down to a particular region, and even disable advertising. DuckDuckGo just asks that you spread the word about the website if you do the latter.

DuckDuckGo offers a huge range of "!bang" features for instantly searching other sites – so you can do "!g Linux Voice" to search for Linux Voice on Google, "!r" for Reddit, and so forth. There are over 6,000 bangs currently available, and the DuckDuckGo team strives to collaborate with the open source community via **http://duckduckhack.com**.


DuckDuckGo !Bang - Mozilla Firefox
File Edit View History Bookmarks Tools Help
DuckDuckGo !Bang
https://duckduckgo.com/bang
Search
!a shoes
!a Amazon
!w Wikipedia
!yelp Yelp
!ebay ebay
!wa Wolfram Alpha
!twitter Twitter
!facebook Facebook
!yt YouTube
!stackoverflow Stack Overflow

What are bangs?

Bangs allow you to search on thousands of sites, directly.

A search for !amazon shoes will take you right to an Amazon search for shoes on Amazon.com. Try !a shoes or !ebay wallet. There are thousands of !bangs and you can even submit your own.

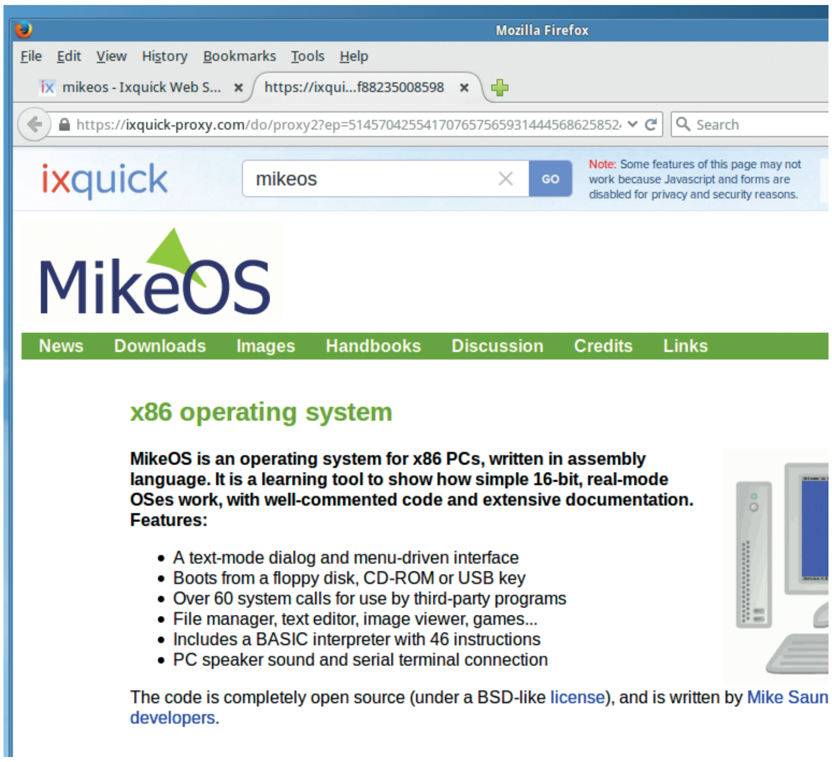
With our !bang autocomplete, it's even easier than ever to search directly on your favourite sites. Try it now!

DuckDuckGo's "bangs" are a killer feature of the engine, and can search via 6,000 other websites.

The Ixquick alternative

Ixquick is a similar service that doesn't record your IP address with your searches, and only uses a cookie to store your preferences. One nice touch is its ability to auto-generate a URL with your preferences encoded in the string, so if you don't even want the anonymous cookie, you can still keep your search settings.

Ixquick has a useful proxy server, with links provided next to each result, so you can view a page indirectly using Ixquick's own servers. This isn't a foolproof way to anonymously access a website, but it's another little bit of privacy and it all adds up. Ixquick isn't a bad server *per se*; it's just that DuckDuckGo has more power user features thanks to the bangs, and a more proactive approach to FOSS.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
mikeos - Ixquick Web S...
https://ixqui...f88235008598
https://ixquick-proxy.com/do/proxy2?ep=5145704255417076575659314445686258524
Search
ixquick
mikeos
GO
MikeOS
News Downloads Images Handbooks Discussion Credits Links

Note: Some features of this page may not work because Javascript and forms are disabled for privacy and security reasons.

x86 operating system

MikeOS is an operating system for x86 PCs, written in assembly language. It is a learning tool to show how simple 16-bit, real-mode OSes work, with well-commented code and extensive documentation. Features:

- A text-mode dialog and menu-driven interface
- Boots from a floppy disk, CD-ROM or USB key
- Over 60 system calls for use by third-party programs
- File manager, text editor, image viewer, games...
- Includes a BASIC interpreter with 46 instructions
- PC speaker sound and serial terminal connection

The code is completely open source (under a BSD-like license), and is written by Mike Saun developers.

Ixquick makes it easy to visit pages via a proxy, which adds another layer of anonymity.

VERDICT

DUCKDUCKGO	IXQUICK
Rapidly improving, and becoming our favourite.	Some good privacy features, but results aren't fantastic.
★★★★☆	★★★☆☆

OUR VERDICT

Search engines

Many of us in the FOSS world have been keeping one eye on DuckDuckGo in the last few years, but we've usually come away slightly disappointed due to the low quality of the search results. Well, those days are gone now. Google is still the king when it comes to the most accurate and relevant results, but Bing and DuckDuckGo are very much providing competition.

DuckDuckGo is now usable as a day-to-day engine: the quality of its results is improving with every month, it's very easy to integrate it with *Firefox* or *Chromium*, and its privacy policy is respectable. Of course, DuckDuckGo is still based in the USA, which poses its own set of privacy issues (we don't know what happens when the government comes knocking on its door), but assuming its privacy policy is 100% legitimate, your searches won't be leaked.

Additionally, DuckDuckGo's interaction with its community, and especially the FOSS world, is worthy of a thumbs-up. A lot of code has been uploaded to **https://github.com/duckduckgo**, for instance, and the whole engine is run on a mixture of Ubuntu and FreeBSD, with *Nginx* doing the web serving. DuckDuckGo has also handled the extra load placed on it in recent years – its traffic has shot up 600% since the Snowden leaks came to light.

So DuckDuckGo is the winner this month, and we recommend that everyone gives it a try. Ixquick has some privacy benefits too, and Bing is worth a shot if you just want to escape from Google's mighty talons for a while. But for the overall best mixture of search results, privacy and advanced features, DuckDuckGo takes the top spot. And in all honesty, that's something we never thought we'd say six months ago – the engine has come on in leaps and bounds, so if you gave it a go before but weren't satisfied, give it one more shot. LV

"The quality of DuckDuckGo's results is improving with every month."

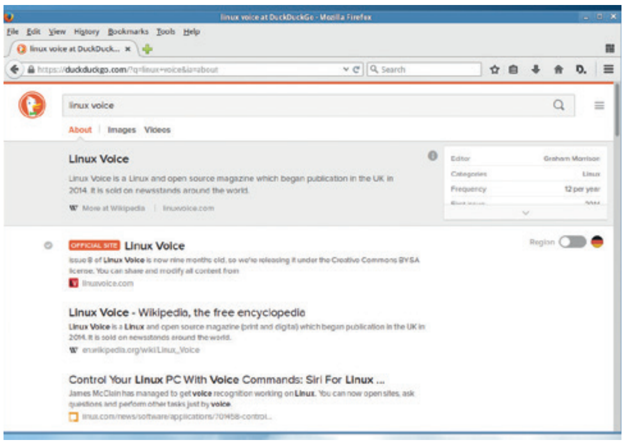
Privacy-centric email

Google and Bing both have email services to complement their search engines, and rumours abound that DuckDuckGo will eventually launch an email service as well. The premise here is simple: your email should only be readable by you. This means that the company providing the email service can't poke around inside your messages and deliver relevant advertising, so it's a thorny topic when it comes to free providers. But a number of startups are offering end-to-end encrypted mail for free, with paid add-ons.

Protonmail (**http://protonmail.com**), for instance, is a Swiss-based provider that uses a dual password system – one to log in, and one to decrypt your mailbox. This means that you can use it as webmail without having to set up a local email client and fiddle around with GPG keys, but it also means that you have to trust that you're being served the right JavaScript each time you access your mailbox (eg it hasn't been intercepted and modified by a third party). Still, it's a start. Some other services worth looking at include:

- CounterMail: **https://countermail.com**.
- Neomailbox: **www.neomailbox.com**.
- Mailpile: **www.mailpile.is**.

Many of these are relatively new and only open to invitation while they build up their infrastructure, but if you'd like us to cover them in a future group test, just drop us a line.


linux voice at DuckDuckGo - Mozilla Firefox

https://duckduckgo.com/?q=linux+voice&ia=about

linux voice

About | Images Videos

Linux Voice

Linux Voice is a Linux and open source magazine which began publication in the UK in 2014. It is sold on newsstands around the world.

More at Wikipedia | linuxvoice.com

Editor: Graham Morrison
Categories: Linux
Frequency: 12 per year

OFFICIAL SITE Linux Voice

Issue 8 of Linux Voice is now nine months old, so we're releasing it under the Creative Commons BY SA license. You can share and modify all content from

linuxvoice.com

Linux Voice - Wikipedia, the free encyclopedia

Linux Voice is a Linux and open source magazine (print and digital) which began publication in the UK in 2014. It is sold on newsstands around the world.

en.wikipedia.org/wiki/Linux_Voice

Control Your Linux PC With Voice Commands: Siri For Linux ...

James McClain has managed to get voice recognition working on Linux. You can now open sites, ask questions and perform other tasks just by voice.

linux.com/news/software/applications/70458-control...

DuckDuckGo has it all: good results, privacy, and user-contributed power searches.

1st DuckDuckGo
Alexa Rank 520

www.duckduckgo.com
Rapidly improving, with lots of hidden feature gems and a good privacy policy.

2nd Wolfram Alpha
Alexa Rank 1,932

www.wolframalpha.com
If you're doing any kind of research and need to compare statistics, this should be your first stop.

3rd Google
Alexa Rank 1

www.google.com
Excellent results, but becoming frustrating to use, and tries to put a filter bubble around you.

4th Bing
Alexa Rank 27

www.bing.com
Our old nemesis Microsoft is catching up with Google, but also has issues with privacy.

5th Ixquick
Alexa Rank 9,858

www.ixquick.com
A mixed bag when it comes to results, but has some decent features for privacy and anonymity.

6th Yacy
Alexa Rank N/A

www.yacy.net
Not usable on a day-to-day basis, but could revolutionise web searches one day.

SUBSCRIBE

shop.linuxvoice.com

Introducing Linux Voice, the magazine that:

- Gives 50% of its profits back to Free Software
- Licenses its content CC-BY-SA within 9 months

12-month subs prices

UK – £55
Europe – £85
US/Canada – £95
ROW – £99

7-month subs prices

UK – £38
Europe – £53
US/Canada – £57
ROW – £60

DIGITAL SUBSCRIPTION ONLY £38

Get 114 pages of tutorials, features, interviews and reviews every month

Access our rapidly growing back-issues archive – all DRM-free and ready to download

Save money on the shop price and get each issue delivered to your door

Payment is in Pounds Sterling. 12-month subscribers will receive 12 issues of Linux Voice a year. 7-month subscribers will receive 7 issue of Linux Voice. If you are dissatisfied in any way you can write to us to cancel your subscription at subscriptions@linuxvoice.com and we will refund you for all unmailed issues.

NEXT MONTH IN LINUX VOICE

ON SALE THURSDAY 24 SEPTEMBER

THE BEST LINUX DESKTOP

Discover the finest graphical glitz available right now for your humble Linux machine.

EVEN MORE AWESOME!

OPEN RIGHTS GROUP

Inside ORG

Meet the people who are trying to persuade the UK government not to sell our digital rights to the highest bidder. Keep up the good work, team.

Blender

Model collisions in 3D, use crazily hard maths and impress everyone you know by mastering the most powerful graphics known to Free Software.

Two-factor authentication

Keep your secrets safe with a password and a key, and never worry about leaving confidential files on the bus ever again.

LINUX VOICE IS BROUGHT TO YOU BY

Editor Graham Morrison
graham@linuxvoice.com
Deputy editor Andrew Gregory
andrew@linuxvoice.com
Technical editor Ben Everard
ben@linuxvoice.com
Editor at large Mike Saunders
mike@linuxvoice.com
Creative director Stacey Black
stacey@linuxvoice.com

Editorial consultant Nick Veitch
nick@linuxvoice.com

All code printed in this magazine is licensed under the GNU GPLv3

Printed in the UK by
Acorn Web Offset Ltd

Disclaimer We accept no liability for any loss of data or damage to your hardware through the use of advice in this magazine. Experiment with Linux at your own risk!
Distributed by Marketforce (UK) Ltd, Blue Fin Building, 110 Southwark Street, London, SE1 0SU
Tel: +44 (0) 20 3148 3300

Circulation Marketing by Intermedia Brand Marketing Ltd, registered office North Quay House, Sutton Harbour, Plymouth PL4 0RA
Tel: 01737 852166

Copyright Linux is a trademark of Linus Torvalds, and is used with permission. Anything in this magazine may not be reproduced without permission of the editor, until May 2016 when all content (including our images) is re-licensed CC-BY-SA.
©Linux Voice Ltd 2015
ISSN 2054-3778

Subscribe: **shop.linuxvoice.com**
subscriptions@linuxvoice.com

Valentine Sinitsyn develops high-loaded services and teaches students completely unrelated subjects. He also has a KDE developer account that he's never really used.

CORE TECHNOLOGY

Prise the back off Linux and find out what really makes it tick.

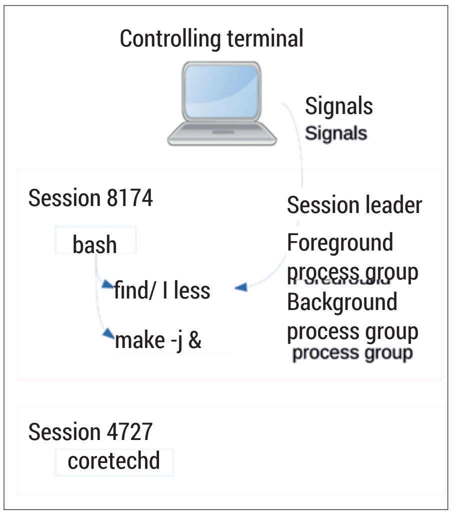
Daemons

Reveal the dark world of things that usually slip unnoticed behind luxury GUIs of Linux distros.

The first scary thing that Windows converts usually learn about Linux is that it doesn't have disk C:. The next one is arguably that it has that mystical daemon thing. A typical example is **crond**: it sleeps most of the time, but wakes up once a minute to execute periodic tasks. Why on the Earth would you use a haunted OS?

In fact, Linux daemons are just long-living background processes that follow a set of conventions. By convention, their names end with a letter "d" to distinguish them from regular processes. Just a decade ago, you were mostly on your own if you wanted to implement daemons properly (although there were helper libraries available). Now, with *Systemd*, *Upstart*, and friends, boilerplate code is reduced to a minimum.

"Daemons are typically written in systems languages, with C and C++ being the most common choices."



Roughly, here's the relationship between process groups, sessions and several other terms you'll encounter.

Daemons are typically written in systems programming languages, with C and C++ being the most common choices. However, nothing prevents you from using Python or even more exotic languages like Erlang.

Daemons in a nutshell

When you execute a command, the shell normally forks a new foreground process (see LV018). It inherits many of the shell's resources, for instance, the controlling terminal. This can be a real device that the user logged from, or a pseudo terminal created by **xterm** or **ssh**. Everything you type goes to the process; everything it writes is sent to the terminal. If you press Ctrl+C, the process receives the signal (**SIGINT**) and terminates. When the controlling terminal is closed, the process receives **SIGHUP** which also terminates it, by default.

Daemons are a bit different. They break the ties with whatever gave birth to them to be as self-contained as possible, because the daemon can last much longer than its parent. So it shouldn't be run on a filesystem that the user may want to unmount, or keep open file descriptors for the same reason. Neither should it send anything to the terminal as the user can log out shortly after starting the daemon. Signals are also handled differently; for instance, **SIGHUP** is often used to reload configuration files.

Steps for proper daemonisation are detailed well in *Advanced Programming in the Unix Environment* (APUE) by W Stevens and S Rago. Another good reference is the **daemon(7)** man page. Note that the steps are slightly different in both resources. Again, a daemon is merely a set of conventions, and there's more than one way to follow them.

Daemons need a clean predefined environment to run, so usually the first step is to create one. First, **umask** must be reset. This ensures that the files the daemon creates will have the exactly those permissions it expects. Signal handlers are also normally reset to their default state. Selected signals like **SIGHUP** are assigned custom handlers later.

The first "real" step is the **fork(2)** syscall, which serves two purposes. First, the parent process can now exit, tricking the shell into thinking that the command has completed. Then, it ensures the child won't be a process group leader. Process groups are subdivisions inside a session used mainly for signal delivery and job control. We won't discuss them here, but you can find all the details in APUE.

"Demoting" the child is essential for the later step. The daemon must lose its controlling terminal. To do that, it creates a new session with no controlling terminal and becomes its session leader. The **setsid(2)** syscall does exactly this, but succeeds only if the process isn't a process group leader. Some recipes even suggest two **fork()**s so the daemon can't re-acquire the terminal, but it's not strictly necessary.

At this point, the daemon is already running standalone. Now, all file descriptors are closed, except 0, 1, and 2 (standard input, output, and error), which are redirected to **/dev/null**. The easiest way to achieve this in Linux is to iterate over **/proc/self/fd**. Portable code can't rely on it though, so it's common to iterate up to **getrlimit(RLIMIT_NOFILE)**, or the highest file descriptor number available. Finally, the daemon's working directory is changed to **/** or another location on a known unmountable filesystem. Optionally, a daemon can also disable core dumps to prevent the leakage of sensitive data.

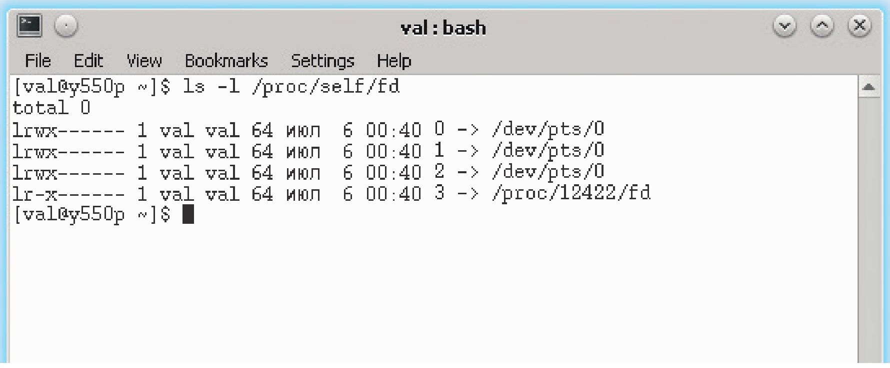
Daemons don't have to run as root except when they do privileged operations. The common case is binding to a network port below 1024, or writing to a system-wide location such as **/run** (see below). Usually, daemons do this during initialisation and drop privileges afterwards. Many daemons have their own designated user (such as **www-data** for *Apache*'s **httpd**), or run as 'nobody', which is essentially the same. This limits potential negative effects in case the daemon appears to be vulnerable.

Are you alive?

Quite often, only one instance of a daemon can run at a time. So, a mechanism for mutual exclusion is needed. In Windows, it would be a mutex, and if we were to follow this route, we could also use a semaphore (covered in LV015). However, historically Unix had another, much simpler tool for this job – a pid file.

A pid file is just a text file with a known name, which contains the daemon's PID. If it exists, the daemon is either running or was shutdown uncleanly. In the former case, you can do **kill -SIGHUP $(cat /run/daemon.pid)** to send the daemon a signal.

Traditionally, pid files used to live in **/var/run**. Newer Linuxes switched to **/run**, as **/var** was often unavailable during early system initialisation steps. The Filesystem Hierarchy Standard (FHS) 3.0, now current, made it official. In your distribution, **/var/run** is probably a symlink to **/run**.



```
[val@y550p ~]$ ls -l /proc/self/fd
total 0
lrwx------ 1 val val 64 июл  6 00:40 0 -> /dev/pts/0
lrwx------ 1 val val 64 июл  6 00:40 1 -> /dev/pts/0
lrwx------ 1 val val 64 июл  6 00:40 2 -> /dev/pts/0
lr-x------ 1 val val 64 июл  6 00:40 3 -> /proc/12422/fd
[val@y550p ~]$ 
```

Iterating over **/proc/self/fd** is a convenient way to know which file descriptors are open in a process.

Pid files are sometimes created with a write lock. This ensures atomicity: only one process can hold such a lock at any one time. So, only one daemon instance can be started. This is not the only possible approach, though. Many daemons opt for simpler ways, like opening the pid file with an **O_EXCL** flag.

A bit of C code

That's enough theory – let's practice a bit. Writing portable daemonisation code that runs across Unices is somewhat non-trivial. Luckily, there are some helpers (albeit with caveats). The GNU C Library (**glibc**) provides the **daemon(3)** function, which already does forking, creates new sessions, changes the current directory to **/** and redirects standard input, output and error to **/dev/null**. It's non-standard (although BSD systems have it as well), so you can't rely on it in portable code. Also, **daemon(7)** advises against using it as it doesn't produce a proper System V daemon. But again, sometimes it's more than enough.

This is how your daemonisation code may look like:

```
#include <errno.h>
#include <unistd.h>

int main(int argc, char **argv)
{
  int nochdir = 0, noclose = 0;
  if (daemon(nochdir, noclose) != -1) {
    postinit_daemon();
    run_daemon();
  } else {
    /* Check errno, handle error and exit */
  }
}
```

daemon(3) accepts two Boolean arguments. The first of these, **nochdir**, prevents it from changing the current working directory to **/**. The second, **noclose**, prohibits redirecting standard input, output and error to **/dev/null**. The function automatically calls **exit(2)** for the parent process, so error-handling appears in the child only. Keep this in mind because it may already have no terminal to print the error message on.

Note that **daemon(3)** neither closes file descriptors, nor resets **umask** or signal handlers. Pid files are also your duty. Indeed, the function is quite straightforward and encompasses fewer than 50 lines of code in the *glibc 2.21* release. So, the **postinit_daemon()** should take care of these details:

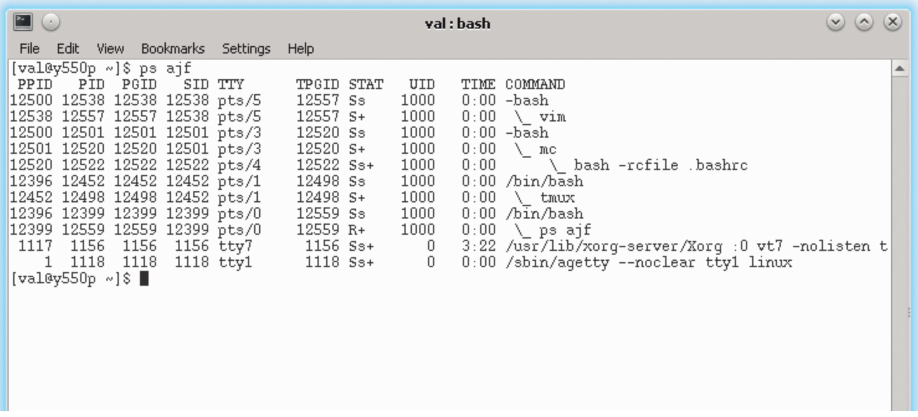
```
#include <signal.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#define PIDFILE_PERMS (S_IRUSR | S_IWUSR |
S_IRGRP | S_IROTH)

void postinit_daemon()
{
  struct sigaction sigact;
  struct flock lock;
  char buf[32];
  int fd;

  /* Clear the umask */
  umask(0);

  /* Assign signal handlers */
```

Leaving traces

Daemons are non-interactive, but they also need your attention from time to time. When this happens, they usually send a message to the system log. The log level indicates when an action on the administrator's side is required immediately, or the daemon merely informs him of some event.

Old-style daemons usually use two logging mechanisms. If they run in foreground, the messages are sent to the console, and sometimes logs are made more verbose automatically. In background mode, **openlog(3)**, **syslog(3)** and **closelog(3)** functions are used to send messages to a syslog flavour listening on **/dev/log** Unix socket (LV015).

New-style daemons unify these mechanisms. They just print log messages to **stdout/stderr**, whether it is real or intercepted by *Systemd* as per the **StandardOutput=** parameter in unit file or system-wide settings. Log level is encoded in square brackets at the very beginning of the message itself (**<0> This is an emergency!**), much as it is done in the Linux kernel.



```
[val@y550p ~]$ ps ajf
 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
12500 12538 12538 12538 pts/5    12557 Ss    1000   0:00 -bash
12538 12557 12557 12538 pts/5    12557 S+    1000   0:00  \_ vim
12500 12501 12501 12501 pts/3    12520 Ss    1000   0:00 -bash
12501 12520 12520 12501 pts/3    12520 S+    1000   0:00  \_ mc
12520 12522 12522 12522 pts/4    12522 Ss+   1000   0:00      \_ bash -rcfile .bashrc
12396 12452 12452 12452 pts/1    12498 Ss    1000   0:00 /bin/bash
12452 12498 12498 12452 pts/1    12498 S+    1000   0:00  \_ tmux
12396 12399 12399 12399 pts/0    12559 Ss    1000   0:00 /bin/bash
12399 12559 12559 12399 pts/0    12559 R+    1000   0:00  \_ ps ajf
 1117  1156  1156  1156 tty7      1156 Ss+      0   3:22 /usr/lib/xorg-server/Xorg :0 vt7 -nolisten t
    1  1118  1118  1118 tty1      1118 Ss+      0   0:00 /sbin/agetty --noclear tty1 linux
[val@y550p ~]$ █
```

With **ps(1)**, you can easily see which sessions and process groups exist on your system. Note that a shell is usually the session leader.

```
  sigact.sa_handler = reload_config;
  if (sigaction(SIGHUP, &sigact, NULL) != 0) {
    /* log error and exit */
  }
  ...
```

Here, we reset **umask** and assign **SIGHUP** a custom handler that re-reads the config. In many real-world cases, **reload_config()** would merely set some flag. The main loop (not shown here, but discussed in LV016) will check it periodically, and reload the config if the flag is raised. Generally, you should avoid signal handlers containing heavy logic such as reading and parsing files.

The rest of the function creates a pid file:

```
  ...
  fd = open("/run/coretechd.pid", O_RDWR | O_
CREAT, PIDFILE_PERMS);
  if (fd == -1) {
    /* Likely insufficient permissions. */
  }

  lock.l_type = F_WRLCK;
  lock.l_whence = SEEK_SET;
  lock.l_start = 0;
  lock.l_len = 0;
  if (fcntl(fd, F_SETLK, &lock) == -1) {
    if (errno == EACCES || errno == EAGAIN) {
      /* Already locked: complain and exit */
    }
  }
  ftruncate(fd, 0);
  snprintf(buf, sizeof(buf), "%lu\n", (unsigned long)
getpid());
  write(fd, buf, strlen(buf));
}
```

Note that we ignore some errors for brevity reasons (don't do this in real code!). **open(2)** creates the file, which can fail if the daemon's user has no write permissions to **/run**. The assumption is that the daemon runs as root while executing this function, so the pid file will be owned by root as well. It will also have `0644` permissions, as the third argument to **open(2)** dictates. That's why resetting **umask** is important: otherwise, actual permissions could differ from what we set here.

Then we request a write lock to a whole file with **fcntl(2)**. If the file is already locked (which means another daemon instance is running), the code complains and exits. As noted above, locks are not the only option. However, they have some benefits. When a process exits, the lock is released automatically, so if the pid file exists but is unlocked, we know it's stale. Finally, the file is truncated to discard possible leftover contents, and the new value is written.

As you see, creating daemons this way involves a decent dose of elbow grease.

"Love it or hate it, but it looks like Systemd is here to stay – even in Ubuntu, the motherland of Upstart."

We'll learn some ways to reduce boilerplate in the next few sections.

A Python way

Remember I told you that C is not the only choice for writing daemons? You can do it in Python if you wish, and as with many things in Python, it's usually simpler. In fact, Python's standard daemonisation interface is in PEP-3143, and a reference implementation is also available from PyPi as **python-daemon**. Together, they provide a high-level interface that hides much of the grunt work needed to produce a well-behaved Unix daemon.

python-daemon is centred around the **DaemonContext()** class. As the name suggests, it implements Pyhton's context manager protocol, and can be used with **with**. To make a daemon, you created an instance of **DaemonContext** configured the way you want, and then run your code under it:

```
import daemon

with daemon.DaemonContext():
  # call your main() function
```

You can try it and see that the empty non-configured **DaemonContext** is already usable, as **python-daemon** provides sensible defaults for all parameters. But there are many adjustments available as well:

```
import signal
from lockfile import pidlockfile

daemon_context = daemon.DaemonContext(
  pidfile=pidlockfile.PIDLockFile('/run/coretechd.
pid'),
  signal_map={
    signal.SIGTERM: cleanup_and_exit(),
    signal.SIGHUP: reload_config(),
  },
  chroot_directory='/var/lib/coretechd'
)
```

Here, we specify a custom pid file and signal handlers, something you would have to do yourself in plain C code. Note this is done in a declarative manner: you say what you need, and it's up to the library to implement how to do it. Unfortunately, due to the way that **pylockfile 0.10.2** is implemented, multiple daemon instances may still hang around.

The **chroot** option sets **/run/lib/example** as a root directory for the daemon. Chroots are good for limiting what the daemon can access on the filesystem. For this reason chroots must be self-contained, as there 's no easy way for a daemon to call a command or load a library from the outside. For instance, many FTP servers chroot into an FTP directory tree, which is why you spot **/bin/ls** in FTP listings occasionally. Note there are still ways to escape the "chroot jail", so this is not a security measure (as some may tell you).

You should really have a look at PEP-3143: it's short, concise, and provides a good summary of what a well-behaved Unix daemon is. A useful read even if you're not going to code your next daemon in Python.

New-style daemons

Now you have some understanding of how to write a daemon. The truth is you don't need to do it in modern Linux. With newer initialisation systems like *Upstart* or *Systemd*, all you have to do is to make a simple terminal application, and they handle the rest. You get a clean environment to run your code and can focus on your application logic instead of implementing infrastructure things like logging.

Love or hate it, but it looks like *Systemd* is here to stay. Even Ubuntu, the motherland of *Upstart*, made a switch with its 15.04 release. The approaches to new-style daemons are quite similar in both systems, but the syntax is (of course) different. Given these facts, we'll focus on *Systemd* here; you can easily adapt to *Upstart*, should you need it.

Systemd operates daemons as units. Units are described in INI-style text files, and administrator-created units are stored in **/etc/systemd/system**. Many unit types are supported; for daemons, you need a **.service** unit. Try this:

```
[Unit]
Description=Core Tech example daemon
[Service]
Type=simple
ExecStart=/usr/sbin/coretechd
Restart=on-abort

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

Surprisingly, that's already enough for *Systemd* to daemonise your code. **ExecStart** is a command to execute when starting the daemon. **WantedBy** effectively makes it a soft dependency for a multi-user (or normal) boot. If the daemon terminates uncleanly, *Systemd* will restart it automatically.

There are some caveats, though. First, your daemon must not fork. In fact, it shouldn't be a daemon at all: it should be a regular program. Many daemons provide a command-line switch (often called **-f**) to enable foreground mode; use it, if possible. For daemons that absolutely need to fork, use **Type=forking** and **PIDFile=/run/coretechd.pid**, so that *Systemd* knows which process to track.

Now, you can start your daemon with **systemctl start coretechd** and grep the logs to see if it succeeded:

```
$ sudo journalctl -b _SYSTEMD_UNIT=coretechd.service
-- Logs begin at Thu 2013-09-12 00:28:54 YEKT, end at Tue 2015-07-07 00:58:06 YEKT. --
Jul 07 00:53:52 y550p coretechd[8291]:  Hello from a new-style daemon!
```

This log entry comes from the ordinary **printf(3)** (see the boxout).

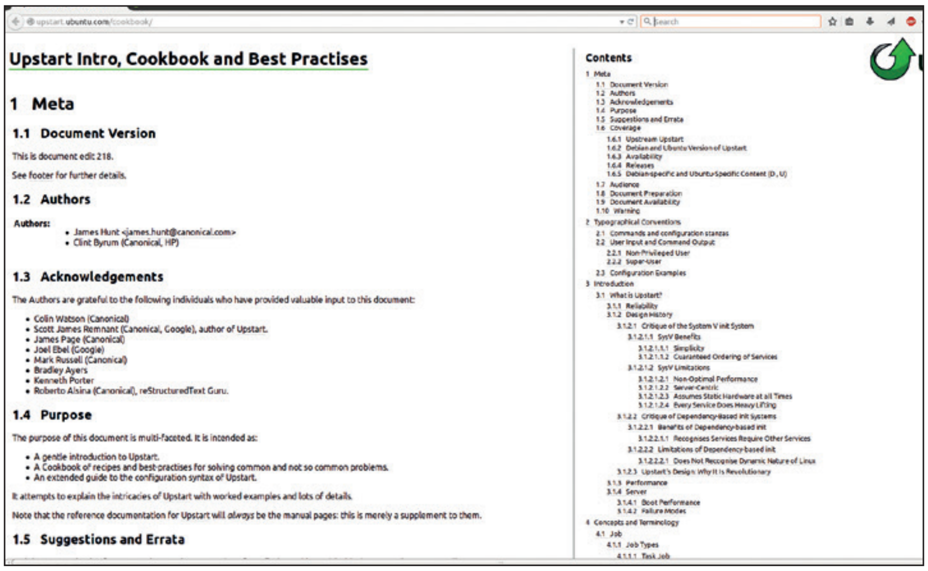
The approach shown here isn't optimal. It's just simplest for the code not written for *Systemd* specifically (hence the name). Tighter integration opens many more possibilities like socket or bus activation, but discussing it would take us too far from the topic of general daemonization. However, if you feel *Systemd* is worth a Core Tech of its own, please drop us a line.

Ad-hoc daemonisation

Imagine you started a command and expected it to run for a minute. Now imagine that it spent half an hour working and you realised that you needed to close your laptop and move to another place. What should you do not to lose your work?

Effectively, you want your command (a foreground process) to become a daemon. There are several recipes, but we'll discuss one. First, press Ctrl+Z to pause the command – you'll return to the command prompt. Now, make the command a background job with **bg**. Finally, run **disown -h %job_id**, where **job_id** is what **bg** returned. This removes the command from the shell's job list, so it won't get a **SIGHUP** when the terminal will be closed.

Note that if you anticipate that the command will last for long from the beginning, using **tmux** or **screen** will probably be a better alternative.


upstart.ubuntu.com/cookbook/

Search

Upstart Intro, Cookbook and Best Practises

1 Meta

1.1 Document Version

This is document edit 218.

See footer for further details.

1.2 Authors

Authors:
- James Hunt <james.hunt@canonical.com>
- Clint Byrum (Canonical, HP)

1.3 Acknowledgements

The Authors are grateful to the following individuals who have provided valuable input to this document:

- Colin Watson (Canonical)
- Scott James Remnant (Canonical, Google), author of Upstart.
- James Page (Canonical)
- Joel Ebel (Google)
- Mark Russell (Canonical)
- Bradley Ayers
- Kenneth Porter
- Roberto Alsina (Canonical), reStructuredText Guru.

1.4 Purpose

The purpose of this document is multi-faceted. It is intended as:

- A gentle introduction to Upstart.
- A Cookbook of recipes and best-practises for solving common and not so common problems.
- An extended guide to the configuration syntax of Upstart.

It attempts to explain the intricacies of Upstart with worked examples and lots of details.

Note that the reference documentation for Upstart will *always* be the manual pages: this is merely a supplement to them.

1.5 Suggestions and Errata

Contents

1 Meta
  1.1 Document Version
  1.2 Authors
  1.3 Acknowledgements
  1.4 Purpose
  1.5 Suggestions and Errata
  1.6 Coverage
    1.6.1 Upstream Upstart
    1.6.2 Debian and Ubuntu Version of Upstart
    1.6.3 Availability
    1.6.4 Releases
    1.6.5 Debian-specific and Ubuntu-Specific Content (D, U)
  1.7 Audience
  1.8 Document Preparation
  1.9 Document Availability
  1.10 Warning
2 Typographical Conventions
  2.1 Commands and configuration stanzas
  2.2 User Input and Command Output
    2.2.1 Non-Privileged User
    2.2.2 Super-User
  2.3 Configuration Examples
3 Introduction
  3.1 What is Upstart?
    3.1.1 Reliability
    3.1.2 Design History
      3.1.2.1 Critique of the System V init System
        3.1.2.1.1 SysV Benefits
          3.1.2.1.1.1 Simplicity
          3.1.2.1.1.2 Guaranteed Ordering of Services
        3.1.2.1.2 SysV Limitations
          3.1.2.1.2.1 Non-Optimal Performance
          3.1.2.1.2.2 Server-Centric
          3.1.2.1.2.3 Assumes Static Hardware at all Times
          3.1.2.1.2.4 Every Service Does Heavy Lifting
      3.1.2.2 Critique of Dependency-Based Init Systems
        3.1.2.2.1 Benefits of Dependency-based init
          3.1.2.2.1.1 Recognises Services Require Other Services
        3.1.2.2.2 Limitations of Dependency-based init
          3.1.2.2.2.1 Does Not Recognise Dynamic Nature of Linux
      3.1.2.3 Upstart's Design: Why It Is Revolutionary
    3.1.3 Performance
    3.1.4 Server
      3.1.4.1 Boot Performance
      3.1.4.2 Failure Modes
4 Concepts and Terminology
  4.1 Job
    4.1.1 Job Types
      4.1.1.1 Task Job

Should you ever need to write an *Upstart* job, *Upstart Cookbook* has everything you'll need to know.

Command of the month: daemon(1)

Systemd and *Upstart* aren't the only ways to "automagically" daemonise your code. A small tool creatively named **daemon** (**http://libslack.org/daemon**) does the same, and plays well with plain old System V init.

daemon(1) starts the command you choose in a well-defined environment, restarts it if necessary, and captures its output as logs. It's highly configurable and you can also use it to manage the daemon or check its status. Here is an example command-line:

```
daemon --respawn --name coretechd --chdir / \
  --pidfiles /run --umask 0 \
  --stdout daemon.info --stderr daemon.err \
  /usr/sbin/coretechd
```

This creates a single named daemon instance and stores its PID in **/run/coretech.pid**. The daemon's standard output is sent to syslog as info messages. If the daemon crashes, it is restarted (**--respawn**) and the error is supposedly logged (**--stderr**).

While this works, it's not the way you usually create daemons. Most of the time, they are started on boot with an init system. For System V init, this means the daemon needs an init script to be placed in **/etc/init.d**. The good news is that daemon comes with the template, and adjusting it to your needs is usually a matter of setting some variables. **daemon(1)** is arguably a preferred way to daemonisation if you can't rely on *Systemd* or *Upstart*.

FOSSpicks

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

Our editor **Graham Morrison** is a fearless explorer of the internet – look, he's found some excellent Free Software on his travels!

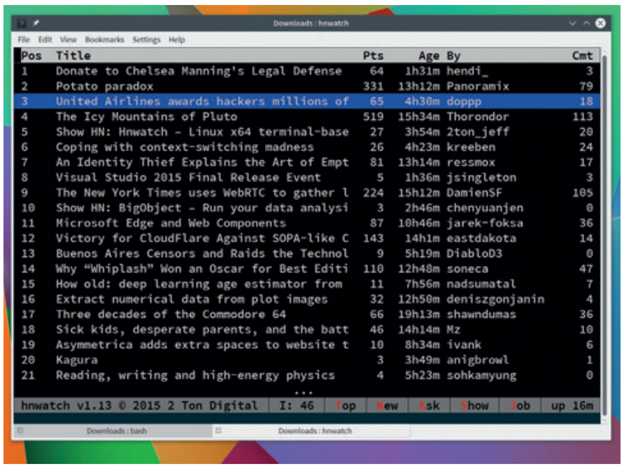
CLI news feed

hnwatch

The internet is so full of distractions that we're contemplating leaving pervasive data behind and founding a disconnected commune in the Garonne, purely to preserve our sanity and some meaning to life.

However, there are still a few online bastions of collective wisdom, and one of these is Hacker News (**https://news.ycombinator.com**). Despite a few bumps along the road, Hacker News has mostly been able to maintain both a high quality of linked content and comments for several years, making it a worthwhile destination for your browser. But browsers are evil tools of distraction. They make it impossible to have only a single tab open. *hnwatch* is the answer, at least until you find a story you want to read. You run it from the command line and it fills your terminal with the latest or top Hacker News stories in real time, as they move up and down the hierarchy. You can click on links to open them in your default browser, or read the comments directly from the command line.

PROJECT WEBSITE
https://2ton.com.au/hnwatch



```
Pos  Title                                         Pts     Age By                 Cmt
1    Donate to Chelsea Manning's Legal Defense      64   1h31m hendi_               3
2    Potato paradox                                331  13h12m Panoramix           79
3    United Airlines awards hackers millions of     65   4h30m doppp               18
4    The Icy Mountains of Pluto                    519  15h34m Thorondor          113
5    Show HN: Hnwatch - Linux x64 terminal-base     27   3h54m 2ton_jeff           20
6    Coping with context-switching madness          26   4h23m kreeben             24
7    An Identity Thief Explains the Art of Empt     81  13h14m ressmox             17
8    Visual Studio 2015 Final Release Event          5   1h36m jsingleton           3
9    The New York Times uses WebRTC to gather l    224  15h12m DamienSF           105
10   Show HN: BigObject - Run your data analysi      3   2h46m chenyuanjen          0
11   Microsoft Edge and Web Components              87  10h46m jarek-foksa         36
12   Victory for CloudFlare Against SOPA-like C    143   14h1m eastdakota          14
13   Buenos Aires Censors and Raids the Technol      9   5h19m DiabloD3             0
14   Why "Whiplash" Won an Oscar for Best Editi    110  12h48m soneca              47
15   How old: deep learning age estimator from      11   7h56m nadsumatal           7
16   Extract numerical data from plot images        32  12h50m deniszgonjanin       4
17   Three decades of the Commodore 64              66  19h13m shawndumas          36
18   Sick kids, desperate parents, and the batt     46  14h14m Mz                  10
19   Asymmetrica adds extra spaces to website t     10   8h34m ivank                6
20   Kagura                                          3   3h49m anigbrowl            1
21   Reading, writing and high-energy physics        4   5h23m sohkamyung           0
                                                ...
hnwatch v1.13 © 2015 2 Ton Digital | I: 46 | Top | New | Ask | Show | Job | up 16m
```

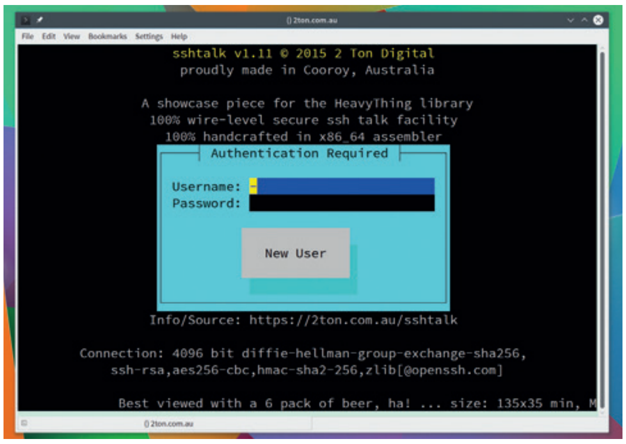
Don't open that browser! Access your favourite Hacker News stories from the command line instead.

Secure group chat

sshtalk

While perusing the site of the developer, Jeff Marrison, who created 'hnwatch' above, we found another little nugget written in the same Turbo Assembler-alike code. It's a great little utility that can turn any SSH server and client into a secure chat server, a little like IRC but without the configuration files.

The huge advantage that *sshtalk* has is that it uses SSH's perfect forward secrecy, protecting your communication with the same heavyweight encryption protocols as SSH itself. It's also easy to run, attaching itself to port 4001 on your server after simply executing the binary. If you want to build these tools yourself, you'll need to grab and build the assembler framework, which adds to the complexity. But if you want to connect to the server, you don't need anything other than the humble SSH itself. Just **ssh** to the server address and you'll see a banner screen followed by a login window. You don't need an SSH account, but you can create a new chat account from this login prompt, after which you'll be dropped into the chat interface itself. You need to first connect or join a room, so you'll need someone else to invite you or tell you the room name. All the Ctrl commands are listed in below the buddy view on the right. You can also create rooms easily with Ctrl+J, after which you can chat away without worrying about security or setup.



```
              sshtalk v1.11 © 2015 2 Ton Digital
               proudly made in Cooroy, Australia

           A showcase piece for the HeavyThing library
            100% wire-level secure ssh talk facility
              100% handcrafted in x86_64 assembler
          ┌───── Authentication Required ─────┐
          │ Username:                         │
          │ Password:                         │
          │                                   │
          │            New User               │
          └───────────────────────────────────┘
            Info/Source: https://2ton.com.au/sshtalk

    Connection: 4096 bit diffie-hellman-group-exchange-sha256,
        ssh-rsa,aes256-cbc,hmac-sha2-256,zlib[@openssh.com]

       Best viewed with a 6 pack of beer, ha! ... size: 135x35 min, M
```

Hand-crafted assembler tools like *hnwatch* and *sshtalk* feel like the products of a computing artisan movement.

"sshtalk can turn any SSH server and client into a secure chat server."

PROJECT WEBSITE
https://2ton.com.au/sshtalk

NAS distribution

Rockstor

There are several great network attached storage distributions, enabling you access your files from anywhere. But Rockstor is the first we've seen to specifically take advantage of the features found in Btrfs. Btrfs, or the B-tree filesystem as it's sometimes known, is finally considered production quality, after many years in development. It's considered a next-generation filesystem because it integrates many advanced ideas that used to be added alongside the filesystem, rather than part of it. These features include storage redundancy, replication, device linking, quota groups, sub-volumes, snapshots and encryption.

Rockstor has used all these features as the foundation for its NAS solution, along with lots of other tools for adding apps and containers and even for migrating your NAS into the cloud, which is where Rockstor is hoping to make its living. As it's open source, Centos-based, and easy to install, it's perfect for running at home.

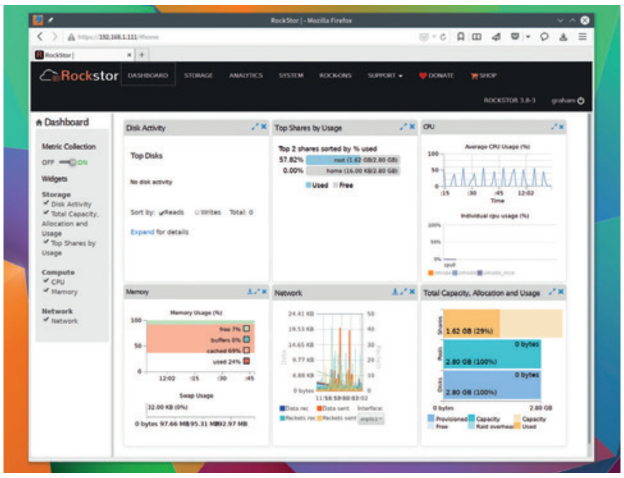
One of the things we really like about both Rockstor's installer and the running environment is that the user-interface design is exceptional. Installation requires just a couple of clicks and the creation of a couple of user accounts, if needed, and takes only a few minutes. After installation has completed, reboot your system and connect to the IP address of the machine (the IP address is output on the console of the server). You now need to create an account, if you hadn't at install time, and log in. The first time you do this, you'll likely be prompted to update your installation. Rockstor is developing at a rapid pace, and recent releases have cleverly side-stepped a few issues with Btrfs, so it's worth keeping things up to date. Our first update took around five minutes to apply.

Share the rocks

Rockstor is easy to use. Your storage can be divided into shares for different categories of data, and different access permissions. Click on Storage to manage these. By default, 'Home' and 'Root' are created by the operating system, and you'll need to add at least one more for your own data.

Thanks to Btrfs, shares can be cloned or snapshots made of their current state, and these features are available from the Shares Management window. For accessing files from other Linux devices, enable Samba file sharing from the System page, then add a Samba export profile by clicking on Storage > File Sharing > Samba. Select a share and a user from the drop-down lists and disable 'read-only'.

One final awesome aspect to Rockstor we'd like to highlight are the 'Rock-ons'. These are Docker-based containers for running all kinds of web apps, installed via an app store front-end. There are 'Rock-ons' for easy WordPress, OpenVPN, OwnCloud, Transmission and BtSync, and we think they're a brilliant idea.

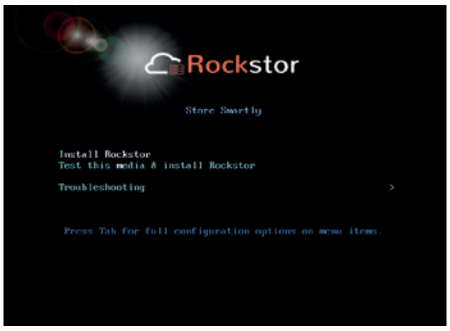


The dashboard provides a great overview of network, storage and CPU activity for your entire NAS

"Rockstor is the first NAS we've seen to take advantage of Btrfs."

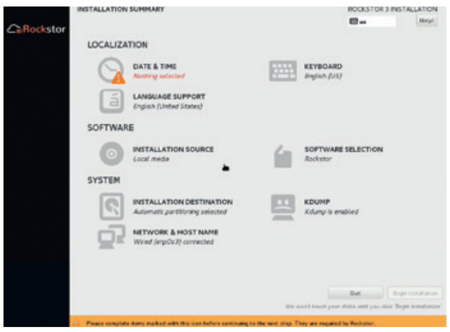
PROJECT WEBSITE
http://rockstor.com

How it works: Install Rockstor


Rockstor
Store Smartly

```
Install Rockstor
Test this media & install Rockstor

Troubleshooting                                   >

 Press Tab for full configuration options on menu items.
```

1 Download the ISO and either burn it to an optical drive or write it to a USB stick. First boot should bring up the above menu.

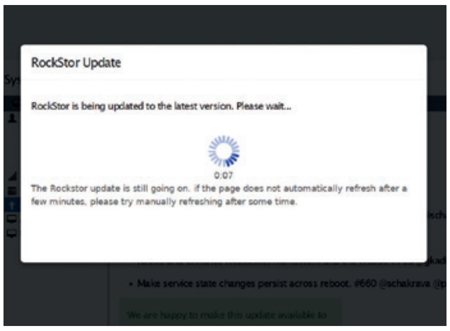

INSTALLATION SUMMARY — ROCKSTOR 3 INSTALLATION

LOCALIZATION
- DATE & TIME — Nothing selected
- KEYBOARD — English (US)
- LANGUAGE SUPPORT — English (United States)

SOFTWARE
- INSTALLATION SOURCE — Local media
- SOFTWARE SELECTION — Rockstor

SYSTEM
- INSTALLATION DESTINATION — Automatic partitioning selected
- KDUMP — Kdump is enabled
- NETWORK & HOST NAME — Wired (enp0s3) connected

2 The installer is easy to use. Unless you want to partition your storage manually, just check the date and create a root account/password.


RockStor Update

RockStor is being updated to the latest version. Please wait...

0:07

The Rockstor update is still going on. If the page does not automatically refresh after a few minutes, please try manually refreshing after some time.

3 After installation, reboot, and from another machine on your network, point a browser at Rockstor's IP address. Allow it to update.

Password manager

QtPass v0.9.0

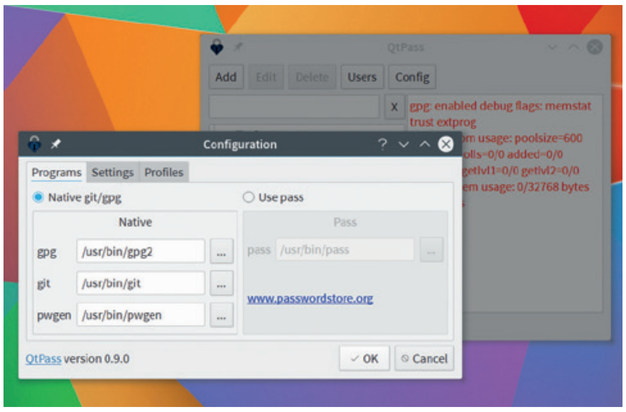
The problem with passwords is that we need to use so many of them – from online shopping to email – that it's easy to forget how important any single password is. Many people use a simple sequence for passwords, or worse, use the same password for multiple sites, exposing them all if one site is hacked.

There isn't a perfect solution, but using a password manager is a huge improvement. This encapsulates all your passwords into a single 'wallet' that can be opened with a one password, hopefully one that's unique and far more secure than the multitude you used before. *KeePassX* is our favourite, and has become something of a standard. Many people use the *Qt* front-end and the Android client, for example, sharing the encrypted keyfile using something like OwnCloud to keep passwords synchronised.

But *KeePassX* is old and we're always looking for new solutions. One of which is a command line tool called *pass*. *Pass* is brilliant because it uses GPG and your own identity key, along with the Unix filesystem itself to store your passwords, naming the files after where they're used and encrypting the contents of each with GPG.

That means, apart from creating and decrypting, you can manage your wallet with normal filesystem commands, and there are no weird file formats or databases to contend with. It all feels much less cryptic than the average password wallet, and it's just as secure. There's even an Android app, a *Firefox* plugin and a GUI client: *QtPass*. *QtPass* makes *pass* operate like any other wallet manager, with password creation and retrieval integrated into the simple, clean UI. It's easy to use, and because the back-end is still *pass*, you get all the advantages of a GUI and its unque way of creating a wallet.

"Pass is brilliant because it uses GPG and your own identity key."



gpg: enabled debug flags: memstat trust extprog
m usage: poolsize=600
olls=0/0 added=0/0
getlvl1=0/0 getlvl2=0/0
em usage: 0/32768 bytes

[✓ Native git/gpg] Use pass

Native
gpg /usr/bin/gpg2
git /usr/bin/git
pwgen /usr/bin/pwgen

pass /usr/bin/pass
www.passwordstore.org

QtPass version 0.9.0

QtPass is a simple GUI to the various commands that make up a working *pass* system.

PROJECT WEBSITE
https://qtpass.org

Password manager

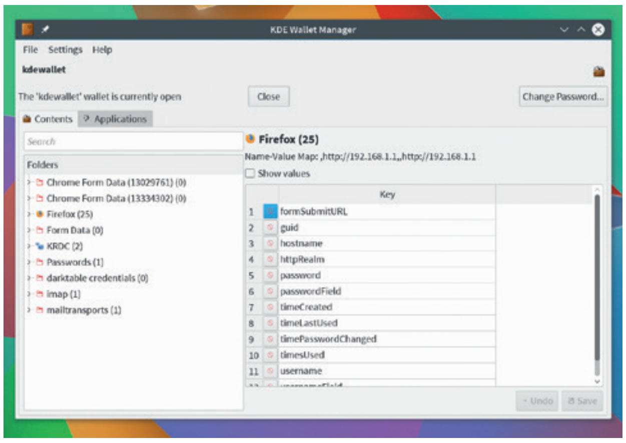
KWallet 5.12.0

There are many different password managers, and the one we'd recommend for general use is a stable version of *KeePassX*, as mentioned above. But if you spend most of your time in KDE, or using KDE applications, we'd recommend *KWallet* as a better alternative.

This is primarily because of its integration with the desktop. Any KDE tool that uses a password can use *KWallet*. There's also an excellent add-on for *Firefox*, which means *KWallet* can be used for nearly everything on a KDE desktop, and it works well with system events like suspending a session or when the screensaver kicks in. In those cases, you can choose to close the wallet, requiring you to enter your password again. *KWallet* has been around for a long time, and it's slightly disappointing that the management app – *KWallet Manager* – hasn't yet been updated for KDE/Plasma 5. The frameworks that do the clever stuff have, though, and the latest release includes a neat feature we think is worth covering.

Keep it secret, keep it safe

Normally, the wallet is a transparent part of your system, only visible when you log on or an application makes a request to access a password. But a new feature in *KWallet 5.12.0* adds an essential command line tool that gives you the ability to interact with your wallet directly from the command line. The command interface can, for example, list the passwords being stored and even access those passwords. More importantly, you can also write new entries and overwrite existing passwords. The reason why these facilities are so useful is that it means you can create your own integration with another password manager, enabling you to import or export your passwords as and when you need to.



The 'kdewallet' wallet is currently open

Chrome Form Data (13029761) (0)
Chrome Form Data (13334302) (0)
Firefox (25)
Form Data (0)
KRDC (2)
Passwords (1)
darktable credentials (0)
imap (1)
mailtransports (1)

Firefox (25)
Name-Value Map: ,http://192.168.1.1,,http://192.168.1.1
Show values

	Key
1	formSubmitURL
2	guid
3	hostname
4	httpRealm
5	password
6	passwordField
7	timeCreated
8	timeLastUsed
9	timePasswordChanged
10	timesUsed
11	username

Even though the *KWallet* front-end hasn't been updated, it still works perfectly with the latest back-ends.

PROJECT WEBSITE
www.kde.org

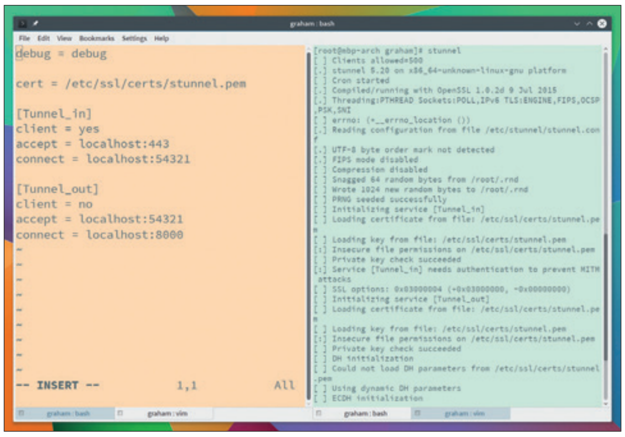
SSL wrapper

stunnel 5.20

Stunnel has been around for a long time. It's a proxy that's used to add SSL encryption to clients and services that don't support it. It's a little like using SSH as a SOCKS proxy, where you create an SSH connection with the **-p** and **-d** arguments before configuring your browser to use this connection as a SOCKS proxy.

Your local HTTP data is encrypted via SSH until it appears at the server. *stunnel* is more flexible and more complex as a result. It can be used to wrap lots of different services without configuring them specifically for SSL, such as SAMBA, POP3 or IMAP, as well as getting around port restrictions. Before you can use it, on the machine that you want to use as the 'server', where your remote clients will connect to, you'll need to create your own OpenSSL certificates and then a configuration file. Creating the certificate is easy enough, as long as you've got the **openssl** packages installed. We used the command **openssl req -new -out /etc/ssl/certs/stunnel.pem -keyout /etc/ssl/certs/stunnel.pem -nodes -x509 -days 365** to generate the prerequisites. The next step is to create a configuration file. At its simplest, you can define both a source and destination port, and the IP addresses needed for both. You'd need two sections in the configuration file for this – **[In]** and **[Out]**, for instance (those names are arbitrary), and they'll both need entries for 'client' (yes or no), 'accept' and 'connect'. In this way, you can tunnel data from one port on the client to another port on the server, which is useful if the client's network connection is blocking the port, for example. In this specific case, the data would go from the tunnel in Accept > Tunnel In Connect > Tunnel Out Accept > Tunnel Out Connect.

"Your HTTP data is encrypted via SSH until it appears at the server."


```
debug = debug

cert = /etc/ssl/certs/stunnel.pem

[Tunnel_in]
client = yes
accept = localhost:443
connect = localhost:54321

[Tunnel_out]
client = no
accept = localhost:54321
connect = localhost:8000
~
~
~
~
~
~
~
~
~
-- INSERT --                1,1           All
```
```
[root@mbp-arch graham]# stunnel
[ ] Clients allowed=500
[.] stunnel 5.20 on x86_64-unknown-linux-gnu platform
[ ] Cron started
[.] Compiled/running with OpenSSL 1.0.2d 9 Jul 2015
[.] Threading:PTHREAD Sockets:POLL,IPv6 TLS:ENGINE,FIPS,OCSP
,PSK,SNI
[ ] errno: (*__errno_location ())
[.] Reading configuration from file /etc/stunnel/stunnel.con
f
[.] UTF-8 byte order mark not detected
[.] FIPS mode disabled
[ ] Compression disabled
[ ] Snagged 64 random bytes from /root/.rnd
[ ] Wrote 1024 new random bytes to /root/.rnd
[ ] PRNG seeded successfully
[ ] Initializing service [Tunnel_in]
[ ] Loading certificate from file: /etc/ssl/certs/stunnel.pe
m
[ ] Loading key from file: /etc/ssl/certs/stunnel.pem
[:] Insecure file permissions on /etc/ssl/certs/stunnel.pem
[ ] Private key check succeeded
[:] Service [Tunnel_in] needs authentication to prevent MITM
 attacks
[ ] SSL options: 0x03000004 (+0x03000000, -0x00000000)
[ ] Initializing service [Tunnel_out]
[ ] Loading certificate from file: /etc/ssl/certs/stunnel.pe
m
[ ] Loading key from file: /etc/ssl/certs/stunnel.pem
[:] Insecure file permissions on /etc/ssl/certs/stunnel.pem
[ ] Private key check succeeded
[ ] DH initialization
[ ] Could not load DH parameters from /etc/ssl/certs/stunnel
.pem
[ ] Using dynamic DH parameters
[ ] ECDH initialization
```

There's not much to see with *stunnel* running, so we've split our console view to show the configuration file and the binary output.

PROJECT WEBSITE
https://www.stunnel.org/index.html

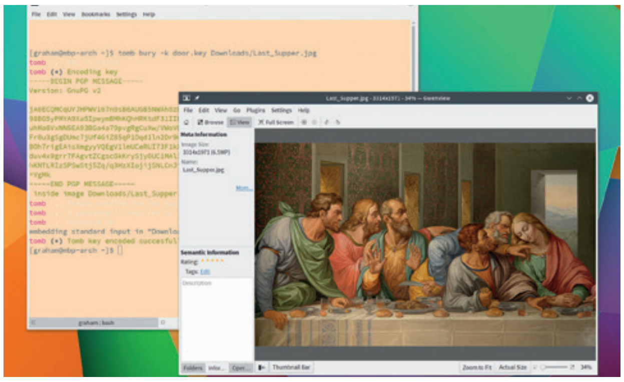
Encryption

Tomb 2.1

Edward Snowden famously used the open source encryption tool, *TrueCrypt*, to secure data partitions and portable storage that could be shared with his contacts. However, in May 2014, the *TrueCrypt* website announced the end of development while pointing to a final new (and read-only) restricted version.

This version was eventually declared safe by security firm Gibson Research Corporation, and as development had halted, several projects forked from the original. *VeraCrypt* is one such fork, and is now considered *TrueCrypt*'s successor. *Tomb*, on the other hand, isn't a fork a fork of *TrueCrypt*, but it has been designed to offer much of the same functionality while keeping usability as simple as possible. It's also a script, rather than a standalone project, encapsulating a variety of common encryption tools, such as *gnupg*, *Cryptsetup* and *Steghide*, into a simple command-driven interface.

What makes *Tomb* rather good is that this script is well documented, enabling almost anyone with some programming experience to take a look and audit for themselves. You start by creating a 'tomb' (**tomb dig -s 100 filename.tomb**). A tomb is a single file that acts like a locked folder containing the data you want to secure, which can then be unlocked, copied or transferred just as you would any encrypted file. You next need to bind a key to the file (**tomb forge filename.tomb.key**), lock and generate the LUKS filesystem on the file (**tomb lock filename.tomb -k filename.tomb.key**) – you'll be asked for a password. The key file can be kept separately for maximum security. When you now open the tomb file (**sudo tomb open filename.tomb -l filename.tomb.key**), your tomb file will be mounted into the filesystem for general use.


```
[graham@mbp-arch ~]$ tomb bury -k door.key Downloads/Last_Supper.jpg
tomb
tomb (*) Encoding key
-----BEGIN PGP MESSAGE-----
Version: GnuPG v2

[illegible]
=VgMk
-----END PGP MESSAGE-----
 inside image Downloads/Last_Supper
tomb
tomb
tomb
embedding standard input in "Downlo
tomb (*) Tomb key encoded succesful
[graham@mbp-arch ~]$ 
```

One of *Tomb*'s best features is the ability to hide a key within an image file, or as a QR code for physical printing.

PROJECT WEBSITE
https://www.dyne.org/software/tomb

Audio editor

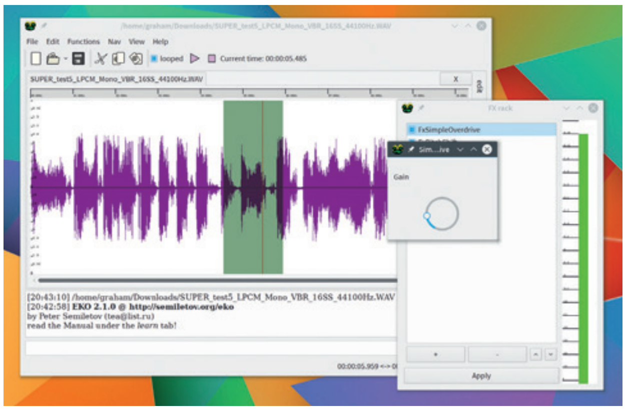
Eko 2.1.0

Like *Gimp* for image files, *Audacity* edits audio. And it's a wonderful tool. It can edit multiple tracks of audio at the same time, process them with all kinds of effects, and perform many essential audio tasks. It's capable of such professional results that many professionals use *Audacity* to fine-tune their recordings after they've mixed down or recorded their audio. But *Audacity* does have a few problems. It's not 100% stable, nor is the user-interface to everyone's taste. Which is a long way of saying, as with *Gimp*, there's space for some competing applications.

Which is why the appearance of first a 2.0 and now a 2.1 release of another audio editor, *Eko*, has us excited. *Eko* is a simple audio editor with a simple set of dependencies. Recent releases have also removed Jack as a requirement, making it work with native audio hardware without configuration. It's difficult to get hold of, however, as the site that hosts the source code was offline as soon as the update was posted. We ended up extracting the **tar.gz** file from the source RPM build for OpenSUSE, and building it that way, so hopefully the home site is back online by the time you read this.

The application itself uses *Qt* for the interface. It's quick and easy to use. You can load more than one audio file and they'll appear as tabs in a display that can be themed. Most importantly, and unlike *Audacity*, you can add effects and process audio during playback, which makes editing more intuitive than *Audacity*.

However, there's still a long way to go before we'd recommend this as a replacement. Zoom levels aren't great, sample rate support is poor and there's no absolutely no input feedback while recording; however, if you're interested in audio, it's definitely worth a try.


/home/graham/Downloads/SUPER_test5_LPCM_Mono_VBR_16SS_44100Hz.WAV
File
Edit
Functions
Nav
View
Help
looped
Current time: 00:00:05.485
SUPER_test5_LPCM_Mono_VBR_16SS_44100Hz.WAV
FX rack
FxSimpleOverdrive
Sim...ive
Gain
Apply
00:00:05.959 <->

```
[20:43:10] /home/graham/Downloads/SUPER_test5_LPCM_Mono_VBR_16SS_44100Hz.WAV
[20:42:58] EKO 2.1.0 @ http://semiletov.org/eko
by Peter Semiletov (tea@list.ru)
read the Manual under the learn tab!
```

Eko is a simple audio editor. It's quick with real-time effects, but no input monitoring.

> "Eko is a simple audio editor with a simple set of dependencies."

PROJECT WEBSITE
http://semiletov.org/eko

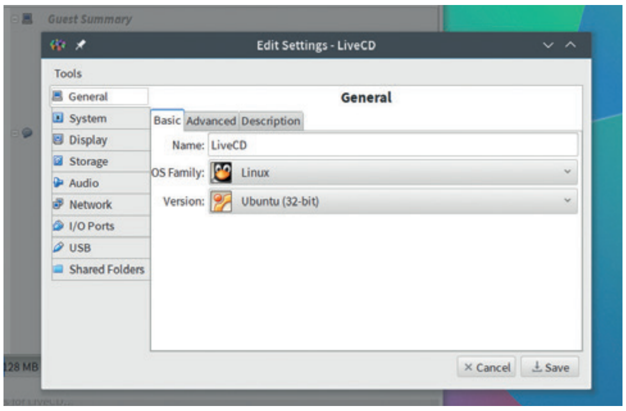
VirtualBox manager

RemoteBox 2.0

RemoteBox is a rather clever management interface for *VirtualBox*, enabling you to remotely manage a remote server running *VirtualBox*. This gives you access and remote control of whatever virtual machines are installed on the server, turning a low-powered remote machine into a virtual desktop, or enabling you to manage containerised *VirtualBox* servers as you would a local installation.

RemoteBox is written in Perl and has a fair few dependencies. You also need to create a simple configuration file on the server and run the **vboxwebsrv** command, either through *Systemd* or your own init scripts. We found it easier when running **vboxwebsrv** as a simple user and using that user's login credentials from *RemoteBox*. This should also provide an extra level of security in case the server you're accessing is doing other important things at the same time.

With everything installed and running, *RemoteBox* provides a very similar interface to the native *VirtualBox*, communicating with the server through SSL and SOAP, which provides access to nearly every parameter of the native client. The interface is actually cleaner and more intuitive in many ways. Virtual machines are listed, run and saved. You can connect to their remote desktops too, using the RDP protocol, letting you point and click your way around whatever machines you have running. It's a good way of getting access to a powerful desktop from a less-powerful client. You can accomplish the same thing by simply using *RDP* from *VirtualBox*, but the management and control interface for more than one machine helps *RemoteBox* to be far more flexible and powerful.



Name: LiveCD
OS Family: Linux
Version: Ubuntu (32-bit)

With *RemoteBox*, you can perform almost any task you can with the *VirtualBox* GUI, only from a remote client.

> "RemoteBox provides a very similar interface to the native VirtualBox."

PROJECT WEBSITE
http://remotebox.knobgoblin.org.uk

FOSSPICKS Old Games

Strategy game

Battle for Wesnoth 1.12.4

Battle for Wesnoth is not a new game – it's been 12 years in development – nor is this a recent release, both normally prerequisites for being featured here. The last major update was version 1.12.0, released in November 2014 after almost three years of work, and is one of the best and most professionally finished open source games you can install.

It's a tactical turn-based strategy game, which means you make decisions while the game is in a pause state, moving your units, upgrading and attacking the enemy before finishing your turn, when the enemy will then respond. There are small missions, tutorials and fully fledged campaigns, all wrapped around excellent art work, audio, scripting and translations, not to mention an engaging and proactive community. It's one of the best examples of what successful open source gaming can look like.

But the reason why we're covering it now and not waiting for the 1.14.0 release is that 1.14.0 might never happen. The project needs help. In a post on the main site, dated 25 July 2015, the project admitted that it's understaffed.

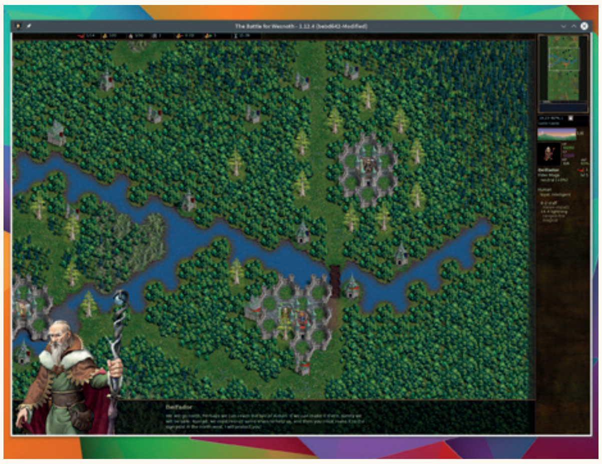
"At this time, there are fewer than half a dozen developers working on each new version of the game, and even fewer of them are able to work on the engine itself. We do not collectively have the time or skills to fix bugs as quickly as we should, or implement features as rapidly as we would like," states the post, before simply stating "the ship is sinking". The project needs programmers. Either intermediate C++, especially for Windows and OS X, Python programmers able to maintain the project's toolchain, and coders of the Wesnoth Markup Language, used to create the campaigns played within the game.

Get involved!

If you think you might be able to help, this might be a great chance to dive into a well established open source project (and one with lots of potential if you're thinking of a career in games design). Take a look at the **#wesnoth-dev** IRC channel on **irc.freenode.net**.

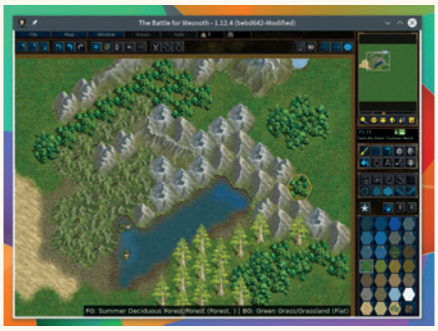
Wesnoth is undoubtedly worth saving. The tutorials ease you into the technical aspects of what each race is capable of, how to build your resources, work your units and expand into new territory. It's really not that difficult, and far too easy to get addicted. You'll soon find yourself sucked into the plot of one of the campaigns, attacking each challenging mission one after the other.

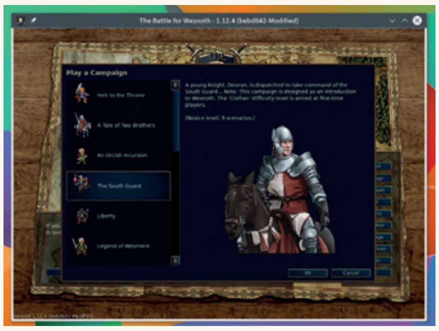
There are very few games we could recommend that are of a similar quality, having been developed for so long, and we'd love to see a new generation of contributors get involved with the project. If you think this sounds like you, download the latest version and take a look at the code then get in touch with the developers. Then let us know how you get on!

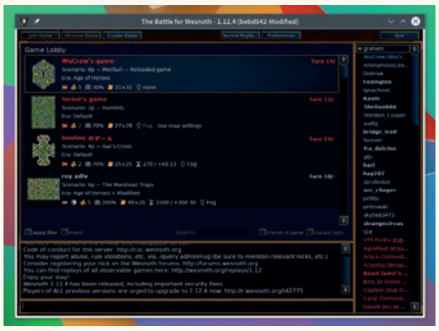

There are many, many hours of gameplay to be had in *Wesnoth*. There's even a multiplayer mode.

PROJECT WEBSITE
www.wesnoth.org

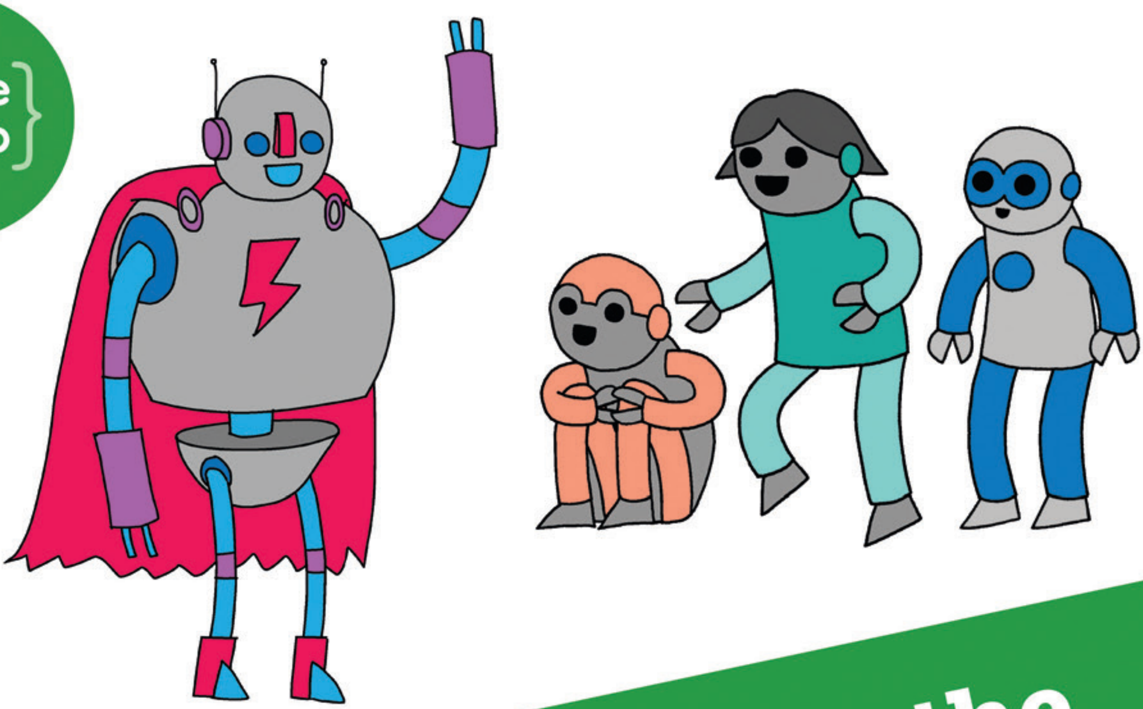
Battle for Wesnoth's best features


1 There's a brilliant map editor where you can create your own challenges, and later, script your own missions and campaigns.


2 There's a huge amount of variety in the difficulty and type of campaigns offered within the game.


3 But there's nothing like dealing with the unpredictable nature of playing with real humans – *Wesnoth* has this too.

{code club}



Can you help inspire the next generation of coders?

Code Club is a nationwide network of volunteer-led after school clubs for children aged 9-11.

We're always looking for people with coding skills to volunteer to run a club at their local primary school, library or community centre for an hour a week.

You can team up with colleagues, a teacher will be there to support you and we provide all the materials you'll need to help get children excited about digital making.

There are loads of ways to get involved!

So to find out more, join us at **www.codeclub.org.uk**

This advertising space was kindly donated to Code Club by Linux Voice

LINUX VOICE

TUTORIALS

Dip your toe into a pool full of Linux knowledge with eight tutorials lovingly crafted to expand your Linux consciousness

Ben Everard
His power consumption has already drowned five polar bears. Bring on climate change!

I run five machines for my personal computing needs (a little excessive I know), and between them, they have 10 different operating systems installed. Nine of these are Linux distros, but one is Windows 7. That Windows system now has the option to upgrade to Windows 10, and so I find myself facing the eternal question: to upgrade or not to upgrade. Usually, the decision focuses on which software is available, how stable I expect it to be and other technical considerations. This time it's different. The two questions I'm pondering are: will it wipe the other OSes on the same machine, and what are the new terms of use?

The answer to the first one is a slightly annoying yes, but that's easily fixed with some backups. The answer to the second one is quite shocking. The Privacy agreement for Windows 10 is 22 pages long, and in those 22 pages, Microsoft outline just how much they'll hoover up from your device and, if they choose, share with unnamed partners.

I don't consider myself a particularly paranoid person, but I'm not comfortable agreeing to give up all my information to a company that is has performed some morally dubious business manoeuvres in the past. For now, I'll stick with Windows 7 and retain what little of my privacy I have left.
ben@linuxvoice.com

In this issue...

LINUX VOICE TUTORIAL

KEEPASSX: CREATE A SHARED PASSWORD DATABASE

GRAHAM MORRISON

Keep all your secrets safe in one place that can then be shared safely wherever you need access.

WHY DO THIS?
- Remember one password for everything
- Encrypt all kinds of data and access credentials
- Synchronise those details to phones and other machines
- Secure password generation

Passwords aren't exciting unless someone guesses yours. In the internet age, your passwords are the keys to both your online and, increasingly, offline life, whether that's your Facebook profile, your bank or your one-click enabled Amazon Prime account. The solution is to make every password you use complex, and every password you use different. But that's hard to do, so so we recommend a password manager called *KeePassX*.

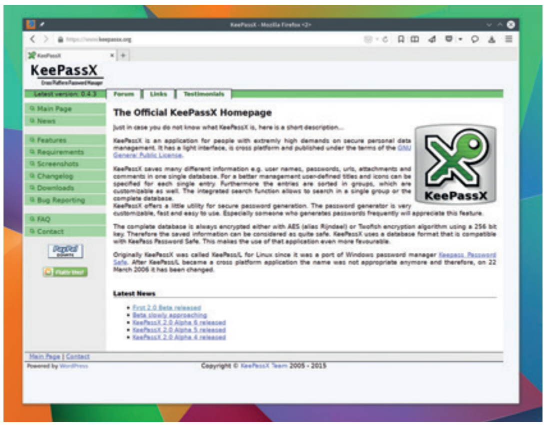
KeePassX is really just a secure database with a simple front-end for data entry. But it's been fine-tuned to store lots of different personal data, including usernames, passwords, files and notes and stores these in a single secure database, and the database itself is stored using AES 256-bit encryption, so it's secure even if someone else gets the file. There are versions for all the popular operating systems, which means that if you can find a way of synchronising the database file across all your devices, you'll have an always up-to-date password manager, regardless of whether you're using your phone or your laptop. And that's what we're going to do in this tutorial.

Step by step: Install and configure Google Authenticator

1 Installation

At the time of writing, beta versions of *KeePassX 2* have started to appear. Version 2 is a significant upgrade on version 1, making its installation over the previous versions worth the slight stability risk. There's also a good chance that 2.0 will be available by the time this tutorial escapes Linux Voice Towers, so there may be no downside.

It also uses a different database format to previous versions but remains compatible with the majority of tools and apps built to talk to the original *KeePass* 1 and 2 database formats, so you'll still be able to access your passwords from Windows, OS X and Android. Installation is obviously going to depend on your distribution, especially if *KeePassX* is still in beta, but there's a PPA for Ubuntu derivatives, and version 2 is in Arch's AUR repository. When first launched, there really is nothing to see. It's not a sign that anything is broken, it's just that *KeePassX* needs a database from which to work.



The Official KeePassX Homepage

Just in case you do not know what KeePassX is, here is a short description...

KeePassX is an application for people with extremly high demands on secure personal data management. It has a light interface, is cross platform and published under the terms of the GNU General Public License.

KeePassX saves many different information e.g. user names, passwords, urls, attachments and comments in one single database. For a better management user-defined titles and icons can be specified for each single entry. Furthermore the entries are sorted in groups, which are customizable as well. The integrated search function allows to search in a single group or the complete database.
KeePassX offers a little utility for secure password generation. The password generator is very customizable, fast and easy to use. Especially someone who generates passwords frequently will appreciate this feature.

The complete database is always encrypted either with AES (alias Rijndael) or Twofish encryption algorithm using a 256 bit key. Therefore the saved information can be considered as quite safe. KeePassX uses a database format that is compatible with KeePass Password Safe. This makes the use of that application even more favourable.

Originally KeePassX was called KeePass/L for Linux since it was a port of Windows password manager Keepass Password Safe. After KeePass/L became a cross platform application the name was not appropriate anymore and therefore, on 22 March 2006 it has been changed.

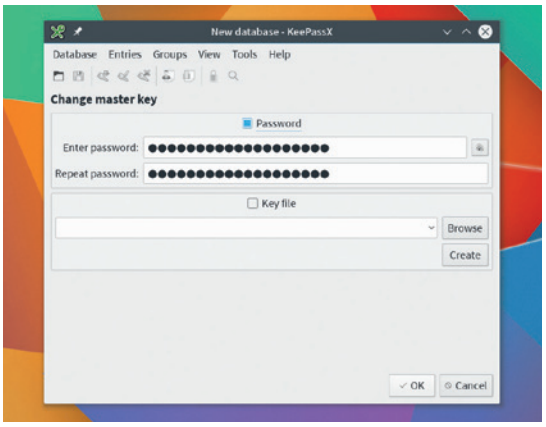
Latest News
- First 2.0 Beta released
- Beta slowly approaching
- KeePassX 2.0 Alpha 6 released
- KeePassX 2.0 Alpha 5 released
- KeePassX 2.0 Alpha 4 released

Main Page | Contact
Powered by WordPress
Copyright © KeePassX Team 2005 - 2015

2 Create the database

Click on 'New' from the Database menu. This opens the 'Change Master Key' pane. This is where you need to enter a password that's going to be used to encrypt your database. It's going to need to be strong and unique. The good news is that from now on, it's going to be the only password you'll need to remember. You also have the option of using a key file. This file can be anything, but because it's used to encrypt your data, the more complex the data in the file the better encrypted your data will be.

Key files are convenient because you don't need to remember a password, but used on their own, they're also a significant security risk. If an attacker gets access to both the key file and the database, they'll be able to access your data. For that reason, key files should be considered an added level of security, alongside the password. That way, if someone gets hold of your database, even if they guess your password they won't be able to decrypt your data.



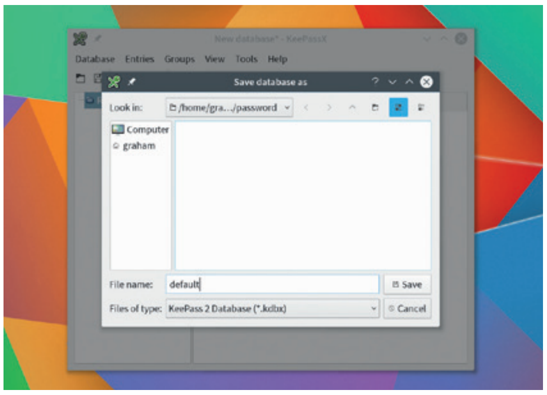
Change master key

[✓] Password
Enter password: ●●●●●●●●●●●●●●●●●●●●
Repeat password: ●●●●●●●●●●●●●●●●●●●●
[ ] Key file

3 Save the database

If you do use a key file, make sure you don't save it with the database. Keeping a key file on an external USB drive is a good option, for example, or mounted from your phone. As long as the two aren't together, your secrets will be safe.

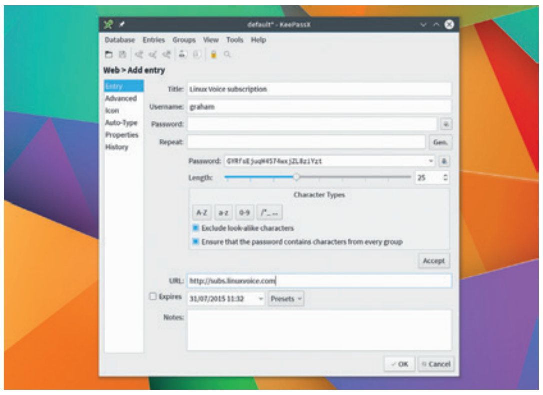
After the database has been created, you'll next need to save it. We'd recommend saving the database to a specific folder for these files. Using more than one database can be useful if you want some level of separation, perhaps between banking and browsing, for instance, but there's a tradeoff in convenience. Saving them all to the same folder makes things slightly easier to manage, and as we don't really believe in security via obscurity, represents no more of a security risk than naming your database **pagefile.sys** and hoping for the best.



Look in: /home/gra.../password
File name: default
Files of type: KeePass 2 Database (*.kdbx)

4 Populate the database

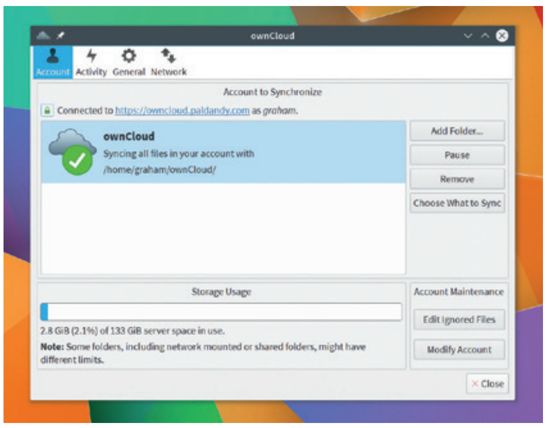
Adding your secret information is the most arduous part of the process. Groups can be used to categorise your collection of information, just as you would use folders in a filesystem. Use 'Add New Entry' for each new secret, and there are lots of different fields that can be used to store your data. You can create attributes and attach files, for instance. But going through each entry is also a good opportunity to update your passwords, especially as the 'New Entry' dialog includes an excellent password generator. Use this to create passwords that you can paste into your online password fields for maximum security. The only downside is that you need to make sure you don't lose your *KeePassX* database or passwords, as this will have a huge impact on how you access your various resources.



Web > Add entry
Title: Linux Voice subscription
Username: graham
Password:
Repeat:
Password: GYRfsEjuqH457AexjZL8ziYzt
Length: 25
✓ Exclude look-alike characters
✓ Ensure that the password contains characters from every group
URL: http://subs.linuxvoice.com
☐ Expires 31/07/2015 11:32
Notes:

5 Synchronise

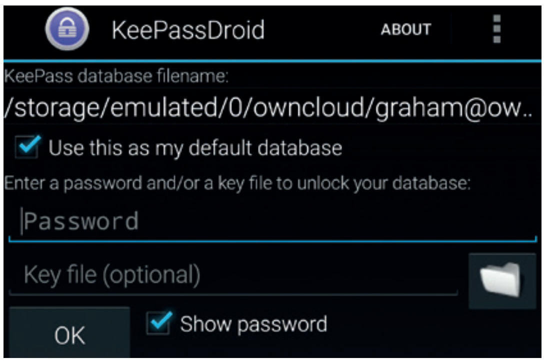
After entering your data into a single password store, you don't want to have to worry about keeping the database synchronised across your various devices. The best solution we've found is to run your own OwnCloud server and use its folder synchronisation. (Our tutorial on installing OwnCloud can be found in issue in issue 6 and online.) With OwnCloud running, you can then use a local client on your desktop or on your phone to synchronise the contents of the folder containing your *KeePassX* database. We'd also recommend enabling 'Automatically Save On Exit' and 'Automatically Save After Every Change' in the *KeePassX* options panel, so that everything is saved and synchronised as you add and edit your database.



Connected to https://owncloud.paldandy.com as graham.
ownCloud
Syncing all files in your account with /home/graham/ownCloud/
2.8 GiB (2.1%) of 133 GiB server space in use.
Note: Some folders, including network mounted or shared folders, might have different limits.

6 Android compatibility

Our preferred Android app for handling the password database is called *KeePassDroid*. It can be installed either through Google Play or the open source repositories of F-Droid. The only difficulty is that it expects a database file ending in **kdb**, so you may need to rename the file created by the desktop client. If you've synchronised the folder holding this file, when you first launch the app you can use the file navigation to access the location and load up the database and keyfile, if you've used one. You'll then need to enter the password, after which you should find all the information you've entered. For convenience, there's a menu option that lets you copy and paste usernames and passwords from each entry, and you can lock the database so it can't be accessed if someone gets hold of your phone. LV



KeePass database filename:
/storage/emulated/0/owncloud/graham@ow..
✓ Use this as my default database
Enter a password and/or a key file to unlock your database:
Password
Key file (optional)
✓ Show password

LINUX VOICE TUTORIAL

RASPBERRY PI: BUILD A CRANE WITH AN EXPLORER HAT PRO

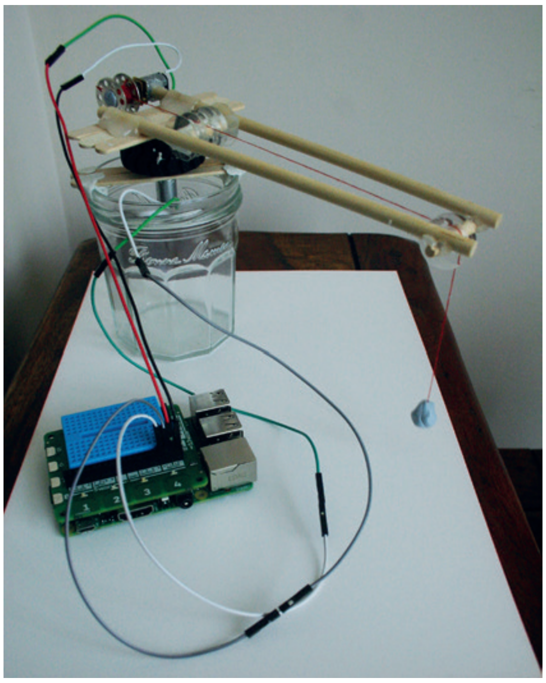
LES POUNDER

When **Les Pounder** grows up he wants to control a crane – until that day he'll have to be content with building one!

WHY DO THIS?
- Learn Python
- Control motors
- Learn engineering principles
- Hack hardware

TOOLS REQUIRED
- A Raspberry Pi Model Pi 2 or B+
- Raspbian operating system
- Explorer HAT Pro
- 2 x micro gear metal motors
- Male-to-female jumper cables
- Strong magnet
- Craft materials to make the crane

When we think of computing we naturally think of code. In movies, whenever they need to illustrate the dark art of computing there's usually line upon line of text flying up the screen as a hacker frantically types commands into a keyboard. But in recent years we have seen a growing trend of makers merging physical skills such as model making and electronics with single-board computers such as the Raspberry Pi. In schools we're seeing more cross-curricular activities where design and technology are merging with the computing department to encompass product design with a technological twist. At a recent Picademy in Exeter I had the pleasure of working with a design and technology teacher who built a basic crane using card and sticky-backed plastic. This got me thinking about how lucky we are now, with technologies such as Raspberry Pi and the Explorer HAT Pro enabling us to hack working models of engineering marvels. In this tutorial we shall construct our own crane.


The finished project – how will you tackle building your very own desktop crane?

Setting up Explorer HAT Pro

Explorer HAT Pro is the bigger brother to the successful Pibrella add-on board. To set up the Explorer HAT Pro, ensure that your Raspberry Pi is powered down and gently attach the board to the GPIO (General Purpose Input Output) pins so that the board overlaps the Raspberry Pi. Now attach all of your peripherals and ensure that your Pi has an Ethernet or Wi-Fi dongle to connect to the internet. Power up your Raspberry Pi and boot up to the desktop. From the desktop open a terminal; you can find the terminal icon in the top-left of your screen.

In the terminal type the following line by line, and at the end of each line press Enter and wait for the action to complete.

```
apt-get update
```

First we ensure that our list of software repositories is up to date, which ensures that we use the latest software available.

```
apt-get install python3-smbus
apt-get install python3-pip
```

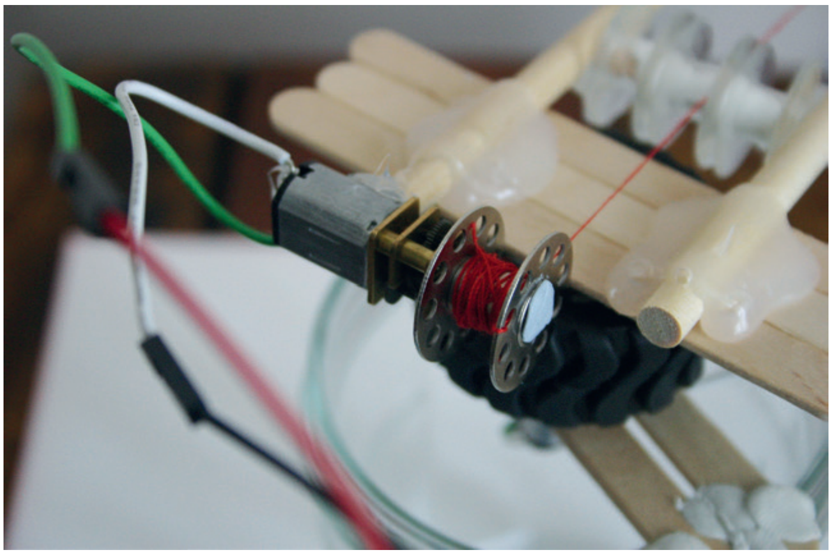
Second we install the Python 3 **smbus** package, which enables Python 3 to use the I2C interface between the Explorer HAT Pro and the Raspberry Pi. Next we install the **pip** package manager for Python 3. This works in much the same was as **apt-get** in that it downloads all the software and dependencies for a package.

```
pip-3.2 install explorerhat
```

Finally we download and install the Explorer HAT library for Python 3.

The Explorer HAT Pro has a lot of functionality, so let's get better acquainted with this board. To start using the board open a terminal and type in the

Using the micro gear metal motor is really easy with the Explorer HAT Pro. Here you can see the soldering used to connect the wires.



following command and press Enter at the end of the line. It will launch *Idle 3*, the Python 3 editor, and put the command into background freeing up the terminal for further use, if we wish.

```
sudo idle3 &
```

Lights

The Explorer HAT Pro comes with four surface mount LEDs that are above the numbers 1 to 4 on the board. They are coloured red, green, yellow and blue. Let's start by controlling the LEDs. Click on File > New Window to open a blank document.

```
import explorerhat
import time
explorerhat.light.on()
time.sleep(5)
explorerhat.light.off()
```

Save this code as **test.py** and click on Run > Run Module to test the LED on your board. At the start of the code we imported the Explorer HAT Python library (we installed this earlier using **pip**), next we instruct all four LED to turn on for five seconds and then turn off. This is known as a blink, and it is the "Hello World" of hardware hacking.

If we wanted to create a looping blink then normally we would use a loop to contain the instructions that perform the blink. But the Explorer HAT library has a special function that we can use in its place:

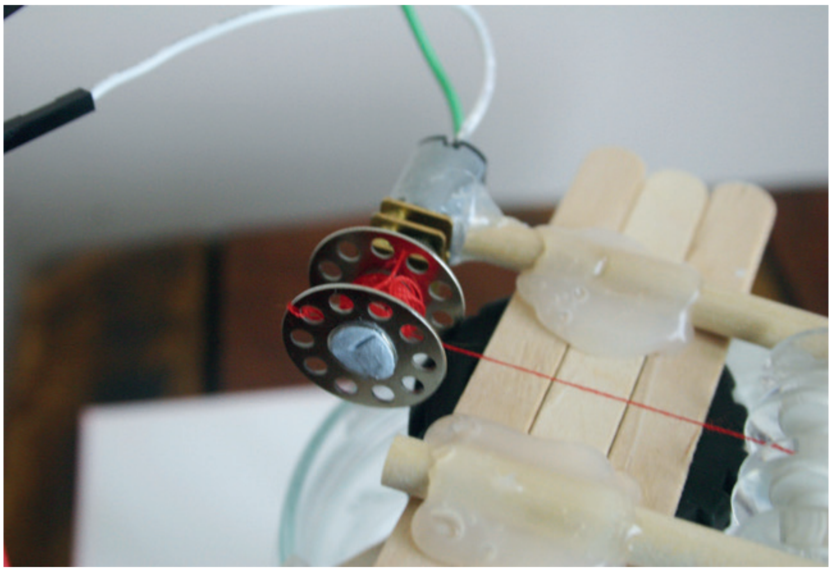
```
import explorerhat
import time
explorerhat.light.blink(0.5,0.5)
time.sleep(5)
explorerhat.light.stop()
```

So this code will blink all of the LEDs, half a second on, half a second off for five seconds before the LEDs are turned off. Save this code and click on Run > Run Module. We can also create a pulse effect using the LEDs, similar to the pulsing effect that you can see on laptops that are sleeping.

Soldering

In this tutorial we used micro gear metal motors to power the winch and control the direction of the crane, but these motors required us to solder wires for connection to the Explorer HAT Pro. Soldering is a great maker skill to learn and surprisingly it's not very hard to do. Soldering uses a hot piece of metal (an iron) to melt solder, which when cooled creates a bond between a component and whatever it is being attached to. Getting started with soldering is great fun, thanks to videos such as Carrie Anne's Geek Gurl Diaries **http://bit.ly/LV19-Solder**. Remember to solder in a well ventilated place and ensure that your workspace is clear and well laid out before turning the iron on. A great tip from Carrie Anne is to use a hot glue gun after soldering the wires to ensure they are firmly held in place.

Soldering kits come in many different price brackets but the key thing to remember is that you get what you pay for, so a cheap kit may look attractive to start with but you will soon grow out of it. Our preference is the Antex range of irons, in particular the XS25, which is the right balance of quality and cost **www.antex.co.uk/home**.


We used lolly sticks and a hot glue gun to build our crane, as it enabled us to quickly prototype an idea and test *in situ*.

```
import explorerhat
import time
explorerhat.light.pulse(1,1,1,1)
time.sleep(5)
explorerhat.light.stop()
```

So in this code we changed the blink function for pulse and passed four arguments, conditions for the function to follow, which controlled the fade in time, fade out time, on time and off time for the LED. Save the code and click on Run > Run Module to test.

All of the code created thus far can also be used on each of the LED individually.

```
import explorerhat
import time
explorerhat.light.red.pulse(1,1,1,1)
time.sleep(5)
explorerhat.light.stop()
```

This repeats the pulsing code we created earlier but now only the red LED will pulse. Save and run the code to see it in action.

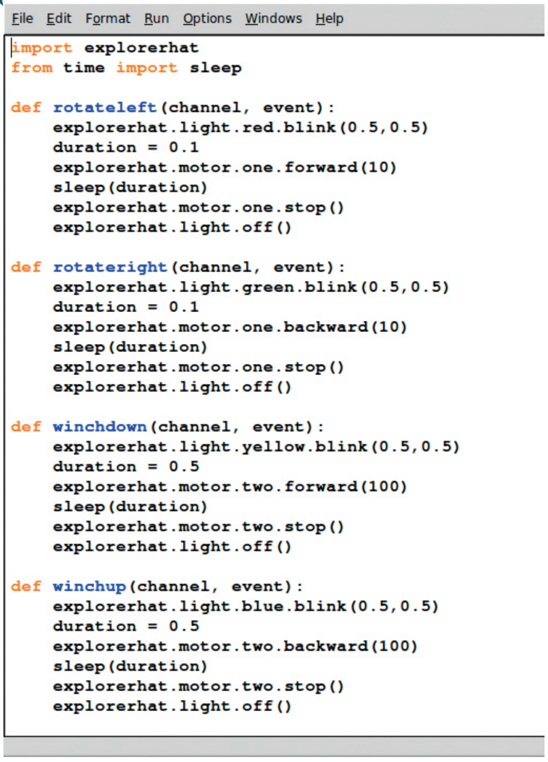
Motors

Our next test is to learn how to use the built-in motor controller. Unlike the Pibrella, which had a basic motor controller only capable of movement in one direction, the Explorer HAT Pro features an H bridge, which enables a motor to run bi-directionally. To test you will need two micro gear metal motors with wires that have been soldered correctly – see the boxout for details on motors and soldering. Connect the wires for one of your motors to motor 1 and the other to motor 2. For each motor you will have one wire in "+" and the other in "-".

Now we shall write some code in *Idle* to test the motors. You'll need to open a new blank document (File > New Window).

```
import explorerhat
import time
```

Running the code via *Idle 3* we can check the output of the code and spot any errors quickly.



```
import explorerhat
from time import sleep

def rotateleft(channel, event):
    explorerhat.light.red.blink(0.5,0.5)
    duration = 0.1
    explorerhat.motor.one.forward(10)
    sleep(duration)
    explorerhat.motor.one.stop()
    explorerhat.light.off()

def rotateright(channel, event):
    explorerhat.light.green.blink(0.5,0.5)
    duration = 0.1
    explorerhat.motor.one.backward(10)
    sleep(duration)
    explorerhat.motor.one.stop()
    explorerhat.light.off()

def winchdown(channel, event):
    explorerhat.light.yellow.blink(0.5,0.5)
    duration = 0.5
    explorerhat.motor.two.forward(100)
    sleep(duration)
    explorerhat.motor.two.stop()
    explorerhat.light.off()

def winchup(channel, event):
    explorerhat.light.blue.blink(0.5,0.5)
    duration = 0.5
    explorerhat.motor.two.backward(100)
    sleep(duration)
    explorerhat.motor.two.stop()
    explorerhat.light.off()
```

```
explorerhat.motor.one.forward(100)
explorerhat.motor.two.forward(100)
sleep(5)
explorerhat.motor.one.stop()
explorerhat.motor.two.stop()
```

In this code snippet we import the Explorer HAT and **time** libraries and then instruct both motors to go forwards at full power **(100)**. The code then sleeps for five seconds before we stop both the motors. Save this code as **motor.py** and click on Run > Run Module to test. Both of your motors will spring into life.

You saw in the motor forward function that we can pass an argument to the function to control the speed, which gives our motors an element of precision. To see this in action we shall use a **for** loop.

```
import explorerhat
import time
for i in range(0,110,10):
    explorerhat.motor.one.forward(i)
    explorerhat.motor.two.forward(i)
    sleep(0.5)
explorerhat.motor.one.stop()
explorerhat.motor.two.stop()
```

Here we use a **for** loop to iterate between 0 and 110 in steps of 10; this gives us the value **i**, so for the first time the loop is run the value is 0, then 10, 20, 30 and so on. This value is used to control the speed of our motor. We then sleep for half a second before repeating the loop. Once we reach the maximum range of the loop, it ends and both of the motors are turned off. Save this code and run it to see the motors gradually speed up.

Touch input

Our last test covers the 8 touch inputs that are on the board. Numbers 1–4 are capacitive touch buttons and register a touch; numbers 5–8 work in the same way but we can also attach crocodile clips to each of the inputs and use conductive materials, such as tin foil and fruit, as inputs.

```
import explorerhat
from time import sleep
def lights(channel, event):
    explorerhat.light.on()
    time.sleep(5)
    explorerhat.light.off()
explorerhat.touch.one.pressed(lights)
```

In this code snippet we import the libraries as before, but then we create a function called **lights** that has two arguments, **channel** and **event**, which are used by the Explorer HAT to identify the input and how it was triggered (long press, short etc). In the function we instruct the code to turn all the lights on for five seconds before turning them off. Save this code as **touchtest.py** and click on Run > Run Module to test. Press the number 1 button and all of the lights will come on for five seconds.

Putting it all together

So we have tested all of the functionality for our crane.

- Touch input to control the crane.
- Lights to warn users that the crane is in operation.
- Motors to control the rotation of the crane and for the grabber.

Now we need to put it all together.

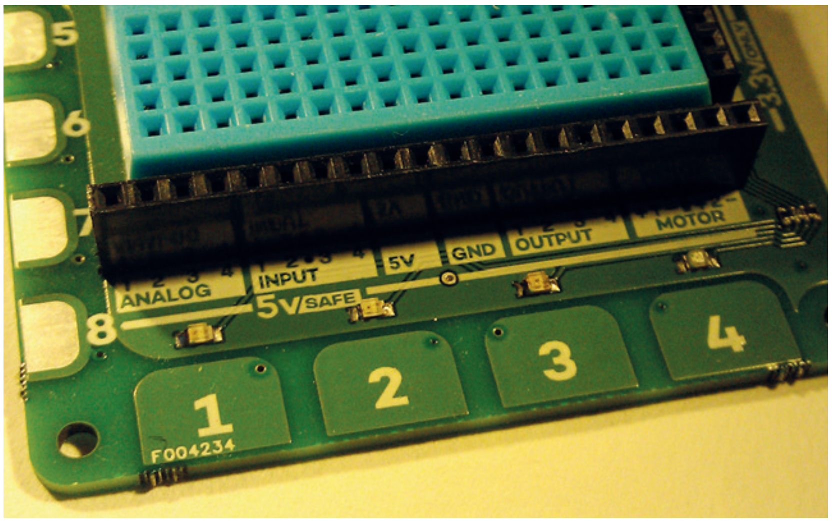
In *Idle* open a new file and save it as **crane.py**.

```
import explorerhat
from time import sleep
```

We start the code by importing the Explorer HAT library and importing the **sleep** function from the **time** library.

```
def rotateleft(channel, event):
    explorerhat.light.red.blink(0.5,0.5)
    duration = 0.5
    explorerhat.motor.one.forward(XX)
    sleep(duration)
    explorerhat.motor.one.stop()
    explorerhat.light.off()
```

The Explorer HAT Pro is a powerful platform for invention. The mix of motor controller and capacitive touch interface sparks many project ideas.



Next we create a function to handle rotating the horizontal motor that controls the crane's direction. The function starts by blinking the red LED, and then we use a variable, **duration**, to set the length of time that the motor is on. Next, motor one is set forward at **XX** percent, then we sleep for the length of the **duration** variable before turning the motor and red LED off.

```
def rotateright(channel, event):
  explorerhat.light.green.blink(0.5,0.5)
... #All other code is as per rotateleft function.
  explorerhat.motor.one.backward(xxx)
```

To rotate the crane to the right we use a similar function to before, but this time we call it **rotateright** and make two further changes illustrated in the code above. This time we flash the green LED, and motor one is run backwards.With the motor used to control the horizontal direction of our crane configured, our attention now turns to the motor that will be used to power the winch that lowers our grabber.

```
def winchdown(channel, event):
  explorerhat.light.yellow.blink(0.5,0.5)
  duration = 0.5
  explorerhat.motor.two.forward(XX)
  sleep(duration)
  explorerhat.motor.two.stop()
  explorerhat.light.off()
```

Again we reuse the code that powers our two previous functions but we make a couple of changes. This function is called **winchdown**, and it lowers the grabber down to its target. When this happens the yellow LED blinks and motor two is driven forward at **XX** percent.

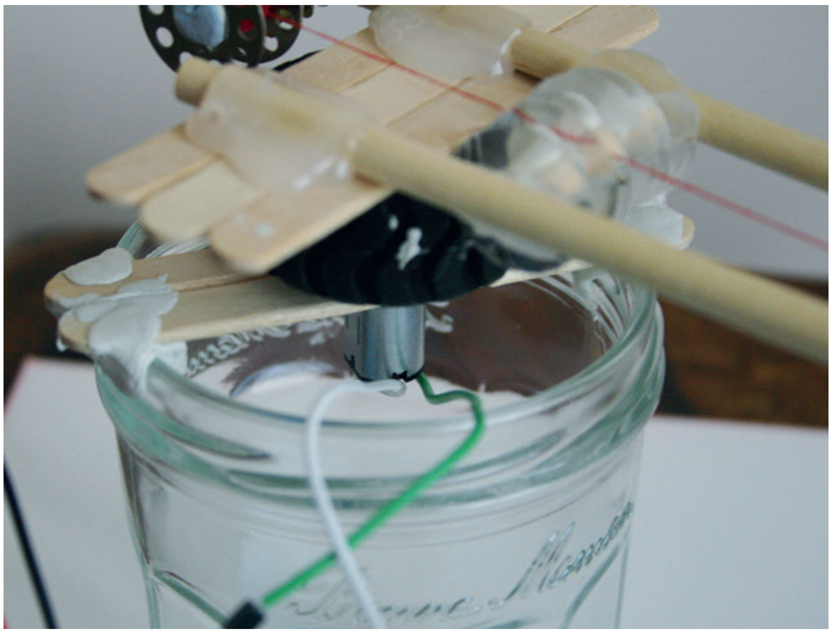
```
def winchup(channel, event):
  explorerhat.light.blue.blink(0.5,0.5)
... #All other code is as per winchdown function.
  explorerhat.motor.two.backward(100)
```

To pull the grabber back towards the crane we first blink the blue LED and then run motor two in reverse.

Motors

The Raspberry Pi has seen its fair share of robotics projects and as such it has also seen many motors. In this tutorial we used the micro gear metal motors from **Pimoroni.com**, as they offered the best power for their size. There are other motors on offer, and a quick search of eBay will show the many on offer. The motors that come with the RyanTeck Budget Robotics kit are much larger and faster than those used in the tutorial but they can be easily controlled via the Explorer HAT Pro, as they also use 5V.

The Explorer HAT Pro uses a motor controller with an H bridge, which is a circuit that enables a voltage to be applied in either direction, or in simpler terms the direction of the motor can be reversed without having to rewire the circuit. This makes the Explorer HAT Pro ideal for robotics as the robot can spin on the spot. Also in the tutorial we specified the speed of the motor using a percentage, this uses PWM (Pulse Width Modulation) to achieve analog precision using digital signals. This gives our project precise control for the speed of the motors, essential when trying to accurately control the direction and position of the two motors.


We used an old jam jar to give us a solid base for our crane but in the spirit of hardware hacking you can use whatever fulfills your design.

We now move from creating functions to the main body of code, which handles the user input and triggers the functions that we have created.

```
explorerhat.touch.one.pressed(rotateleft)
explorerhat.touch.two.pressed(rotateright)
explorerhat.touch.three.pressed(winchdown)
explorerhat.touch.four.pressed(winchup)
```

We will use the buttons for 1, 2, 3, 4 and 8, and for each of these buttons we'll assign a function as per the code above. Buttons 1 and 2 handle the direction of the crane. Buttons 3 and 4 handle the winch, and button 5 is used to toggle the magnetic grabber.

Save your code and ensure that everything is connected correctly (it is wise to keep the magnet away from your Raspberry Pi and any storage media!). When you're ready, click on Run > Run Module and test each of the inputs in sequence.

"We've used coding, electronics and engineering practices to build our very own model crane."

Crane design

Congratulations, you have used coding, electronics and engineering practices to construct your very own model crane. The design of our crane is entirely based on research via web searches. Your crane can look entirely different to ours and it would be great of you to show us your constructions by tweeting us at Linux Voice!

All of the code for this project can be found at our GitHub repository: **https://github.com/lesp/LV_19_Education** and you can download the project as a ZIP file from **https://github.com/lesp/LV_19_Education/archive/master.zip**.

Les Pounder divides his time between tinkering with hardware and travelling the United Kingdom training teachers in the new IT curriculum.

LINUX VOICE TUTORIAL

WORDPRESS: SET UP A SIMPLE BLOGGING ENGINE

MARCO FIORETTI

Most of the time the best solution is the simplest – so here's the simplest way to set up a dynamic-content website (aka a blog!).

WHY DO THIS?

- If you want your own blog, this is the best way to do it. It's fun, it's easy and it's really yours. A man's *WordPress* blog is his online castle.
- Publishing with *WordPress* is quick, even while you are browsing the web, and can be partially automated
- It's a skill that may turn useful on the job, someday. Look at us at Linux Voice!

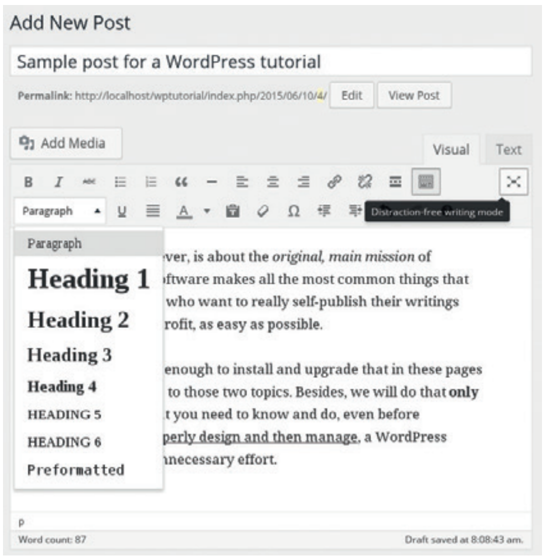
Being the master of your own online space, and managing it with Free Software however you want, wherever you want, may be less fashionable and convenient than squatting in some corner of the walled garden *du jour* (yesterday MySpace, today Facebook, tomorrow – who cares?) but believe us: it feels really good. It may also be your small, but still necessary contribution to keep the whole web as open as possible. Of course, there are many ways of doing this.

One of these solutions is *WordPress* (**http://wordpress.org**), the Free Software behind the **wordpress.com** blogging platform that you can run yourself on any web space with PHP support and access to a *MySQL* or *MariaDB* database.

WordPress is not only for personal websites. The best proof we can give you of this fact is that **www.linuxvoice.com** itself, where many authors and editors work together, thanks to *WordPress 4*.

This tutorial, however, is about the original, main mission of *WordPress*: make the most common things that 90% of individuals who want to really self-publish their writings online as easy as possible.

WordPress 4 is easy enough to install and upgrade that in these pages we give as little space as we can to those two topics. We prefer to explain what you need to know and do, especially before installation, to properly design and manage a *WordPress* website without unnecessary effort.


Add New Post

Sample post for a WordPress tutorial

Permalink: http://localhost/wptutorial/index.php/2015/06/10/4/ Edit View Post

Add Media Visual Text

Paragraph

Distraction-free writing mode

Paragraph
Heading 1
Heading 2
Heading 3
Heading 4
HEADING 5
HEADING 6
Preformatted

ver, is about the *original, main mission* of
ftware makes all the most common things that
who want to really self-publish their writings
rofit, as easy as possible.

enough to install and upgrade that in these pages
to those two topics. Besides, we will do that **only**
t you need to know and do, even before
perly design and then manage, a WordPress
nnecessary effort.

p

Word count: 87 Draft saved at 8:08:43 am.

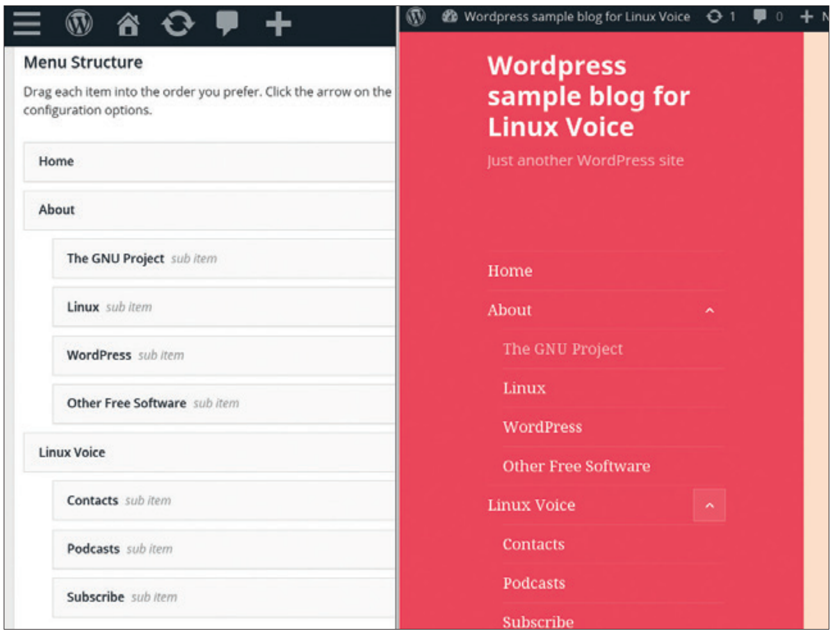
Fig 1: We like old text editors as much as the next Linux fan, but the *WordPress* editor (only partially visible here) is hard to beat when it comes to user-friendliness.

Main functions and concepts

The internal editor of *WordPress* is shown in figure 1. Clicking on the "Distraction-Free" button on the right makes everything disappear except the actual text box and the toolbar above it. It's not as flexible as plain text editing with MarkDown or a similar system but it's as easy as it's possible to get without having to dumb yourself down to Facebook-level "posting". Just try clicking on all the buttons and then on Preview to see how it works.

The other publishing options of *WordPress* are just as powerful as they are easy to miss. You can publish posts remotely with scripts or desktop tools (see the Installation And Updates box), or activate the Post Via Email function in the Writing panel.

You can also go from looking at any web page to writing a post about it in your blog in one click. Go to the Tools panel and drag the Press This button in the bookmarks bar of your browser. Then, whenever you find some inspiring paragraph in a web page, select it and click on that button: you'll get a simplified version of the *WordPress* editor, with that text and a link to the source already pasted in!


Menu Structure

Drag each item into the order you prefer. Click the arrow on the configuration options.

Home
About
The GNU Project *sub item*
Linux *sub item*
WordPress *sub item*
Other Free Software *sub item*
Linux Voice
Contacts *sub item*
Podcasts *sub item*
Subscribe *sub item*

Wordpress sample blog for Linux Voice

Wordpress sample blog for Linux Voice

Just another WordPress site

Home
About
The GNU Project
Linux
WordPress
Other Free Software
Linux Voice
Contacts
Podcasts
Subscribe

Fig 2: *WordPress* pages and categories can be easily dragged and dropped (on the left) to build menus. How they actually look to visitors depends on which theme you install.

By default, what *WordPress* creates when you click on the Publish button is a standard "post": a piece of content to list in the home page, with all the others of the same type, in reverse chronological order. The URLs of all posts have a common pattern, which you can customise in the "Permalinks" section of the dashboard.

Categories and Tags

Each post can have both Categories and Tags attached to it. They help your readers and search engines to immediately understand what each post is about. Categories – which ideally should all be defined before you start blogging – should correspond to the main sections of your site. A blog on tourism, for example, may have Categories like "Backpacking", "Cruises", "B&B" and so on. Categories can also be arranged hierarchically.

Tags are completely free labels that you can create and attach as you please to each single post, to describe its content in more detail. A post in the "Cruises" category above may get tags like "Caribbean", "Family-Friendly" and "Low Cost".

One way to explain all this is that Categories are the Table of Contents of your blog, and Tags its Index: adding new tags with each new post is no problem, but if you find yourself doing the same with Categories, it may mean that you had not structured your blog well when you started it.

Beside posts, *WordPress* can create several other types of content, but here we're only looking at its Pages, which are meant for high-level, relatively static information like "About" or "Privacy Policy". Pages and Categories can be arranged in Menus, as in figure 2. Its left half shows the *WordPress* dashboard panel where you drag and drop page titles to build a menu, and the right-hand side shows what the result looks like in the default theme.

Another essential component of *WordPress* is its collection of widgets: little frames, each encasing one piece of constant or dynamic content, that you can drag and drop in the available areas of the chosen theme. Figure 3 shows some of the default widgets in *WordPress*, but you can create your own ones, with custom HTML code.

"Themes are the graphic and PHP files that define every detail of the layout and visual style."

We have already mentioned themes twice, so let's explain them. Themes are the graphic and PHP files that define every detail of the layout and visual style of your blog. Switching themes takes only a few clicks, but can completely alter what *WordPress* looks like. You can see what we mean in figure 4, which is the dashboard "Theme Preview" pane: the blog is the same as in figure 3, but the visual elements (colours, fonts, backgrounds) and the layout of menus and widgets, are changed.

Finally, there are plugins – optional bunches of code to perform any function you may imagine. Half the rest of this tutorial describes just how you can manage several key aspects of a *WordPress* website with the right plugins.

LV PRO TIP

Be prepared for the worst! Go to the PHPMyAdmin demo at **http://demo.phpmyadmin.net/master-config** to get acquainted with the *MySQL* administration tool that can save your *WordPress* installation should anything go wrong.

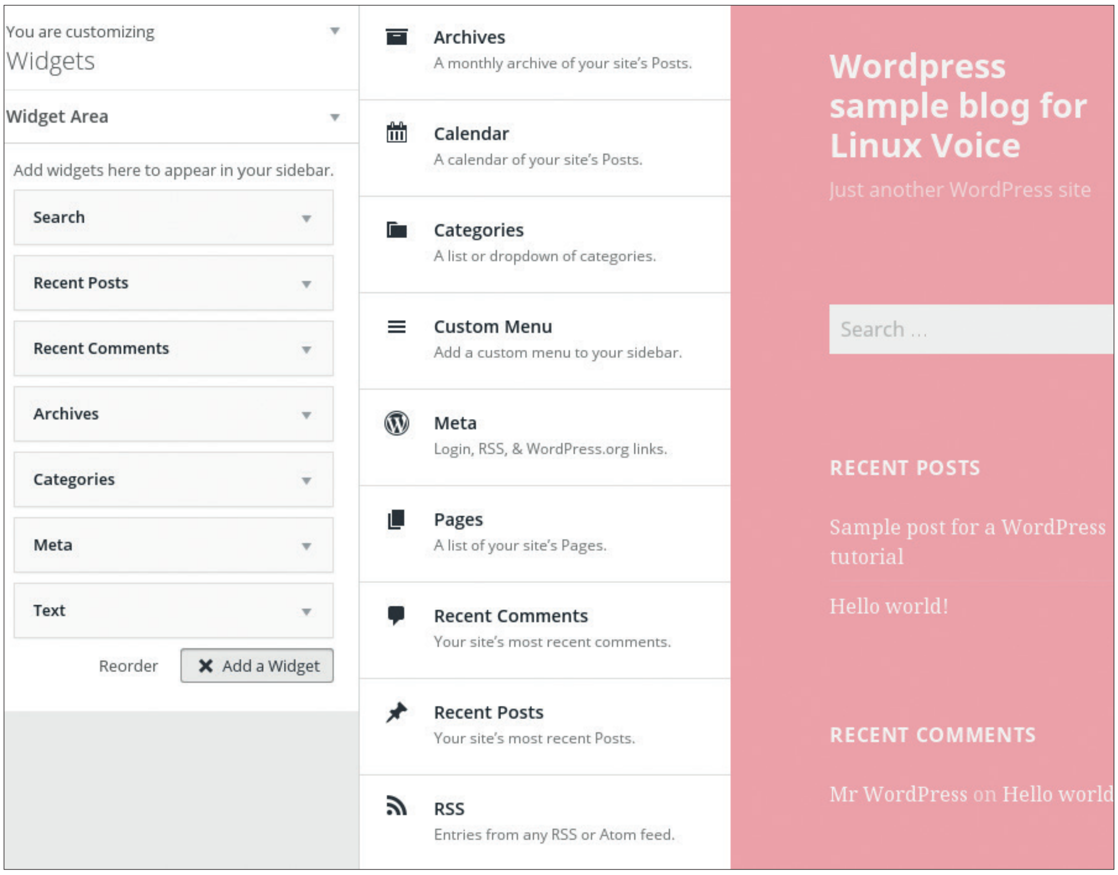
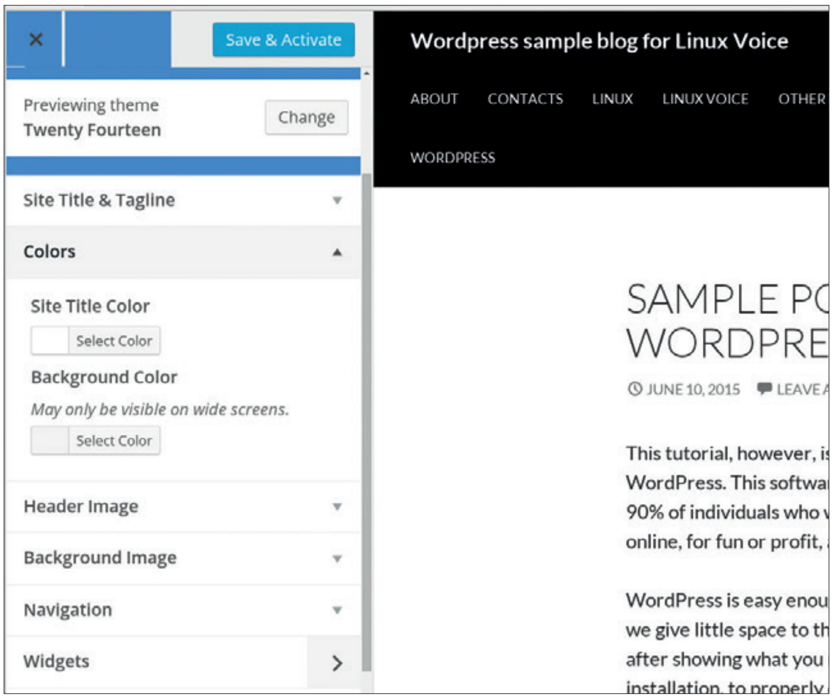

You are customizing
Widgets
Widget Area
Add widgets here to appear in your sidebar.
Search
Recent Posts
Recent Comments
Archives
Categories
Meta
Text
Reorder
Add a Widget
Archives
A monthly archive of your site's Posts.
Calendar
A calendar of your site's Posts.
Categories
A list or dropdown of categories.
Custom Menu
Add a custom menu to your sidebar.
Meta
Login, RSS, & WordPress.org links.
Pages
A list of your site's Pages.
Recent Comments
Your site's most recent comments.
Recent Posts
Your site's most recent Posts.
RSS
Entries from any RSS or Atom feed.
Wordpress sample blog for Linux Voice
Just another WordPress site
Search ...
RECENT POSTS
Sample post for a WordPress tutorial
Hello world!
RECENT COMMENTS
Mr WordPress on Hello world

Fig 3: *WordPress* widgets are little, single-purpose blocks of content that you can place in several areas of a *WordPress* blog, with an interface very similar to that used to build menus.



This tutorial, however, is
WordPress. This softwa
90% of individuals who v
online, for fun or profit,

WordPress is easy enou
we give little space to th
after showing what you
installation, to properly

Fig 4: Compare the right half of this picture with the same part of figure 3 to see how merely switching theme can really change the appearance of *WordPress*.

The last general thing to know about *WordPress* is what it cannot do as well as other website management software, for example *Drupal*. To begin with, running more, independent sites off the same *WordPress* installation is not impossible, strictly speaking, but we frankly don't recommend it. The same is true for sites that should be 100% multilingual. In both cases, you are likely to find both structural limitations that are hard to cope with, as well as lots of plugins and themes that simply aren't designed to work in such environments. If you really need to handle several independent *WordPress* blogs, install one database and a copy of the software for each of them. It's inelegant, yes, but it also is much less troublesome than the alternatives.

"WordPress can run its own commenting system or embed the Disqus platform."

One other thing that is more difficult in *WordPress* than, say, *Drupal*, is relocating an existing blog to a different URL. Check the online documentation about the **WP_SITEURL**, and **WP_HOME** *WordPress* variables for details.

What can you do with WordPress?

A lot! As we anticipated, there are enough plugins (and tutorials) around to make most of what you may want from *WordPress* pretty easy to do. In order to give you a better, though still incomplete idea, we'll look at three very different "services": image management, user engagement and SEO (Search Engine Optimisation) – how to format your content to make it show up as high as possible in online search results.

WordPress 4 can embed a simple Gallery, with "Next Image" and "Previous Image" buttons, inside a post (see figure 5): first, you upload all the images you need (Media > Add). Then, from the post editor, select "Add Media", choose and reorder from the Library all the images you want, and click on "Create Gallery". Done!

If the result looks dull, however, or if you want to generate independent galleries and/or sliders, no problem: try plugins like NextGen, Simple Photo Gallery or (especially for videos and Flash widgets) the WP Video Lightbox. Another useful image management plugin may be the EWWW Image Optimiser, which automatically reduces the size of every uploaded image, to make backups and page loads faster.

There are several ways to draw users to your blog, keep them there as long as possible and get feedback. The most "traditional", but still valid tools include post notifications, newsletters, contact forms, and good old RSS feeds. They can all be handled with plugins. The Mailpoet Newsletters plugin can help with the first two cases, and many other plugins generate forms for all uses. If the RSS feeds built into *WordPress* are too generic, try the solutions listed at **https://wordpress.org/plugins/tags/rss-feed**.

To engage with your readers in more interactive ways, you can use both comments and buttons that facilitate "sharing" what you publish on the main social networks.

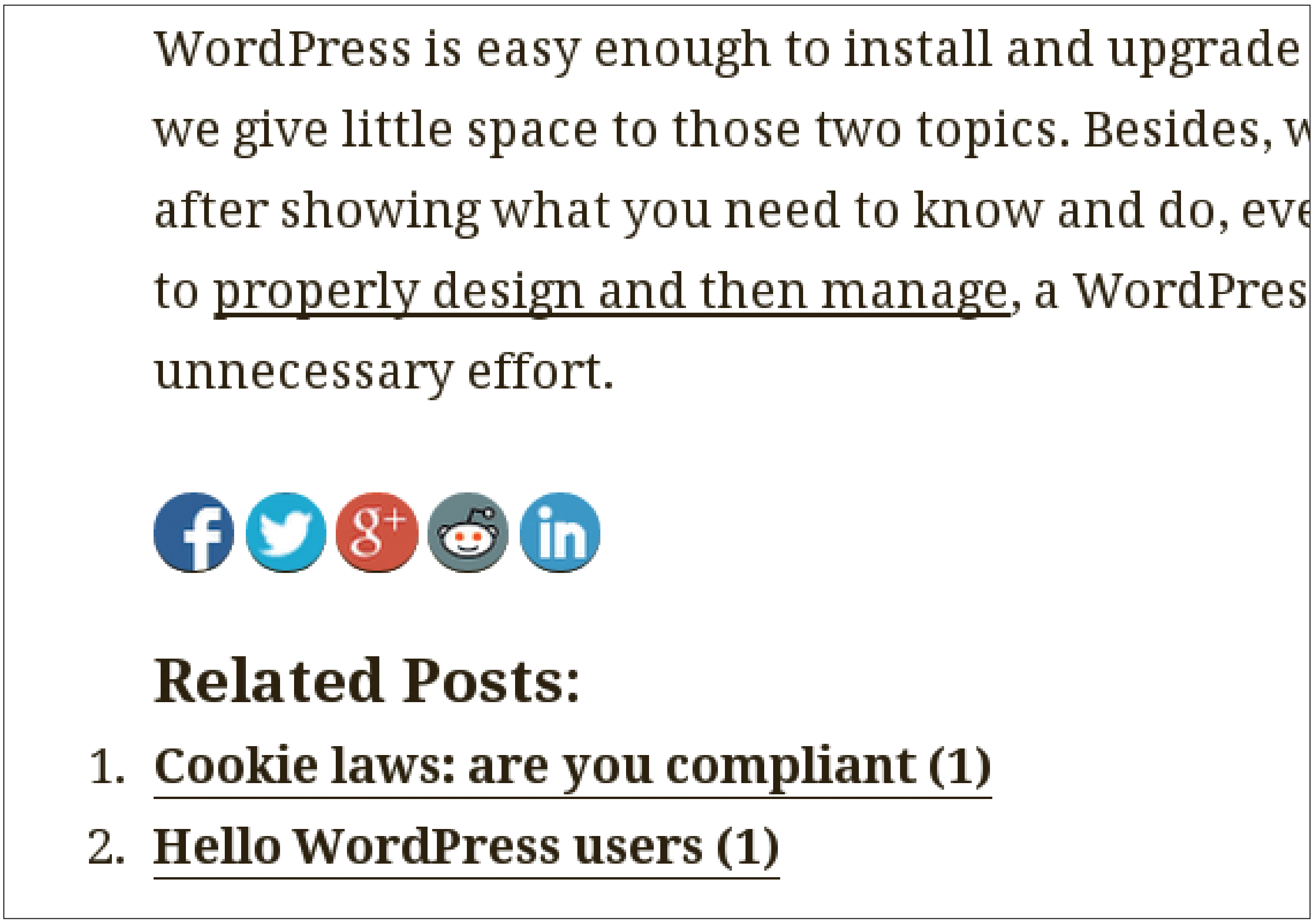
WordPress can run its own commenting system or embed (again, with a plugin) the Disqus platform. We recommend the first choice, even if it means more load for your server and, very likely, extra plugins to fight spam. Online discussion is already too centralised in too few places to contribute to that trend. On the other hand, reality says that if you don't at least make it easy to announce what you write on certain social networks, almost nobody will know about it and come to you.

Of the many "social sharing" plugins available for *WordPress* we prefer the one called "Simple Social Share". It's lightweight, simple to configure and looks good. Don't take our word for it though: what will work for you depends on what theme you choose and which networks you want to support!

A great WordPress add-on: PHPMyAdmin

To keep *WordPress* happy, its database must always be in good health. Sometimes, however, peaks of traffic, server glitches, software bugs or plain human error may corrupt that database in ways that cannot be fixed from inside *WordPress*. When this happens, there is only one solution, besides replacing the whole database with its last working backup (which will cause you to lose everything that happened after that point). First, find online what the exact problem is, then fix it directly inside the database tables.

This is much less scary than it may seem. Install *PHPMyAdmin* (**www.phpmyadmin.net**) alongside *WordPress*, and you will be able to see and change any field in the database right inside your browser. *PHPMyAdmin* will also accept raw, complex SQL commands, if manual field editing is not enough. Besides, *PHPMyAdmin* is the only way to access your database on many web hosting providers, so you'd better know how it works!


WordPress is easy enough to install and upgrade
we give little space to those two topics. Besides, w
after showing what you need to know and do, eve
to properly design and then manage, a WordPres
unnecessary effort.

Related Posts:
1. Cookie laws: are you compliant (1)
2. Hello WordPress users (1)

The YARPP and "Simple Social Share" plugins at work, adding social icons and lists of related content below each post.

Another great way to keep people on your blog is to show them content similar to what they are already reading. Try YARPP (Yet Another Related Posts Plugin) for that. Finally, you can increase the chances of appearing at the top of online searches related to your blog with plugins like Yoast or the "All in One SEO Pack". Beware though: these plugins will improve your rankings only if you take the time to add to each post the metadata they require.

Security

A blog that is only fast and good looking won't last very long, if it is an easy target for crackers. *WordPress* is reasonably robust out of the box, but there are many ways to increase its security, almost all outlined at **https://codex.wordpress.org/Hardening_WordPress**.

The available techniques can be roughly grouped according to the three levels at which they operate: web server configuration; core *WordPress* configuration and *WordPress* plugins. Of course, there's nothing to stop you mixing different techniques. Here's a simple overview, since the configuration details heavily depend on your own setup.

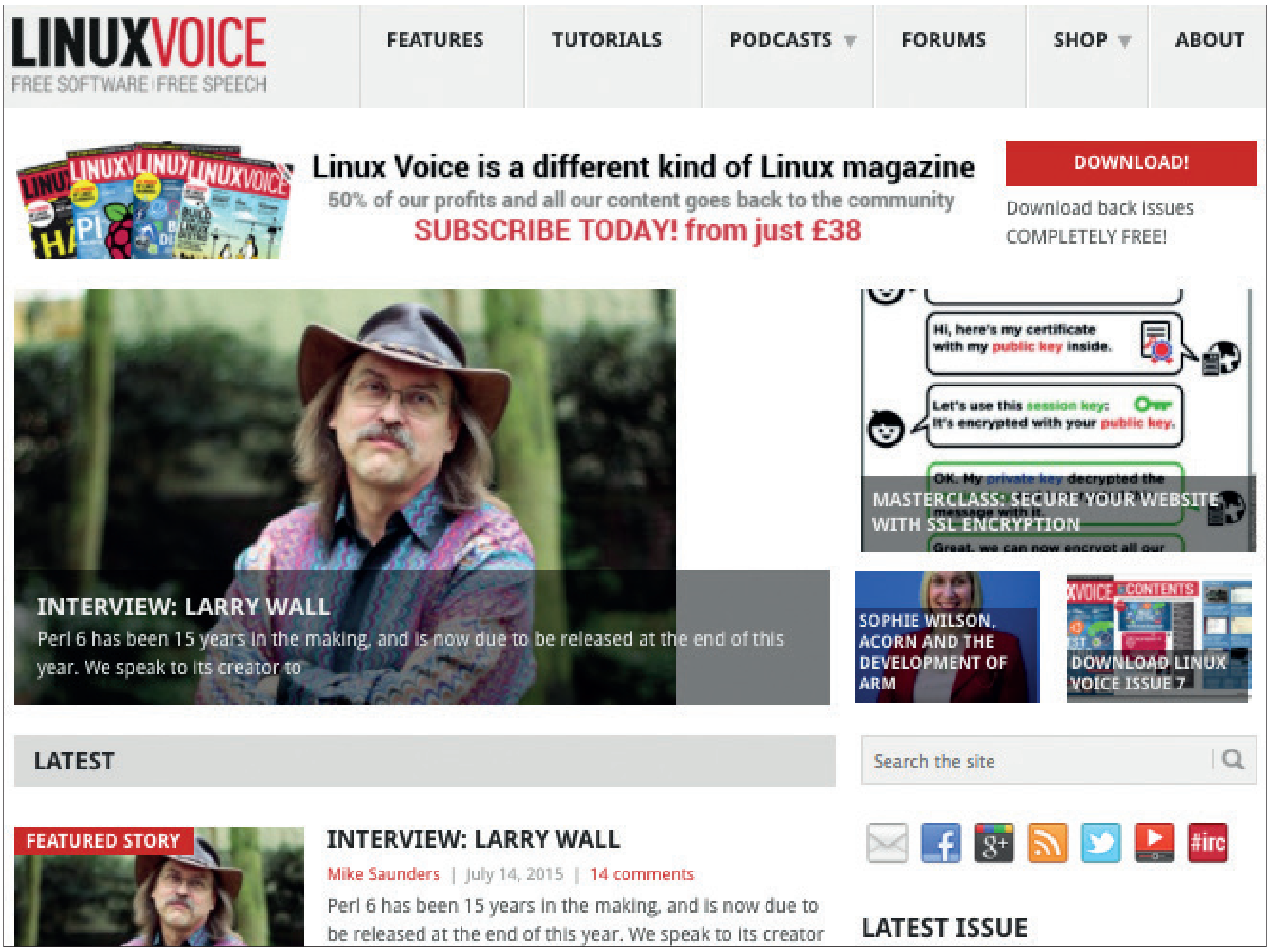
Working at the (web and/or *MySQL*) server level is very effective, but rarely possible on standard shared hosting: you need to lease a Virtual Private Server with root access to use them. The exception is setting the file permissions as described in the *WordPress* documentation, which is possible even on low-end accounts. On your own server, you may also allow access to the *WordPress* administration folder (**/wp-admin**) only from your home IP address. Alternatively, you may protect the same folder with a totally separate password, which has to be entered even if you're already logged into *WordPress* proper. You may also configure the web server to only allow secure (HTTPS) connections to that area of your blog, to avoid password sniffing.

One easy way to harden *WordPress* from inside its own dashboard is based on a principle that is very stupid but, as long as you do not rely only on it, can block many of the equally stupid, script-kiddie-level attacks: security through obscurity.

During installation, set both the first account name and the DB table prefix to something different than the standard **admin** and, respectively **wp_** values. Then, configure the user account(s) to use a nickname (what shows as post author) that is different from the username (what you have to type to log in!) and tell *WordPress* to only display the former string in all public pages. Used together, these three tricks will stop all the attacks written to exploit the default settings of *WordPress*. Plugins-wise, try BulletProof, the "All In One WP Security & Firewall" or Wordfence to (among

LV PRO TIP

Managing *WordPress* with your mouse is easy. Having your server do it for you is much better. If you have shell access to it, try the WP-CLI (**http://wp-cli.org**) command line interface to *WordPress*.



Linux Voice is a different kind of Linux magazine
50% of our profits and all our content goes back to the community
SUBSCRIBE TODAY! from just £38

Download back issues COMPLETELY FREE!

INTERVIEW: LARRY WALL
Perl 6 has been 15 years in the making, and is now due to be released at the end of this year. We speak to its creator to

MASTERCLASS: SECURE YOUR WEBSITE WITH SSL ENCRYPTION

SOPHIE WILSON, ACORN AND THE DEVELOPMENT OF ARM

DOWNLOAD LINUX VOICE ISSUE 7

FEATURED STORY

INTERVIEW: LARRY WALL
Mike Saunders | July 14, 2015 | 14 comments
Perl 6 has been 15 years in the making, and is now due to be released at the end of this year. We speak to its creator

LATEST ISSUE

The joy of *WordPress* is how tweakable it is; here's **LinuxVoice.com**, gloriously powered by *WordPress*.

other things) enforce strong passwords or block login attempts, or other attacks, after a predefined number of failures. Of course, the best plugin-related security measure is... using as few plugins as possible! Code that is not there will never be exploited (and won't slow your server down).

Optimisation

If your posts take too long to load, many people won't even start reading them. Very sad, but true. Besides, slowness may lower your SEO ratings, if you care about that. Solutions to these problems are in the same categories as those for security: web server, *WordPress* itself and *WordPress* plugins.

Reliable and high bandwidth, as well as enough RAM and CPU cycles to make web and *MySQL* servers work decently, cannot be free. If you really want a fast website, paying (within reason, of course!) for a good hosting account will do at least half of the job.

Your website may also go faster by just changing looks, if your aesthetic conscience can afford it. The default WP themes may be dull and anonymous, but they are fast and guaranteed to work with any plugin you may have installed. A more sophisticated way to reduce load times could be serving images, stylesheets, scripts and other completely static content from a "cookie-free domain". Check the *WordPress* documentation on **WP_CONTENT_URL** and **COOKIE_DOMAIN** to know how to do that.

Years ago, a very simple way to make a blog load faster was to trim its home page, setting it to only show the excerpts of a few posts, with as few widgets (including sharing buttons) as possible. These settings are less effective these days, when most visitors won't pass through your home page

"The default WP themes may be dull, but they are fast and guaranteed to work."

La cookie law in breve

Figure 5: Adding single images, or whole galleries, to a *WordPress* post is much easier than on other content management systems.

Installation and updates

The *WordPress* claim of a "5 Minute Install" is true… as long as you read, before installing, the boring but simple documentation at **https://codex.wordpress.org/Installing_WordPress** and **https://codex.wordpress.org/Editing_wp-config.php**. To prepare yourself, here's a stripped-down example of how it may work in the most complicated case, that is when you have root access to the web server. On a shared hosting account, things would be simpler. First, create, just for *WordPress* a new *MySQL* database, and a *MySQL* user with enough privileges to modify it:

```
> CREATE DATABASE wptutorial;
> GRANT ALL PRIVILEGES ON wptutorial.* TO
"linuxvoice"@"localhost" IDENTIFIED BY "golinux";
> FLUSH PRIVILEGES;
> EXIT;
```

Next, download the current version of *WordPress* from **https://wordpress.org/latest.tar.gz**, uncompress it and move the resulting folder, named **wordpress**, to what will be the blog root directory on the web server. The test blog in these screenshots, which ran at **http://localhost/wptutorial** on a Linux desktop, was created with these commands:

```
[root] cd /var/www/html
[root] wget https://wordpress.org/latest.tar.gz
[root]# tar xf latest.tar.gz
[root]# mv wordpress wptutorial
```

Next, we pointed our browser to **http://localhost/wptutorial** and followed the instructions, entering parameters like site title, name of the *WordPress* username (which is not the same as the *MySQL* account!) and *MySQL* table prefix (beware: unlike the others, this setting cannot be changed after installation!).

WordPress stores all its configurations inside one file called **wp-config.php**, which the installer attemps to create and fill with the parameters it asks you for. Should it fail to create that file because of file permission issues, it will ask you to do it manually, showing you the text to copy and paste into it. *WordPress* updates are a variant of this procedure that takes care to not remove the configuration and any other media file attached to posts. Sample scripts to partially automate updates, or post to *WordPress* from the command line, are available on the author's blog at **http://freesoftware.zona-m.net/tag/Wordpress/**.

but directly land on a post from some social network. However, applying them takes 10 seconds and cannot hurt, so go for them in "Settings > Reading".

By default, every *WordPress* post is regenerated more or less from scratch, every time somebody asks for it, by PHP code that also queries the *MySQL* database. This is a useless overload when many people try to access a post that would always look the same anyway. The usual antidotes to this behaviour are the so-called caches, that is (simplyfying!) archives of static HTML copies of all your posts.

Plugins like "W3 Total Cache" or "WP Super Cache" create just such copies and then pass them to visitors right away, instead of running the whole *WordPress* engine every time. You can get similar results playing with the web server, but caching plugins are more portable. Don't expect performance miracles, however. Their real-world effectiveness will heavily depend on how many other plugins are running, actual traffic patterns and other factors equally hard to estimate beforehand.

Other optimisation plugins that are worth trying, at least for certain blogs and themes, are WP-SmushIt and WP Minify. The first automatically reduces, as much as it's possible without visible degradation, the size of all the images you upload; the second combines and compresses, for the same reason, the JavaScript and CSS files associated to each page.

Other best practices

First, backups! Regular backups, of both the database and the whole *WordPress* folder! Again, if you can handle this at the server level, that's the best solution. Otherwise, try the BackUpWordPress plugin to schedule automatic backups.

Second, never skip an upgrade, whenever the *WordPress* dashboard tells you that it is available. Yes, it's boring, but it's much less time consuming (and risky!) doing it one step at a time, than jumping three version numbers only to discover that your database is corrupted, because nobody had tested that particular update path!

The "last" step: pre-installation planning

If reading this tutorial has left you very eager to try *WordPress* we're very happy, but before going live, remember to:

1. Figure out and write down all the functions (widgets and plugins) that you really need. The fewer, the better: your blog will run faster, need updates less frequently and be less exposed to crackers. Oh, and always check what other users say of a plugin (or theme) before considering it!
2. Using the default theme, try different combinations of site-wide settings, categories, permalinks, menus and widget layouts until you have found a structure you like. Don't waste time on looks yet.
3. Create several dummy posts, just to see what they look like.
4. Find and test, possibly both on your desktop and your smartphone, a theme that renders the structure that you just created. Again, go for the simplest one you can live with. Regardless of their licence and price, many great-looking themes for *WordPress* have the same intrinsic limit: just because they are so sophisticated and tweaked to the very last pixel, they may work as advertised only with some combinations of plugins than may not have been documented before.

If at all possible, perform the whole analysis above at home, on a throwaway blog inside your own Linux computer. When you're happy, and not one minute before, install *WordPress* from scratch on the real server, cloning that test installation.

Marco Fioretti is a Free Software advocate and open data campaigner who has advocated FOSS all over the world.

LINUXVOICE TUTORIAL

EXTEND GRUB WITH A CUSTOM MODULE

JOHN LANE

Is your favourite bootloader lacking that killer function you need? Here's how to add custom commands to Grub…

WHY DO THIS?
- Have the very latest *Grub*
- Add your own custom commands

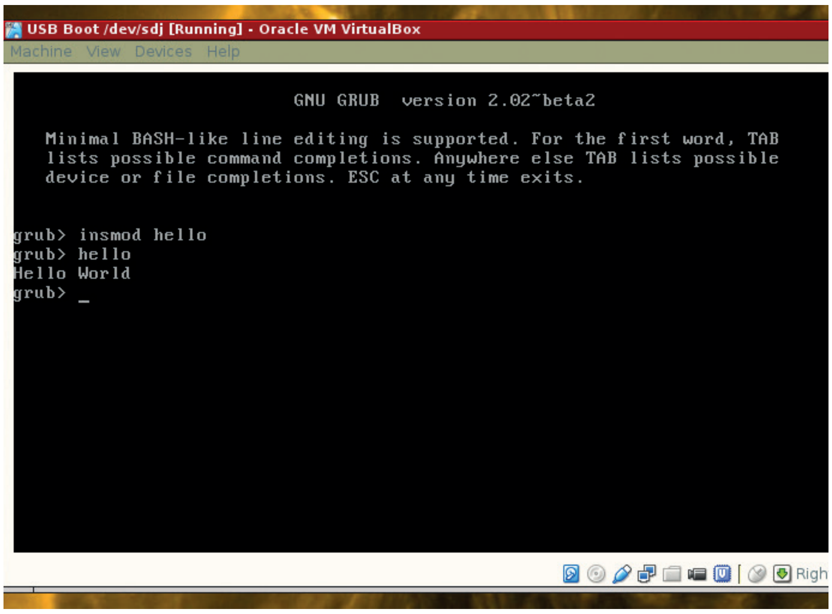
Grub is the bootloader used by most Linux distributions. It's one of those applications that many of us rely upon to deliver us safely to our login prompt. Thanks to its modular architecture, it's able to boot from a multitude of devices and disk formats. This also makes it extendable – if you find it lacks a capability that you need, you can write your own modules to extend its functionality. This month, we show you how.

Modules contain functionality that can be loaded on demand rather than being part of *Grub*'s core. The Linux kernel uses a similar concept, but don't confuse kernel and *Grub* modules – they're entirely different and have nothing in common beyond that concept.

Modules are usually loaded according to the needs of the *Grub* configuration. This uses **insmod** commands to load the modules it needs (the kernel also uses a similarly named command but, again, they aren't the same). You can also use **insmod** from an interactive *Grub* prompt to load modules and there's also **rmmod** to unload them.

"Modules are usually loaded according to the needs of the Grub configuration."

Loading a module augments *Grub*'s available commands with any new ones defined in the module.


USB Boot /dev/sdj [Running] - Oracle VM VirtualBox
Machine View Devices Help

```
                    GNU GRUB  version 2.02~beta2

   Minimal BASH-like line editing is supported. For the first word, TAB
   lists possible command completions. Anywhere else TAB lists possible
   device or file completions. ESC at any time exits.


grub> insmod hello
grub> hello
Hello World
grub> _
```

We booted *Grub* in *VirtualBox* to try out its "hello" module.

Grub modules are relocatable ELF binary object files that are usually written in C. *Grub* is a 32-bit single-threaded real-mode application that is mostly written in ANSI C with a little sprinkling of assembly language. You'll need the *GCC* compiler, along with the usual development utilities like **git** and **make**. We'll assume a basic knowledge of writing and building C applications, along with a working knowledge of **git**.

Get the latest code

Before you begin, it's well worth obtaining the latest development checkout (the master branch is updated almost daily) and familiarising yourself with its changelog in case your killer feature is already there - there's little point reinventing the wheel!

Go to a suitable working directory (we'll use **~/work** for this tutorial) and grab the latest sources from the **git** repository. You can browse at **https://git.savannah.org/cgit/grub.git** and use **git** to download:

```
$ cd ~/work # or wherever you please!
$ git clone http://git.savannah.gnu.org/r/grub.git
```

Change into the directory that this creates. You can create your own working branch if you prefer not to work on the master branch.

```
$ cd grub # our git clone
$ git checkout -b work
```

The source code needs to be prepared before it can be built. This step generates various files that the build process needs. Run the supplied script:

```
$ ./autogen.sh
```

You can build in the directory containing the prepared source code or in another one, either below or separate to it. Do either:

```
$ ./configure
$ make
```

or something like this:

```
$ mkdir ~/work/build
$ cd ~/work/build
$ ~/work/grub/configure
$ make
```

The default install destination is **/usr/local**, but you can install to a different location if you prefer. Either install to the default location (use **sudo** to gain the permissions to write there):

```
$ sudo make install
```

or somewhere else; perhaps a **run** subdirectory:

```
$ make DESTDIR=~/work/run install
```

The **DESTDIR** is used by GNU makefiles and must be given as an absolute path.

You should check that your custom install is used, and perhaps try it out (we describe some test environments in the boxout, right).

The usual programming introduction is to write something that outputs a greeting and our first *Grub* module need be no different. In fact, we don't even need to write it, because such an example is provided for us as our springboard into *Grub* module writing.

It's built along with the rest of *Grub* so you can try it out the next time you boot your computer. Open a *Grub* prompt (press C at the menu) and do the following:

```
grub> insmod hello
grub> hello
Hello World
```

The source code for each *Grub* module is located within its own subdirectory of the **grub-core** directory in our **git** clone. Take a look at the source for the **hello** module at **grub-core/hello/hello.c**. We've also printed it in the box on page 92 so you can refer to it quickly.

The **hello** module implements a single **hello** command. Its code is pretty straightforward: it includes some headers, declares its licence and implements the new command in the **grub_cmd_hello** function.

More Grub

There are two more functions that register (**GRUB_MOD_INIT**) and unregister (**GRUB_MOD_FINI**) the module. The upper-case names reflect the fact that these function definitions use macros (they're defined in **include/grub/dl.h**); a lot of *Grub* functions use macros to simplify their usage.

When the module is installed (eg by **insmod**), the **GRUB_MOD_INIT** function calls **grub_register_extcmd** to register the new **hello** command. This returns a pointer (**cmd**) that's used when **GRUB_MOD_FINI** calls **grub_unregister_command** to unregister the module.

The **grub_register_extcmd** function takes a number of arguments, the first of which is the command's name; what the user would type at the *Grub* prompt to use it. The second argument is the function that implements the command and the third one is for flags that can restrict how and where the command can be used (most commands, like **hello**, aren't restricted; they just pass **0** here).

Two strings follow this: the first is a short usage summary and the second is for more verbose help text. The final argument is an array describing the command's options (we'll cover this in more detail later). These arguments accept a **0** value if they are not required – **hello** doesn't provide usage information or options. All commands automatically get **--usage** and **--help** options that display their usage and help text. The latter option displays the usage and the help

The Grub legacy

There are two main versions of *Grub*. The original version is now called 'Grub Legacy' and is what you have if your version is less than 1. The last release of Grub Legacy is version 0.97; if you have a version of at least 1 then you have *Grub 2*, which is what's now called 'Grub'. The latest version, 2.00, was released in June 2012, but some distros carry a more recent beta release.

You can check which version you have. It's displayed at the top of the *Grub* boot menu or you can view it from a command prompt:

```
$ grub --version
2.0.0-5ubuntu2
```

Grub is a complete rewrite of Grub Legacy. It has completely different and incompatible internals and configuration files. It does, however, include a legacy module that can read the **menu.lst** configuration file used by Grub Legacy.

LV PRO TIP

make clean removes compiled files from your build directory. **make distclean** also removes what **configure** generated.

Test environments

Testing *Grub* requires installing to a bootable medium and booting from it. This would quickly become very tiresome if you needed to keep rebooting in order to test but, fortunately, you don't have to.

One option is to install to a removable medium such as a USB stick and then boot it on a spare machine. Use **dmesg** to identify your device after inserting it – take care to get the right device, because mistakes can cause data loss! Our examples use **/dev/sdX**, but you'll need to replace **X** with the digit that matches your setup.

```
# mount /dev/sdX /mnt
# grub-install --no-floppy --boot-directory=/mnt /dev/sdX
# umount /mnt
```

You could use a virtual machine, say with *VirtualBox*, to boot a USB stick that you've installed *Grub* on. If you want to do this, create a raw disk image first:

```
$ VBoxManage internalcommands createrawvmdk -rawdisk /dev/sdX -filename usb-stick.vmdk
```

Then create a new *VirtualBox* machine and use **usb-stick.vmdk** instead of creating a hard disk. Start the VM and it will boot from the USB. The other option is to use the *Grub* emulator, **grub-emu**.

You need to build the emulator separately – it isn't built along with the real *Grub*. To do this, configure your working copy to build the emulator. Here's one way to do this:

```
$ mkdir ~/work/build-emu
$ cd ~/work/build-emu
$ ../grub/configure --with-platform=emu
$ make
$ make DESTDIR=~/work/emu install
```

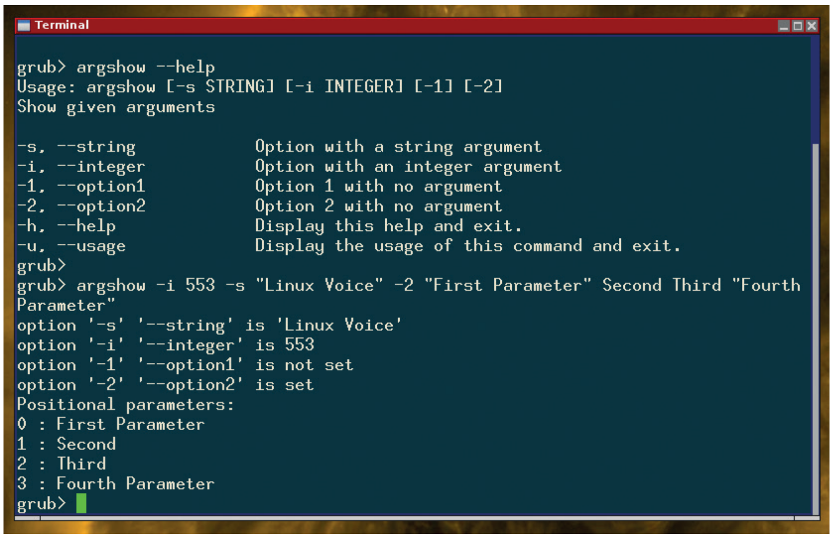
Then, to run it:

```
$~/work/emu/usr/local/bin/grub-emu
```

Enter **exit** at the emulator's **grub>** prompt to exit back to your shell. Run it with **--help** for more information.

The emulator will use **libSDL** (Simple DirectMedia Layer) to run in a window if possible, but it will otherwise run in your terminal. Adding **--disable-grub-emu-sdl** to the **configure** command will disable SDL if you don't want to use it.

Modules are built into the emulator, so you don't need to use **insmod** to load them. You can still do it but **insmod** does nothing in the emulator. The emulator has a few limitations that may affect your testing depending on what your module does. If you encounter problems with it, a virtual machine is a good fallback option.



```
grub> argshow --help
Usage: argshow [-s STRING] [-i INTEGER] [-1] [-2]
Show given arguments

-s, --string            Option with a string argument
-i, --integer           Option with an integer argument
-1, --option1           Option 1 with no argument
-2, --option2           Option 2 with no argument
-h, --help              Display this help and exit.
-u, --usage             Display the usage of this command and exit.
grub>
grub> argshow -i 553 -s "Linux Voice" -2 "First Parameter" Second Third "Fourth
Parameter"
option '-s' '--string' is 'Linux Voice'
option '-i' '--integer' is 553
option '-1' '--option1' is not set
option '-2' '--option2' is set
Positional parameters:
0 : First Parameter
1 : Second
2 : Third
3 : Fourth Parameter
grub>
```

Grub's new **argshow** command, shown here in all its glory in the emulator. Booting will never be the same again!

text followed by a summary of any options that the command supports.

The **_()** and **N_()** macros that are used in the **hello** module are for native language support.

Grub commands accept command-line arguments – a mixture of options and positional parameters. Options begin with a hyphen and can be given in a short form such as **-h** or long form such as **--help**. They can appear in any order and may expect a value. Positional parameters don't begin with a hyphen, but the command will usually expect them to be given in a specific order.

A command is implemented in the function passed to **grub_register_extcmd** like the **hello** command's function **grub_cmd_hello**. Command functions take three arguments:

- **struct grub_extcmd_context_t ctxt** is a pointer to the command's context,
- **int argc** and
- **char **args** point to its positional parameters.

The command's context contains its options and other things. The body of the function outputs the famous greeting:

```
grub_printf ("%s\n", _("Hello World"));
```

It highlights a major difference to most Linux C code: *Grub* doesn't use the standard C library (**libc**). Instead, it includes equivalent implementations of the main functions. These have the same names but are prefixed with **grub_**. Most of these functions are accessible after including **misc.h** instead of the usual C headers such as **stdio.h**.

LV PRO TIP

It's a good idea to follow the GNU coding standards and is essential if you plan to submit your work upstream **www.gnu.org/prep/standards**.

Your own module

An easy way to start developing a new module is simply to copy the **hello** module and modify it. As an example, we'll write a module that just displays its arguments. We'll call the module and its command **argshow**.

You can choose where to write your module – either in the *Grub* source tree in a similar way to **hello** or in a separate, self-contained one. The latter is useful if you want to keep your work separate, perhaps in its own *Git* repository. We'll use **~/work/modules**. This working directory must not contain the *Grub* source tree.

If you want to work in the *Grub* tree, use the **grub-core** subdirectory as your working directory.

Your module's files will go in a subdirectory of your chosen working directory. Go to your working directory and copy the **hello** sources:

```
$ mkdir argshow
$ cp ~/work/grub/grub_core/hello/hello.c argshow/argshow.c
$ sed -i -e 's/hello/argshow/g' argshow/argshow.c
```

A module definition needs to be provided to tell **autogen.sh** about the new module so that it gets built. Write a module definition file, **Makefile.core.def** in the same directory as **argshow.c**:

```
AutoGen definitions Makefile.tpl;
module = {
 name = argshow;
 common = contrib/argshow/argshow.c;
};
```

If you chose to write your module under **grub-core**, just append the **module** block at the end of the existing **grub-core/Makefile.core.def** file but without the **contrib** path element.

You need to set **GRUB_CONTRIB** to the parent of

The basic structure of a Grub module.

```
#include <grub/types.h>
#include <grub/misc.h>
#include <grub/mm.h>
#include <grub/err.h>
#include <grub/dl.h>
#include <grub/extcmd.h>
#include <grub/i18n.h>

GRUB_MOD_LICENSE ("GPLv3+");

static grub_err_t
grub_cmd_hello (grub_extcmd_context_t ctxt __attribute__
((unused)),
                        int argc __attribute__ ((unused)),
                        char **args __attribute__ ((unused)))
{
 grub_printf ("%s\n", _("Hello World"));
 return 0;
}

static grub_extcmd_t cmd;

GRUB_MOD_INIT(hello)
{
 cmd = grub_register_extcmd ("hello", grub_cmd_hello, 0, 0,
                                    N_("Say `Hello
World'."), 0);
}

GRUB_MOD_FINI(hello)
{
 grub_unregister_extcmd (cmd);

}
```

your module's directory if that isn't **grub-core**. From your working directory:

```
$ export GRUB_CONTRIB=~/work/modules
```

This is used by **autogen.sh** to symlink **grub-core/contrib** to your working directory. Any modules beneath it will be included in the build.

That completes the plumbing for our module and it's sufficient for a build. Return to the *Grub* source directory and run **autogen.sh** again. Then return to the directory in which you want to build and run **configure** and **make** like we did before.

If all went well, install *Grub* in a test environment, try installing the module and then execute the new command. From a *Grub* prompt:

```
grub> insmod argshow
grub> argshow
hello, World!
```

You should see the same familiar greeting because we haven't written any code yet. Let's do that now.

Make your argument

Before you start coding, decide how you want to license your work and declare this in the source file (the build will fail otherwise). We'll follow *Grub*'s lead and use the GPLv3; we declare this in our source file like this:

```
GRUB_MOD_LICENSE ("GPLv3+");
```

The headers included by default cover the module's basic needs and are sufficient for our example. You may need to include others when writing a real module; browse **include/grub** to see what's available.

Most commands will need some data to work with and this is usually supplied on the command line. Our module demonstrates how command-line arguments (options and other positional parameters) work in *Grub*. Positional parameters are easy – they're passed into the command function as **argc** and **args**.

Options, however, need to be defined in an array of **grub_arg_option** structures, one per option, and passed to **grub_register_extcmd** when registering the command. Each definition comprises:

- A long (string) name;
- A short (character) name;
- A string for "--help" to display;
- A default value and;
- A type.

We declare some options below; the end of the list is marked by an empty record:

```
static const struct grub_arg_option options[] = {
 {"string", 's', 0, _("Option with a string argument"), 0, ARG_
TYPE_STRING},
 {"integer", 'i', 0, _("Option with an integer argument"), 0, ARG_
TYPE_INT},
 {"option1", '1', 0, _("Option 1 with no argument"), 0, ARG_TYPE_
NONE},
 {"option2", '2', 0, _("Option 2 with no argument"), 0, ARG_TYPE_
NONE},
 {0, 0, 0, 0, 0, 0}
};
```

Update the command registration to describe the command and its options:

```
GRUB_MOD_INIT(argshow)
{
 cmd = grub_register_extcmd ("argshow", grub_cmd_argshow, 0,
                  N_("[-s STRING] [-i INTEGER] [-1] [-2]"),
                  N_("Show given arguments"),
                  options);
}
```

One disappointing thing to note here is that **grub_register_extcmd** expects the help text to be a constant; this means that it can't be translated.

> **"Grub takes care of invalid options before invoking the command function."**

Now that the command accepts arguments its function can process them. *Grub* takes care of invalid options before invoking the command function so it only needs to consider valid ones. Their "state" is contained in the command's context that is passed into the function. An option's state describes whether it is set and, if applicable, its value.

We set up a "state" pointer to access options' states.

```
struct grub_arg_list *state = ctxt->state;
```

Our example function just iterates over its options and displays their state:

```
int i = 0;
for (i=0; options[i].shortarg; i++)
 {
  grub_printf(_("option '-%c' '--%s' is "),
              options[i].shortarg, options[i].longarg);
  if (state[i].set)
   if (options[i].type == ARG_TYPE_STRING)
    grub_printf("'%s'\n", state[i].arg);
   else if (options[i].type == ARG_TYPE_INT)
    grub_printf("%lu\n", grub_strtoul (state[i].arg, 0, 0));
   else
    grub_printf(_("set\n"));
  else
   grub_printf(_("not set\n"));
 }
```

The options are described in the global **options** array; this is where you can find their name, value, description and type. You get their states via the **state**

Native language support

Grub can display text in languages other than English. It uses GNU GetText for this native language support, which can provide text translations into multiple languages.

To help with this, wrap translatable text **_("like this")** and text that should not be translated **N_("like this")**. If writing a new **.c** file, include the relevant internationalisation header:

```
#include <grub/i18n.h>
```

Here are some examples:

```
grub_printf (_("Hello %s, this will be translated."), name)
grub_printf (N_("Hello %s, this won't be translated."), name)
```

Grub participates in the Translation Project (**translationproject.org**) so that others can translate text that coders mark up in this way. You don't need to provide translations yourself – if you use the text wrappers in your code, other translators will be able to do it for you.

argshow.c – the code

```
/* argshow.c - Linux Voice Tutorial */
#include <grub/types.h>
#include <grub/misc.h>
#include <grub/mm.h>
#include <grub/err.h>
#include <grub/dl.h>
#include <grub/extcmd.h>
#include <grub/i18n.h>

GRUB_MOD_LICENSE ("GPLv3+");

static const struct grub_arg_option options[] = {
  {"string",  's', 0, N_("Option with a string argument"), 0, ARG_TYPE_STRING},
  {"integer", 'i', 0, N_("Option with an integer argument"), 0, ARG_TYPE_INT},
  {"option1", '1', 0, N_("Option 1 with no argument"), 0, ARG_TYPE_NONE},
  {"option2", '2', 0, N_("Option 2 with no argument"), 0, ARG_TYPE_NONE},
  {0, 0, 0, 0, 0, 0}
};


static grub_err_t
grub_cmd_argshow (grub_extcmd_context_t ctxt, int argc, char **args)
{
  struct grub_arg_list *state = ctxt->state;
  int i = 0;

  /* options */
  for (i=0; options[i].shortarg; i++)
   {
    grub_printf(_("option '-%c' '--%s' is "), options[i].shortarg, options[i].longarg);
     if (state[i].set)
      if (options[i].type == ARG_TYPE_STRING)
       grub_printf("'%s'\n", state[i].arg);
      else if (options[i].type == ARG_TYPE_INT)
       grub_printf("%lu\n", grub_strtoul (state[i].arg, 0, 0));
      else
       grub_printf(_("set\n"));
     else
      grub_printf(_("not set\n"));
   }

  /* positional parameters */
  if (argc)
    grub_printf ("%s\n", _("Positional parameters:"));
    for (i=0; i<argc; i++)
     grub_printf ("%d : %s\n", i, args[i]);

  return 0;
}

static grub_extcmd_t cmd;

GRUB_MOD_INIT(argshow)
{
  cmd = grub_register_extcmd ("argshow", grub_cmd_argshow, 0,
                   N_("[-s STRING] [-i INTEGER] [-1] [-2]"),
                   N_("Show given arguments"),
                   options);
}

GRUB_MOD_FINI(argshow)
{
 grub_unregister_extcmd (cmd);
}
```

Listing of **argshow.c**. You can download the sources from **http://git.io/vqc4s**.

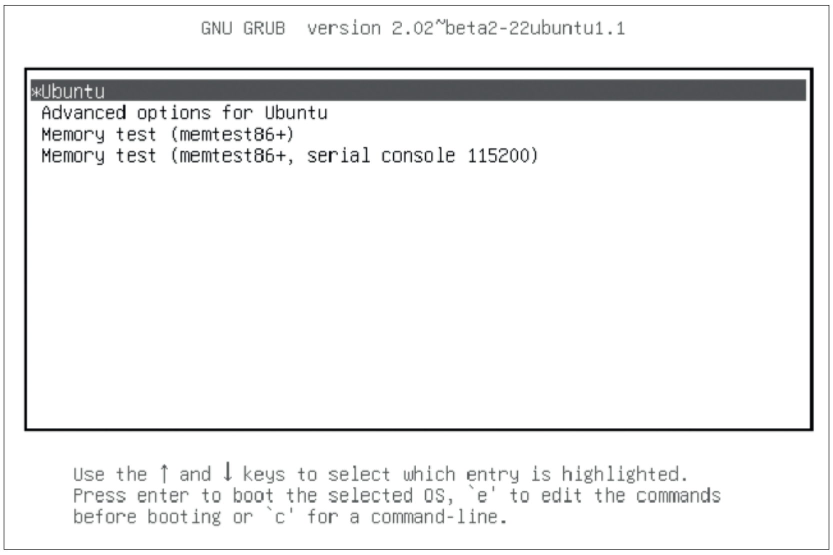
pointer. Notice how the **integer** option is given as a string and needs to be converted.

"With the basics in place, you're free to enhance Grub with that command you've always wanted."

The command function receives any positional parameters as **args** and the number of them in **argc**. They're just string values, and we can use a simple loop to display them:

```
if (argc)
 grub_printf ("%s\n", _("Positional parameters:"));
 for (i=0; i<argc; i++)
  grub_printf ("%d : %s\n", i, args[i]);
```

Return to the directory where you build from and remake *Grub*. The build configuration hasn't changed since the last time, so running **make** once more should get the job done. Install again to your test environment and try out the module's argument handling.



```
        GNU GRUB  version 2.02~beta2-22ubuntu1.1

 *Ubuntu
  Advanced options for Ubuntu
  Memory test (memtest86+)
  Memory test (memtest86+, serial console 115200)

     Use the ↑ and ↓ keys to select which entry is highlighted.
     Press enter to boot the selected OS, `e' to edit the commands
     before booting or `c' for a command-line.
```

Grub is magic. Without it, we would be able to try nearly as many Linux distributions as we do.

Environmental impact

The *Grub* environment is another place where your module can access information. It's basically a list of name/value pairs that you can read and write. You use **grub_env_get** and **grub_env_get** to do this. They read and write string values; you need to include **grub/env.h** in your source file to use them.

With the basics in place, you're free to go ahead and enhance *Grub* with that command you've always wanted. Your next step will probably be to learn more about what's actually possible. Most of the interesting source code is under **grub-core** and organised in subdirectories such as **disk**, **net** and **video** to name just a few.

There's an active mailing list at **https://lists.gnu.org/mailman/listinfo/grub-devel**, which is where you need to send anything you want to contribute upstream. If you're considering contributing upstream, **http://bit.ly/code4opensource** is a good read.

An alternative, and perhaps more appropriate, potential home for your work is in the "Grub Extras" repository. You'll find it at **http://savannah.nongnu.org/projects/grub-extras**. It takes advantage of *Grub*'s framework for building modules externally in order to provide them from a separate package – just like we did in this tutorial.

John Lane hacks on Linux for fun and profit. Even his bootloader couldn't escape.

Ever wondered who supports open source developers?

We are Bytemark and we support the people who support the free software community.

Come and play around with your projects on a server for £10/month*.

For more details visit: **bytemark.co.uk/bigv**

BYTEMARK

*Excl VAT

LINUX VOICE TUTORIAL

EDIT VIDEO WITH KDENLIVE FOR YOUTUBE STARDOM

BEN EVERARD

Free software can help you make it big on the silver screen, or reach millions of the great unwashed via YouTube.

WHY DO THIS?
- Learn how to cut and mix video like a pro
- Create helpful videos for your fellow Linux users
- Amass a YouTube following and become rich

Video screen captures are a great way of sharing tips and tricks with the world at large. With YouTube and other video sharing sites, you don't need to worry about bandwidth or the complexities of hosting streamed videos; only about making the content interesting for your viewers. In this tutorial, we'll look at how to make a tutorial video using the popular *Kdenlive* video editor.

The first step of making a video isn't the editing though; it's not even the recording of it. The first step is to make sure you've got a proper plan of what you want to shoot. It's best to break the plan down into the constituent scenes, each of which will be you performing one task and explaining it. By planning in this way, you'll make it easier to record, and avoid the problem of realising halfway through recording that you forgot to perform a task earlier on.

"After you've got everything recorded and exported, it's time to mix it all together in Kdenlive."

Once you've got your plan, the next step is to record the video and audio. There are two approaches here. One is to record them together. This method is quicker and easier since you don't have to worry about cutting, and mixing becomes easier. However, it's hard to do this well since it's difficult to cut out any umms and arrrs you say, and redoing a single bit usually means rerecording everything. The other approach is to record the two pieces separately. This takes longer to do, and the editing stage is a bit more involved, but it can result in a much higher quality video. The choice, as they say, is yours.

Whichever option you use, you'll need a method for recording the screen and a microphone for recording the audio. *RecordMyDesktop* does a great job for the former. There are *GTK* and *Qt* front-ends available in most package managers that make the task straightforward.

You could use a laptop's built-in microphone for audio, but these won't allow you to record high-quality audio. It's much better to use an external mic. You don't need a top-of-the-line microphone for this, and an external USB mic should be fine.

Recording the easy way

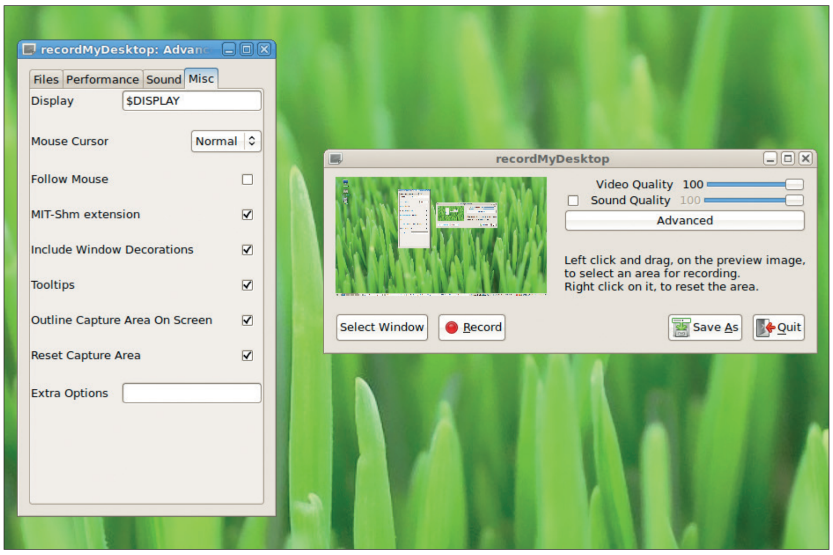
If you're recording video and audio in one go, all you need to do is configure *RecordMyDesktop* to capture both the audio and video, and you're done (we told you it was easy). However, if you want to make a higher-quality recording with the video and audio done separately, you'll need to first record the video. This is best done in one go (we'll chop it up into the scenes a bit later). Once the video's done, you can record the audio, and *Audacity* is the best software for this.

The basic usage of *Audacity* is just to make sure that you're recording from the right source, press the big red button to start recording, then press it again when you're finished. If you get everything in a single recording, you can copy and paste sections of the track into new *Audacity* projects to get the bits you want in the right clips.

Microphones can pick up a lot more than just the sounds you want, so it's a good idea to run the noise reduction effect. This removes a lot of hissing and background noise, leaving a much clearer sound. Once you have a project with just the sections you want, you can export the audio by going to File > Export Audio. The default settings will produce clips that you can import into *Kdenlive*.

After you've got everything recorded and exported, it's time to mix it all together to create your final video. The first thing to do is import all the bits into *Kdenlive*. You can do this by going to Project > Add Clip and selecting the files that you've recorded. Both audio and video segments are termed clips, and they will appear in the clips list on the top-left of the window. From here, you can drag and drop them into position on the various tracks. If you've recorded your video in one take, you should now have one video clip and a

GTK-RecordMyDesktop has lots of options if you want to tweak the recording, but we find that the defaults are right for most purposes.


recordMyDesktop: Advanc

Files | Performance | Sound | Misc

Display	$DISPLAY
Mouse Cursor	Normal
Follow Mouse	☐
MIT-Shm extension	✓
Include Window Decorations	✓
Tooltips	✓
Outline Capture Area On Screen	✓
Reset Capture Area	✓
Extra Options	

recordMyDesktop

Video Quality 100
☐ Sound Quality 100

Left click and drag, on the preview image, to select an area for recording.
Right click on it, to reset the area.

bunch of audio clips. To start with, drag the video clip onto the first video track. You can now play the video, and it should appear in the monitor on the right-hand side of the window.

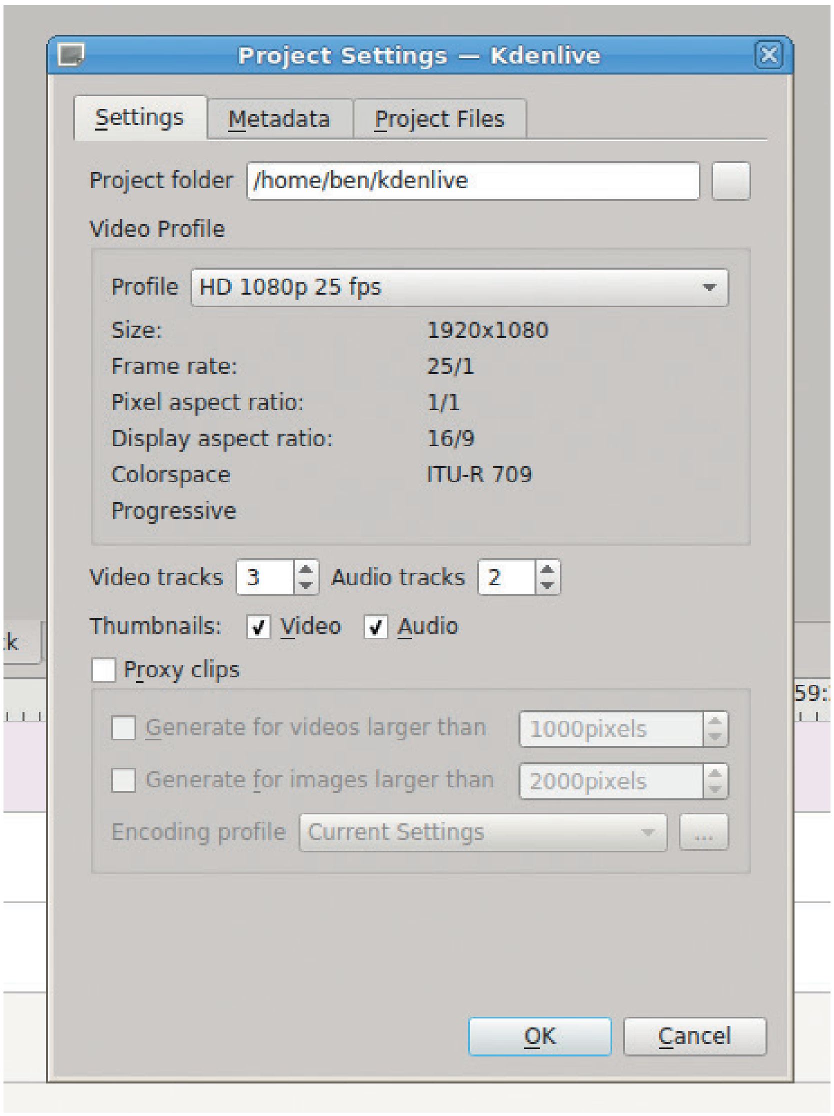
Matching audio and video

In order to match up the video with the audio, you need to separate the video into the scenes that you recorded the audio against. First find the point where one scene ends and the other begins. You can either do this by playing through the clip (press Space to play and pause), or by moving the cursor by clicking in the timeline. Once you've found the end of the scene, you can cut the video by right-clicking and selecting Cut Clip from the popup menu (or press Shift+R). This will cut the clip into two separate ones. You can drag the second clip (which contains the remainder of the video) down to the second video track. This gives us space to alter the scene without it overlapping.

With the clip cut to length, you can now drag the relevant audio clip into the audio track inline with the video (you named them well enough to be able to identify the right one didn't you?). You'll probably find at this point that the two aren't the same length. Either the video is longer than the audio or *vice versa*. Exactly what you do here is a question of artistic preference. If the video is intricate, it may be best to leave a few seconds of silence to let the viewers focus on what's happening.

One of the great things about screen captures is they can be sped up or slowed down without making them look strange. You can use this to your advantage to make the video the same length as the audio. This is particularly useful if there's something that's slow to start or load – there's no need to waste the viewer's time with unnecessary pauses.

To adjust the speed, right-click on the clip in the timeline and select Add Effect, Motion, Speed. In the Effects list pane, make sure that the Effect Stack tab is selected, and you will be able to adjust the speed by changing the percentage (a smaller percentage runs slower and a faster percentage runs faster). You can either calculate the correct value, or just find it using a bit of trial and error. It doesn't have to be exactly the same length, and it can be good for a video to have some slight pauses so the viewer doesn't feel overloaded with information.


Project Settings — Kdenlive

Project folder: /home/ben/kdenlive

Video Profile

Profile: HD 1080p 25 fps

Size:	1920x1080
Frame rate:	25/1
Pixel aspect ratio:	1/1
Display aspect ratio:	16/9
Colorspace	ITU-R 709
Progressive	

Video tracks 3 Audio tracks 2

Thumbnails: [✓] Video [✓] Audio

[ ] Proxy clips

[ ] Generate for videos larger than 1000pixels

[ ] Generate for images larger than 2000pixels

Encoding profile: Current Settings

When you start a project you have to select the profile. This can't easily be changed afterwards.

Add clips to the video sequence

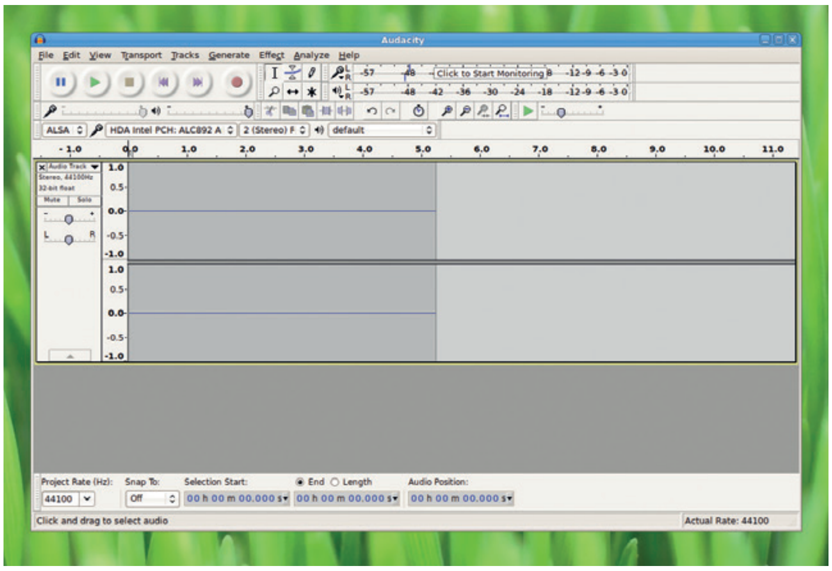
Once you've adjusted the speed of the clip to sync, you can then go through the same process for the remaining clips. Each time, after you cut out the

Non-Linear Video Editor Other open source video editors

The name *Kdenlive* may sound confusing, but it's an acronym of what it is: a KDE Non-Linear Video Editor. The split between linear video editing and non-linear editing came with the advent of digital video. This ended the reliance on reels of film that had to be viewed (and therefore edited) in sequence, also known as linearly. With digital video, you can instantly jump to any point in the video, so can change a frame without first having to go through all the previous frames, therefore you can edit it non-linearly. Almost all modern video editors are non-linear.

We've used *Kdenlive* for this tutorial, but it's just one of a growing number of open source non-linear video editors that run on Linux. Here are a few of the other options:

- **Pitivi** This project is raising money to help develop version 1.0 of the project. At the time of writing, it had raised €23,000 out of a goal of €35,000. At present the pre-1.0 release has a nice interface, but we found it a little too buggy for serious work. It's progressing quickly, and if the cash injection can help apply polish, this could become a great option in the future.
- **OpenShot** For a while, *OpenShot* was the go-to video editor on Linux, but not much seems to be happening with the project any more. The last version came out in 2012. It still works, though can be a little crash-prone. In recent years, this software has been surpassed by *Kdenlive* and *Pitivi*.
- **Blender** The 3D modelling tool also has a built-in video editor that's really quite capable. We could easily have done this tutorial using *Blender* instead of *Kdenlive*, but ultimately, we prefer *Kdenlive*. However, it's well worth investigating if you're doing serious video editing.


Audacity
File Edit View Transport Tracks Generate Effect Analyze Help
Click to Start Monitoring
ALSA
HDA Intel PCH: ALC892 A
2 (Stereo) F
default
Audio Track
Stereo, 44100Hz
32-bit float
Mute Solo
Project Rate (Hz): 44100
Snap To: Off
Selection Start: 00 h 00 m 00.000 s
End Length 00 h 00 m 00.000 s
Audio Position: 00 h 00 m 00.000 s
Click and drag to select audio
Actual Rate: 44100

Audacity has a lot of options, but don't let it intimidate you: it's easy to use for basic audio processing.

correct section of video, you can drag and drop it into video track one next to the previous clip, and likewise with the audio.

There's a slight bug in some versions of *Kdenlive* where it sometimes won't let you drop a video clip next to another one after the speed of a clip has been changed. It just throws an error and puts the video clip in its previous place. If you get this, just save the project, close *Kdenlive*, then reopen the program and the project and it should work.

Rough cut

By this point you should have a rough version of your video. There should be a full set of video and audio clips, but let's take a look at how to make it a bit more polished. One problem with cutting audio together is that it can leave audible clicks at the point where tracks start or finish. This can be solved by fading tracks out and in. If you left a small period of silence at the start and end of your audio, this is quite simple to do using effects. Right-click on the audio track and do Select > Add Effect > Fade, and you'll need to add both fade ins and fade outs for every track. As with the speed effect we added to the videos, the fades create an entry in the effects stack pane. Using these you can adjust the amount of time the fade takes. The amount will depend on your recordings, so just experiment and use the smallest amount that gets rid of any audio artifacts.

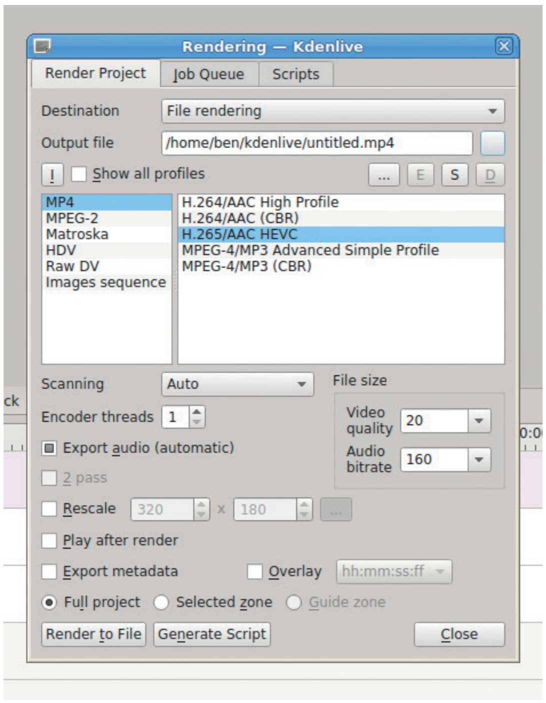
"There are are several types of transition, which swap from one track to another."

Another thing our video is missing is an opening title that will introduce the video to the viewers. *Kdenlive* can import most image files as clips, so the first thing is to create your intro scene in your favourite image editor and export it as an image file (a PNG works well). Once you've got this, you need to add it to the front of the video, but there's no space there because the first clip starts at 0. Of course, we could have planned ahead and left space at the start, but that wouldn't have left us with a way to introduce *Kdenlive*'s grouping feature. This works in a very similar way to grouping in other media software. You highlight all the clips (in the tracks rather than in the clips list) that you want to group, then right-click on one of the clips and select Group Clips. Once this is done, any action you take on one of the clips will be performed on all of them. To move all of the clips further to the right and create space for our intro, you just need to select all of the clips with Ctrl+A, then group them and drag them about 6 or 7 seconds to the right. This will move everything and keep the video and audio in sync.

Add a title screen

You should now have space to insert your title screen. You can expand still image clips by dragging the edge of the clip to the length you want it to be. If you put the title clip immediately before the start of the first video there will be a jarring cut from the title to the screencast, but there are a few things you can do to make this cut a bit more pleasant on the eye. The simplest option is to fade out the title screen, then fade in the video. This is done in a very similar way to fading in and out audio tracks except instead of using the effects Fade Out and Fade In, you need to use Fade To Black and Fade From Black.

Another option is a transition. There are several different types of transition that swap from one track to another with different effects. The first thing you need to do when applying a transition is put the two video clips on separate tracks. In our case, it's easiest to put the title clip on the second video track and leave the main videos on the first. You then need to make


Rendering — Kdenlive

Render Project | Job Queue | Scripts

Destination: File rendering
Output file: /home/ben/kdenlive/untitled.mp4
Show all profiles

MP4
MPEG-2
Matroska
HDV
Raw DV
Images sequence

H.264/AAC High Profile
H.264/AAC (CBR)
H.265/AAC HEVC
MPEG-4/MP3 Advanced Simple Profile
MPEG-4/MP3 (CBR)

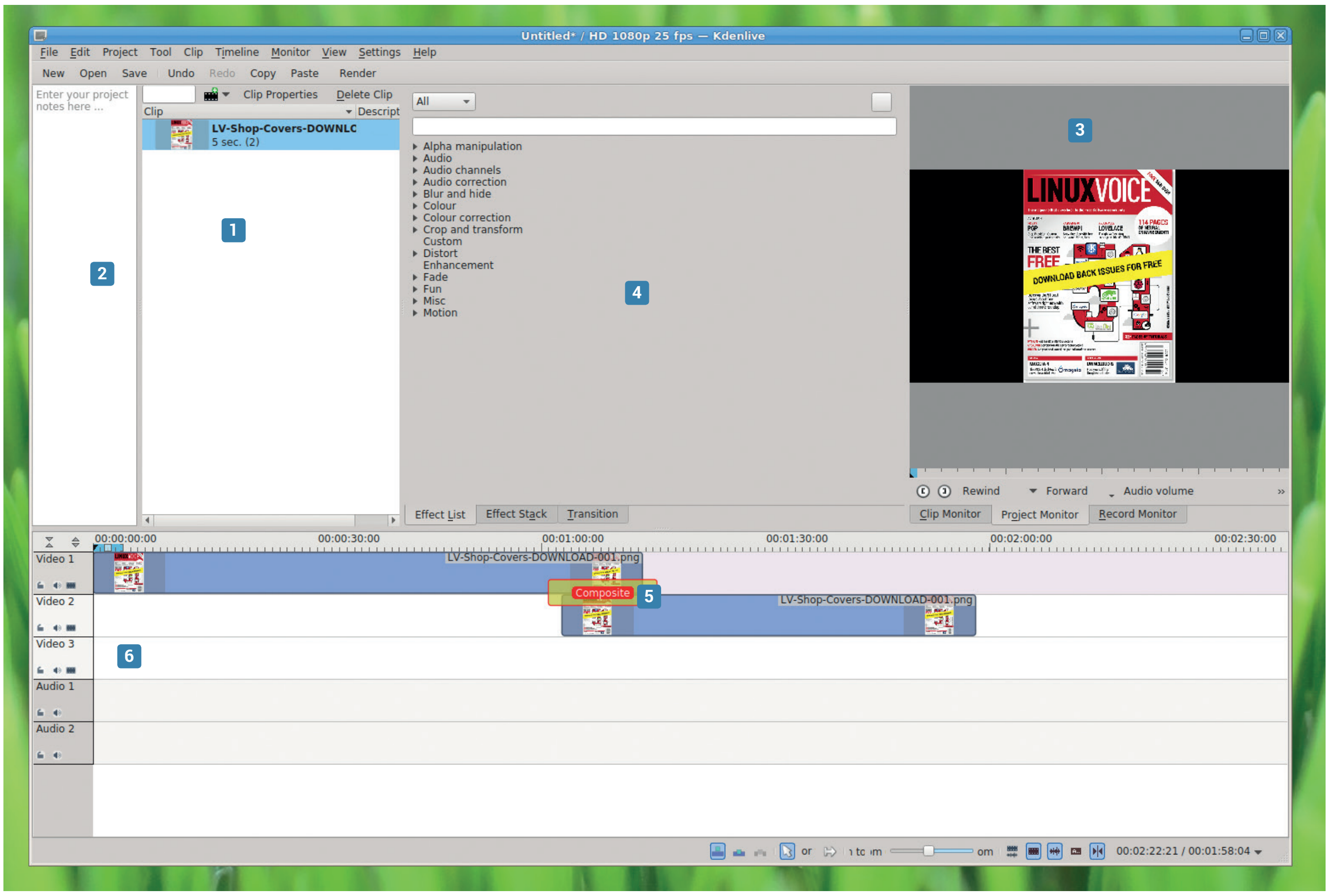
Scanning: Auto
Encoder threads: 1
Export audio (automatic)
2 pass
Rescale 320 x 180
Play after render
Export metadata
Overlay hh:mm:ss:ff
Full project | Selected zone | Guide zone

File size
Video quality: 20
Audio bitrate: 160

Render to File | Generate Script | Close

Video formats are a minefield of quality and compatibility issues, but you can always re-export in another format if you have problems.

The Kdenlive interface


Untitled* / HD 1080p 25 fps — Kdenlive
File Edit Project Tool Clip Timeline Monitor View Settings Help
New Open Save Undo Redo Copy Paste Render
Enter your project notes here ...
Clip Properties Delete Clip
LV-Shop-Covers-DOWNLC 5 sec. (2)
Alpha manipulation
Audio
Audio channels
Audio correction
Blur and hide
Colour
Colour correction
Crop and transform
Custom
Distort
Enhancement
Fade
Fun
Misc
Motion
Effect List Effect Stack Transition
Rewind Forward Audio volume
Clip Monitor Project Monitor Record Monitor
Video 1 Video 2 Video 3 Audio 1 Audio 2
Composite
LV-Shop-Covers-DOWNLOAD-001.png
00:02:22:21 / 00:01:58:04
1 2 3 4 5 6

1 The clips list From here you can drag clips on to the required tracks. Each clip can be on more than one track if desired.
2 Notes This is free-form text that you can use to add any notes that you want to remember when doing the editing. This doesn't change the video.
3 Monitor You can see the current state of the video in this box.
4 Effects pane Here you can adjust the effects and transitions.
5 Transition You can place a transition between two video tracks to add an effect as the video mixes from one track to the other.
6 Tracks Place video or audio clips onto the tracks to add them to the film.

sure that there is an overlap between the two video clips. It's this overlap that is where the transition will take place. Since we can expand the title clip without problems, the easiest way is to extend this another second or two over the other track.

Transitions

You can then create a new transition by right-clicking on the title clip, selecting Add Transition and selecting one. Slide, for example, will slide the old video out revealing the new video clip as it goes, while Dissolve will gradually fade out the old video showing the new clip behind it. *Kdenlive* can sometimes place the transition in funny places when you first insert it, but you can drag and drop it to the correct position.

Once you have everything in place, the only thing left to do is export the project to a video format that's best for your intended use. Go to Project > Render to begin the process. The first thing you need to do is select the correct video format. There aren't any hard-or-fast rules about what's the best format, but a good basic guideline is that MPEG-2 is compatible with more devices, while MPEG-4 produces smaller files at higher qualities. If you're uploading video to an external site such as YouTube, check their guidelines on what they accept.

Rendering the video can take a while depending on the speed of your computer and the complexity of the video. Rather than locking up the interface for this time, *Kdenlive* will start a background process for rendering and notify you when it's complete.

It's always a good idea to keep the *Kdenlive* project files even after you've exported the video because these keep all the data, so if you need to make any further edits, it's better to use this than trying to import a previously exported clip. You can also re-export the project if you need it in a different format.

That's all there is to it! By choosing your subject well, planning properly and spending a little time editing, you can make interesting and useful videos to help other Linux users and gain online fame. LV

Ben Everard is the author of books, builder of robots, maker of cider and rider of bikes.

LINUXVOICE TUTORIAL

PROLOG: NATURAL LANGUAGE AND ARTIFICIAL INTELLIGENC

JULIET KEMP

Instead of operating on objects or performing functions, Prolog works backwards to find a result – weird, eh?

In lots of ways, Prolog is quite different from the other languages we've looked at. Its roots are in natural language processing and computational linguistics, and it's strongly associated with artificial intelligence. If you're used to other types of language, its structure can be a bit challenging to get your head around initially. But if you put the effort in, it's a fascinating language – and still used in anger in some artificial intelligence work, in database searching, and in natural-language applications.

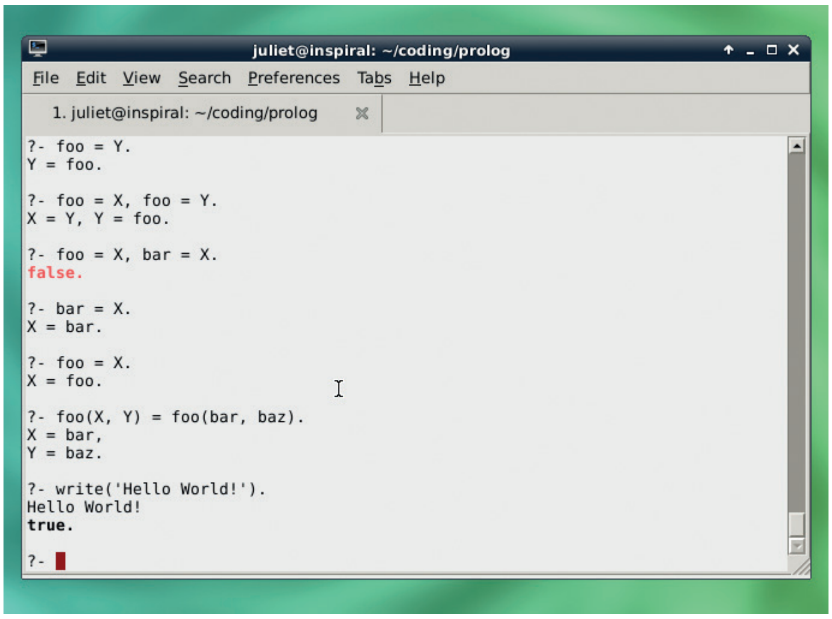
Prolog originated from research at the University of Aix-Marseille around 1971, where Alain Colmerauer and Phillipe Roussel were working, not on programming languages, but on computer processing of natural languages. Colmerauer was interested in communicating directly with computers in French (and in natural, or human, languages in general). This meant, among other things, finding ways for a computer to draw the implicit conclusions and inferences that humans do from sentences in natural languages. For a human, saying "Tom is a cat; Jerry is a mouse", together with the base knowledge "Cats chase mice", would enable them to answer the question "Will Tom chase Jerry?". Getting a computer to do this was a significant challenge.

Roussel, together with another colleague, Jean Trudel, were working on automated theorem-proving (a mathematical/computational logic challenge). This fitted well with Colmerauer's research. Working with Trudel and Roussel's theorem-provers, the team developed a very basic natural language communication system. A user could give the computer a set of facts in natural language, and the computer would be able to draw certain logical conclusions from those facts and respond to questions; just like the Tom and Jerry example. It was after this that they met up with Robert Kowalski, at the University of Edinburgh, who was a specialist in automated theorem-proving, working particularly on SL-resolution (a type of inference rule which enables logic sentences to be proved or refuted).

Between them, they worked throughout 1971 and 1972 on what would become Prolog, and the first version was implemented, in Algol-W, in autumn 1972. At the same time, a natural language computer program was being written in this new Prolog language. A second version of Prolog followed in 1973 that bears a stronger resemblance to modern Prolog.

Over the next years, copies of Prolog slowly made their way out into the world, as researchers from other institutions would come, spend some time in Marseille, and take a copy away with them. This was still an interpreter, though, and thus very slow-running. The first Prolog compiler, the Warren Abstract Machine, was written in the mid 1970s by David Warren at the University of Edinburgh. Compiling Prolog into a lower-level language meant that it could run much faster, and the WAM's Prolog dialect became the basis for most modern implementations.

Turbo Prolog, released by Borland in the 1980s, helped popularise Prolog further. For more on the rise of Prolog compilers, and some detailed discussion of how they work, see this paper by Peter van Roy – **http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.8982**. Prolog is still in use today: Watson, the natural language AI computer that won the TV quiz show *Jeopardy* in 2011, uses Prolog (among other languages). Read on to try it out.


juliet@inspiral: ~/coding/prolog
File Edit View Search Preferences Tabs Help
1. juliet@inspiral: ~/coding/prolog

```
?- foo = Y.
Y = foo.

?- foo = X, foo = Y.
X = Y, Y = foo.

?- foo = X, bar = X.
false.

?- bar = X.
X = bar.

?- foo = X.
X = foo.

?- foo(X, Y) = foo(bar, baz).
X = bar,
Y = baz.

?- write('Hello World!').
Hello World!
true.

?-
```

Getting the interpreter to say hello, and experimenting with unification (see boxout).

Hello world

The most popular Prolog interpreter for Linux is the free SWI Prolog. This is available as a package for most major distros (check their website for more information). For Debian/Ubuntu, just install the package **swi-prolog**. A GNU Prolog compiler is also available but seems to be less fully-featured and doesn't have the libraries and other tools that SWI Prolog does. I'll use SWI-Prolog in this tutorial.

Once you've installed your Prolog interpreter, fire up the interactive interpreter with **swipl**. You'll get a bit of boilerplate, then an interactive prompt. Try it out:

```
?- write('Hello World'), nl.
Hello World
true.
```

write('Hello World') is a Prolog fact. (See the boxout for more on Prolog atoms, facts, and predicates.) In this case, we use the built-in predicate **write/1**, which has the side-effect of outputting something to the screen, and which is useful entirely for this side-effect. The number 1 in its description means that it takes a single argument (in Prolog terms, it has arity 1). It evaluates **true** simply because it exists. **nl** is also a predicate (**nl/0**, strictly speaking), which outputs a newline. As here, you can run several clauses one after the other, separated by commas; to actually evaluate them all, you need a full stop at the end.

To exit the interpreter, type **halt.**. Now, let's try running a code file. Open a file **hello.pl** in your editor:

```
main :- write('Hello World!'), nl.
Now start up swipl again:
?- [hello].
% hello compiled 0.00 sec, 3 clauses
true.
?- main.
Hello World!
true.
```

This loads in the file, so the Prolog interpreter now knows about this rule. A rule is a way of connecting two items together, and **:-** can be read as "if". So:

```
A :- B.
```

means that A is true if B is true. In this case, **main** is true if **write('Hello World!'), nl, halt** is true. When we then type **main.**, we are asking "is main true?". To evaluate this, the interpreter evaluates the B side of the rule, with the side-effect that the string is printed. It then tells you that yes, main is true. You can also set up a file to call the interpreter internally:

```
#!/usr/bin/swipl
:- initialization(main).
main :- write('Hello World!'), nl, halt.
```

Make this executable with **chmod u+x hello.pl**, then run it with **./hello.pl** and you'll get the expected output. (Note that this time we've added **halt** to the list of predicates, so the interpreter halts at the end.) The initialisation at the start does various bits of necessary Prolog setup to call the main goal.

Finally, you can return to this single line:

```
main :- write('Hello World!\n'), halt.
```

and compile it like this:

```
swipl --goal=main --stand_alone=true -o hello -c hello.pl
```

Run it with **./hello**. The **--goal** flag identifies where the code should start; **-o** gives the name for the output file, and **-c** identifies the source code file or files.

Travelling around

To get a bit further into Prolog, we're going to look at a version of the Travelling Salesman problem. If you're not familiar with this, the basic idea is that you take a database of roads between towns, and find the shortest route that covers all of the towns. In our variant, we'll ask Prolog to provide all the routes between any two towns, and the total distance for each route.

Prolog is useful for this kind of problem because of the way it deals with its knowledge base. Given a particular rule, Prolog will try to find as many solutions as possible, trying to instantiate variables in ways that fit the constraints. As with the box on unification, I find it useful to think of Prolog as doing its best to find a way in which the rule can be true, and only returning "false" if all its efforts fail.

We'll start off by creating a basic database of roads between four towns. Open a file, **travelling.pl**:

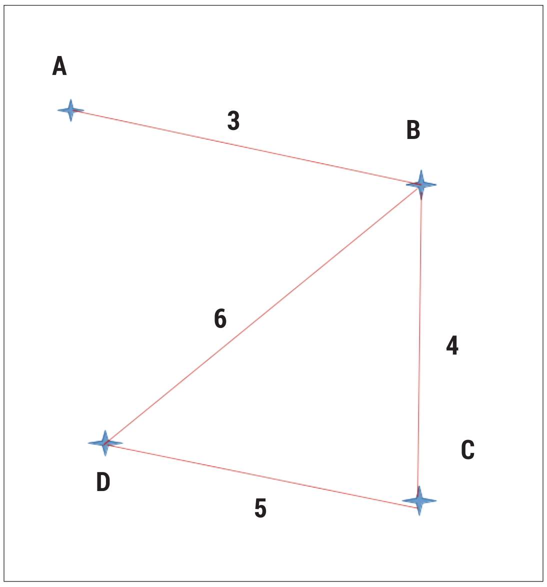
```
road(a, b, 3).
road(b, c, 4).
road(c, d, 5).
road(d, b, 6).
```

Next, let's create a predicate that finds the distance

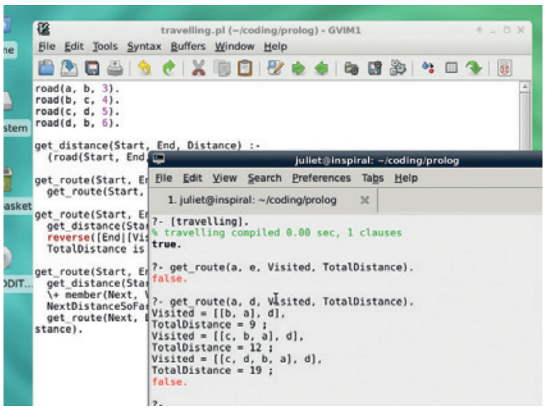
Variables, constants and atoms

Prolog has four basic building-blocks:

- **Atoms** Either a character string beginning with a lower-case letter and containing no spaces; or any character string (including spaces) enclosed in single quotes; or a special character string (like **:-**), which may have a particular meaning. Atoms can also be thought of as constants.
- **Numbers** As you'd expect: numbers! Prolog is most interested in integers, but can handle floating-point arithmetic if need be.
- **Variables** Variables are, as you would expect, variable; that is, they can be assigned a particular value (see the box on unification). Variables begin with either a capital letter or an underscore, and can't contain spaces.
- **Complex terms** A functor (a name, which must be an atom), followed by brackets (no space between functor and bracket!) and a sequence of arguments separated by commas. In other languages you might call something like this 'function' or 'method'. **foo(bar)** is a complex term, as is **write('hello')**. foo, bar, write, and 'hello' are all atoms.
- **Complex terms** are also known as predicates: a predicate is, then, a type of function that produces a true/false answer. You may also see a predicate being evaluated with an argument referred to as a fact – so **write/1** is a predicate and **write('Hello').** is a fact.
- **Rules** are if-then structures, with the head (a predicate) and body separated by **:-**. This reads as "head if body", ie the head is true if the body is true. The rule body can be built out of multiple Prolog clauses or sentences, separated by a comma, which acts as an 'and' operator.



Towns and roads in a connection graph.



travelling.pl (~/coding/prolog) - GVIM1

```
road(a, b, 3).
road(b, c, 4).
road(c, d, 5).
road(d, b, 6).

get_distance(Start, End, Distance) :-
  (road(Start, End

get_route(Start, E
  get_route(Start,

get_route(Start, E
  get_distance(Sta
  reverse([End|[Vi
  TotalDistance is

get_route(Start, E
  get_distance(Sta
  \+ member(Next,
  NextDistanceSoFa
  get_route(Next,
stance).
```

juliet@inspiral: ~/coding/prolog

```
?- [travelling].
% travelling compiled 0.00 sec, 1 clauses
true.

?- get_route(a, e, Visited, TotalDistance).
false.

?- get_route(a, d, Visited, TotalDistance).
Visited = [[b, a], d],
TotalDistance = 9 ;
Visited = [[c, b, a], d],
TotalDistance = 12 ;
Visited = [[c, d, b, a], d],
TotalDistance = 19 ;
false.

?-
```

File Edit Tools Syntax Buffers Window Help
File Edit View Search Preferences Tabs Help
1. juliet@inspiral: ~/coding/prolog

Loading the file, asking for a non-existent location which returns false, asking for an existent location and getting multiple solutions. (Note that I'd made a mistake in this version and have too many brackets in my lists.) With many thanks (as ever!) to StackOverflow commenters, especially m09 in a 2011 thread.

between two connected points:

```
get_distance(Start, End, Distance) :-
  road(Start, End, Distance).
```

Start up the interpreter, load the file, and try the predicate out:

```
?- [travelling].
?- get_distance(a, b, Distance).
Distance = 3.
?- get_distance(a, d, Distance).
false.
```

To use the predicate, you put in two constants and a variable, **Distance**. Prolog will then return possible solutions for **Distance** (in this case, only one solution exists for each question). It works fine for the facts exactly as we put them in; but you can't reverse the order of the towns. As our roads run in both directions, we'll improve the rule:

```
get_distance(Start, End, Distance) :-
  road(Start, End, Distance).
get_distance(Start, End, Distance) :-
  road(End, Start, Distance).
```

Our predicate now has two clauses. A predicate consists of one or more clauses, and two clauses belong to the same predicate if they have the same functor (name) and the same arity (number of arguments), as is the case here. **get_distance(Start, End) :- CLAUSE** would be a different predicate as it has arity of 2, not 3. Prolog will try to solve both clauses in its attempt to find a solution; so if you reload the file and try the reversed roads query, **get_distance(b, a, Distance).** should now return 3.

You could also write this as

```
get_distance(Start, End, Distance) :-
  (road(Start, End, Distance) ; road(End, Start, Distance)).
```

The semi-colon acts as an 'or' operator.

Next, let's start on the travelling problem proper. First, let's tackle the case where there is a road between the start and end points:

```
get_route(Start, End, Visited, TotalDistance) :-
 get_distance(Start, End, Distance),
 reverse([End|[Start]], Visited),
 TotalDistance is Distance.
Load this and run a query:
?- [travelling].
?- get_route(a, b, Visited, TotalDistance).
Visited = [a, b],
TotalDistance = 3;
false.
```

Enter the semi-colon to continue Prolog's search for solutions. In this case there is only one solution.

So, what's that code doing? The first thing to notice is that the right-hand side of this rule is made up of several independent sentences separated by commas ("and" operators). The whole sentence here will be true if and only if all the sentences within it are true. In this case, the first sentence will return false unless there is a road between **Start** and **End**; if there is such a road, **Distance** gains a value, and the sentence is true.

The next sentence uses a SWIPL library predicate, **reverse/2**. This takes two lists, and returns true if the first list, reversed, is equivalent to the second list. However, in this instance, the variable **Visited** is not yet assigned. Prolog always tries to find a solution to the given constraints; a way to return true. Here, the obvious way to do that is to assign the reverse of the first list to **Visited**. Now they are equivalent and the sentence is true.

But what is this list? **[End|[Start]]** uses **Head|Tail** list notation. You can use this to decompose a list:

```
?- [1, 2, 3, 4] = [A | B].
A = 1
B = [2, 3, 4]
```

A is assigned to the **Head** of the list, the first element of the list; B is assigned to the **Tail** of the list, all the rest except for the first element.

But you can also use H|T notation to create a list. **[Start]** turns the variable **Start** into a list with a single member, then **[End|[Start]]** prepends **End** to that list. This gives the list **[End, Start]**, which we then reverse to give **[Start, End]** which is then assigned to

Unification

One of the basic ideas of Prolog is unification. The operator that symbolises unification is the equals sign, =, but the idea isn't quite the same as our normal concept of equals.

Two terms (constants, variables, or complex terms) unify if either:

- They are the same term; or
- They contain a variable that can be instantiated to make them the same term.

Here are some examples. **foo = foo.** is true, because the terms (in this case a constant) are the same. Similarly **15 = 15.** is true.

What about **foo = X.**? Well, X is a variable, so it can be instantiated to **foo**, in which case the terms can be unified. So that's what Prolog does. (You can think of Prolog as trying to solve for unity if at all possible – so it looks for a way to make the unity true and does that.) You'll get the output **X = foo.**. If you now type **bar = X.**, that's fine too. But **X = foo, X = bar.** will return false; you can't instantiate X twice in the same statement.

Prolog will also try to unify complex terms like predicates, if they have the same functor (name) and arity (number of arguments). Again, it will assign variables as necessary to make this work:

```
?- foo(X, Y) = foo(bar, baz).
X = bar
Y = baz.
```

Visited. In this instance, since you have only two list members, you could just as easily use the **Head|Tail** concatenation the other way around, and avoid the use of **reverse/2**. However, in general, prepending to a list is much easier and cheaper in Prolog than appending to a list. In later stages of development of this code, **Start** might already be a long list of previously-visited places, in which case prepending **End** is much easier.

The final sentence is easy: assign the value of **Distance** to **TotalDistance**, the variable passed into the rule at the start. This allows it to be output, as Prolog is solving for **Visited** and **TotalDistance. is** is an arithmetic operator used to compute the right-hand expression and assign (unify) it with the left-hand variable.

And that's it. However, this will only work if there is a single road between the start and end points. We want something that calculates a longer route if needed, so let's expand it into a recursive rule.

First, let's recast the rule above so that it'll work at the end of a longer recursive process:

```
get_route(Current, End, VisitedSoFar, DistanceSoFar, Visited,
TotalDistance) :-
 get_distance(Current, End, Distance),
 reverse([End|VisitedSoFar], Visited),
 TotalDistance is DistanceSoFar + Distance.
```

The logic of this is the same, but if we imagine that this is the final step of a route, it's clear that we will already have an ongoing distance tally, and we will already have a set of visited points. So if there is a road between **Current**, the point we're currently solving for, and **End**, the first sentence returns true, we go on to assign values to **Visited** and **TotalDistance**, and we're done.

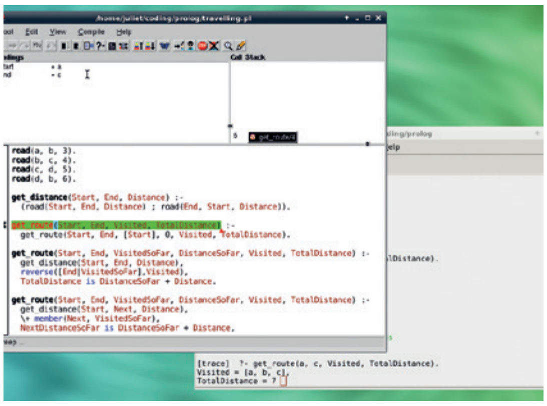
What about the recursive step?

```
get_route(Current, End, VisitedSoFar, DistanceSoFar, Visited,
TotalDistance) :-
 get_distance(Current, Next, Distance),
 \+ member(Next, VisitedSoFar),
 NextDistanceSoFar is DistanceSoFar + Distance,
 get_route(Next, End, [Next|VisitedSoFar], NextDistanceSoFar,
Visited,
TotalDistance).
```

The first line (the expression on the left-hand side of the rule) is identical to that of the rule above. As with **get_distance/3**, these are acting as two clauses of a single rule. Prolog will try the first one, then the second one, in its search for a solution.

The **get_distance** line has one assigned variable, **Current**, and two new variables, **Next** and **Distance**. Since these are unassigned, Prolog will try to find solutions to slot in; in this case, it will try to find values for **Next** such that the sentence evaluates true and there is thus a value for **Distance**.

However! The next line gives an extra constraint for **Next**: it must not already be a member of **VisitedSoFar**. **\+** in Prolog is the negation operator (also known as negation-as-failure). (Logical negation



```
road(a, b, 3).
road(b, c, 4).
road(c, d, 5).
road(d, b, 6).

get_distance(Start, End, Distance) :-
  (road(Start, End, Distance) ; road(End, Start, Distance)).

get_route(Start, End, Visited, TotalDistance) :-
  get_route(Start, End, [Start], 0, Visited, TotalDistance).

get_route(Start, End, VisitedSoFar, DistanceSoFar, Visited, TotalDistance) :-
  get_distance(Start, End, Distance),
  reverse([End|VisitedSoFar],Visited),
  TotalDistance is DistanceSoFar + Distance.

get_route(Start, End, VisitedSoFar, DistanceSoFar, Visited, TotalDistance) :-
  get_distance(Start, Next, Distance),
  \+ member(Next, VisitedSoFar),
  NextDistanceSoFar is DistanceSoFar + Distance.
```

```
[trace]  ?- get_route(a, c, Visited, TotalDistance).
Visited = [a, b, c],
TotalDistance = 7
```

You can step through the code slowly and find out exactly what Prolog is up to. Load the graphical debugger with **guitracer.** then start tracing with **trace.** – subsequent statements will fire up the tracer.

is a bit theoretically complicated, but for now just think of it as Prolog's version of **not**, which indeed was the older form of the same operator.) So we're looking for a solution to the first line where **Next** hasn't been visited before. (This avoids loops.)

Once we've found such a solution, we record the new distance in a temporary variable, **NextDistanceSoFar**, then recurse by calling **get_route/6** again, with **Next** in the place of the **Current** node, adding **Next** into the **VisitedSoFar** list (using **Head|Tail** notation again), and using the new distance tally. Note that **Visited** and **TotalDistance** pass through the whole process unchanged. This is so that in the end, once a route has been found, they can be returned to the user with their final values.

"If you get hooked, you too might find yourself trying to work out the logic behind language."

The final thing we need to do is to create an easy way to call this rule (to start the problem-solving process):

```
get_route(Start, End, Visited, TotalDistance) :-
 get_route(Start, End, [Start], 0, Visited, TotalDistance).
```

We seed the **VisitedSoFar** list with our starting point, and set **DistanceSoFar** as zero. Prolog will then go forth and try to find a solution or solutions within the given rules.

Load it and run it (eg **get_route(a, d, Visited, TotalDistance)** and see what happens.

If you want to take your Prolog experimentation further, we strongly recommend the excellent website **www.learnprolognow.org**. There's a free online version or a dead-tree book. You could also look at the Prolog debugger, which can give you a lot of information about what's actually going on when Prolog tries to find solutions for your code. Try extending the code above to solve the 'original' Travelling Salesman problem (the shortest possible tour), or to look through solutions for a minimum one. Who knows, if you get hooked, you too might find yourself trying to work out the logic behind language like Prolog's original designers.

Juliet Kemp is a scary polymath, and is the author of Apress's *Linux System Administration Recipes*.

LINUXVOICE TUTORIAL

CODE NINJA: SPREAD THE LOAD WITH GNU PARALLEL

BEN EVERARD

Speed up your shell scripts by sharing the load across all the under-utilised cores of your CPU.

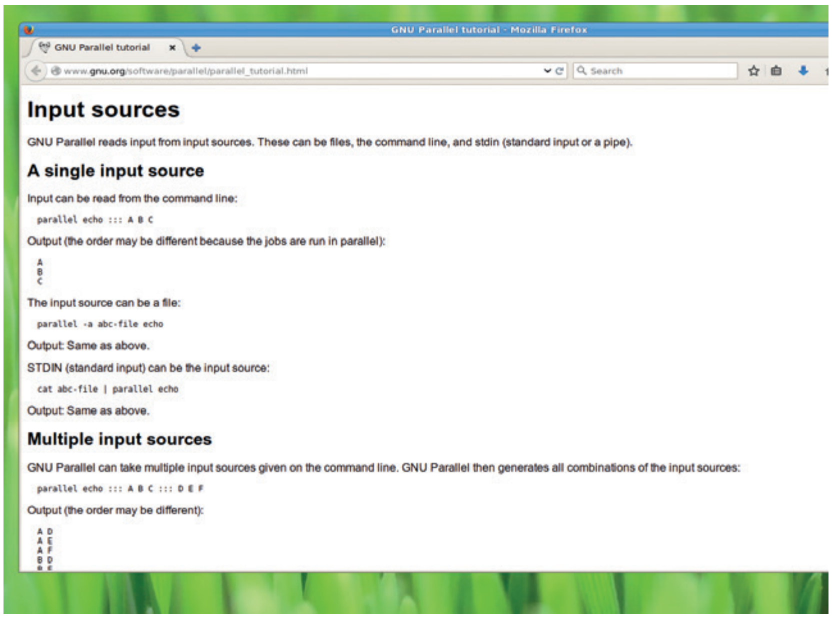
WHY DO THIS?
- Write scripts that run faster
- Instantly add parallelisation to your programs
- Fully utilise your multicore CPU

In this month's code ninja we're going to take a detailed look at an often overlooked command that can make your programs run faster. *Parallel* is a command for running the same program multiple times with different inputs. The idea is simple: we live in a world where almost every computer has a multicore CPU, yet very few programs can balance their work effectively across many cores. This means that you can often find yourself waiting for something to finish running, yet most of your CPU cores are idle.

Lets take a look at a simple example: compressing every subdirectory of the current directory into separate **tar.gz** files. This is useful when archiving files that you'll need later. This can be done with a simple script like so:

```
for dir in */
do
 tar -cfz ${dir%/} "${dir%/}.tar.gz"
done
```

The expression **${dir%/}** just returns the contents of the **dir** variable with slashes stripped away. This code loops through every subdirectory in turn, compresses it, and moves on to the next one only once the previous directory has been successfully compressed. Since the **tar** command only uses one CPU core at a time, this script will likewise only run on a single core leaving the rest of your computer idle.


GNU Parallel tutorial - Mozilla Firefox
GNU Parallel tutorial
www.gnu.org/software/parallel/parallel_tutorial.html
Search

Input sources

GNU Parallel reads input from input sources. These can be files, the command line, and stdin (standard input or a pipe).

A single input source

Input can be read from the command line:

```
parallel echo ::: A B C
```

Output (the order may be different because the jobs are run in parallel):

```
A
B
C
```

The input source can be a file:

```
parallel -a abc-file echo
```

Output: Same as above.

STDIN (standard input) can be the input source:

```
cat abc-file | parallel echo
```

Output: Same as above.

Multiple input sources

GNU Parallel can take multiple input sources given on the command line. GNU Parallel then generates all combinations of the input sources:

```
parallel echo ::: A B C ::: D E F
```

Output (the order may be different):

```
A D
A E
A F
B D
B E
```

The official *Parallel* tutorial is a very detailed, but quite dry, guide to the tool. **www.gnu.org/software/parallel/parallel_tutorial.html**.

A simple way to spread this load across all the cores on a machine is to start each **tar** command in its own process. This is done by adding an ampersand to the end of the **tar** line. With this in place, the code runs through every subdirectory and starts compressing it, then moves straight onto the next without waiting for the previous one to finish.

Another approach is to use a different compression program. *Pigz* is an implementation of *Gzip* that parallelises the compression for better performance. You can do this with the following script:

```
for dir in */
do
 tar -cf - ${dir%/} | pigz > "${dir%/}.tar.gz"
done
```

Introducing Parallel

Here's where we start using the **parallel** tool. This is a load-balancing program that executes tasks and spreads them out among the available CPU cores in the most efficient way. In some respects, this is similar to the version above where we put an ampersand at the end of the **tar** line; however, **parallel** doesn't just launch all the tasks simultaneously. Instead, it will launch one per CPU core and then when each is finished, it launches another on the free core. This approach keeps the load on all CPU cores evenly.

To compress our subdirectories in this manner, we need two parts. First, we need to write a script to list all the subdirectories that we want to compress (without the path or trailing slash):

```
for dir in */
do
 echo ${dir%/}
done
```

Then we can pipe the output of this into **parallel**. This will take each line of the input and use it as an argument to the **tar** command. Each time it, it replaces the **{}** symbols with the line read from input.

```
bash parallels-out.sh | parallel tar -czf {}.tar.gz {}
```

Don't fall into the trap of premature optimisation! For simple jobs, it would be easy to spend longer thinking of ways to parallelise them than you would actually save in computation. As well as simplicity, the single-core method has the advantage that it won't overload the computer's processor. By only using a single core and not distributing the work over all the cores, the first method won't lock up the machine if you run it on a large data set. Other processes can use the other cores and the machine should remain responsive.

The second method we looked at, which used simple forking to run each process in its own thread, should work well for most tasks. If the whole computation is going to take less time than it takes to look through the **parallels** or **pigz** man page, then this is definitely the best option for a quick speed boost.

Save time carefully

It's only worth investigating more complex operations if it's either a very complex computation, or if it's a computation that you have to perform many times.

The two most advanced forms of parallelisation we look at here (**pigz** and **parallel**) have different payoffs. **Pigz** should run faster on each file, so if you have fewer files than processor cores, **pigz** should be the quickest to run. The more files there are, the better **parallel** can balance the load across all the cores.

The biggest advantage that **parallel** has over **pigz** is that it's a standard method that you can apply to anything, where as **pigz** is a specilised tool that does just one task. This means that if you want to go down this route, you'll have to learn how to use the parallelised tools for every task rather than just remeber one (**parallel**), and use that in every situation.

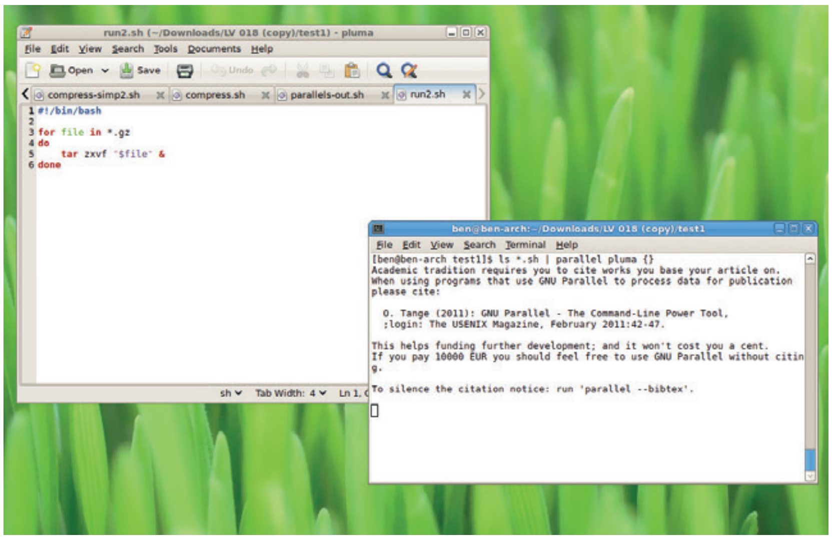
Another advantage of **parallel** is that you can use it to instantly parallelise any program you create. As long as you design your program so that you split up the data to be processed (as we did in our compression example by creating a script that put together the directories and filenames that should be processed) and the actual computation (which in the previous example was the compression tool), then you can just launch your program via **parallel**. This works regardless of the language that you wrote the program in. It is worth noting that interpreted languages can be slow to start, so this may not be a great solution if there are a lot of small computations to do.

Before we get too carried away with when to use **parallel**, let's take a closer look at how to use it. In our example, we piped out a series of lines of text, and each line contained just the directory name. This method of piping data to **parallel** is a common way of using the command, but it's not the only one. You can also put a series of inputs after three colons. As a trivial example, the following will print the numbers one, two and three (though not necessarily in that order):

```
parallel echo ::: 1 2 3
```

Tuning performance

By default, **parallel** will choose the optimum number based on the amount of cores in your CPU, however, there are times when you want to change this. For example, by leaving a core or two free, other programs can continue to run without too dramatic a slow down. This is done using the **-j** flag. For example, to run our previous command but using just two tasks at a time, use the following:


run2.sh (~/Downloads/LV 018 (copy)/test1) - pluma
File Edit View Search Tools Documents Help
Open
Save
Undo
compress-simp2.sh
compress.sh
parallels-out.sh
run2.sh
sh
Tab Width: 4
Ln 1, C
ben@ben-arch:~/Downloads/LV 018 (copy)/test1
File Edit View Search Terminal Help

```
#!/bin/bash

for file in *.gz
do
    tar zxvf "$file" &
done
```

```
[ben@ben-arch test1]$ ls *.sh | parallel pluma {}
Academic tradition requires you to cite works you base your article on.
When using programs that use GNU Parallel to process data for publication
please cite:

  O. Tange (2011): GNU Parallel - The Command-Line Power Tool,
  ;login: The USENIX Magazine, February 2011:42-47.

This helps funding further development; and it won't cost you a cent.
If you pay 10000 EUR you should feel free to use GNU Parallel without citin
g.

To silence the citation notice: run 'parallel --bibtex'.
```

It is possible to run parallel with graphical software (such as the Pluma text editor), although we can't think of a reason why you'd ever want to.

```
bash parallels-out.sh | parallel -j2 tar -czf {}.tar.gz {}
```

Clustering

The biggest speedup you can get running commands using **parallel** is by distributing the load around not only different cores, but different machines entirely. This means that you could launch a task from a lowly Raspberry Pi yet take advantage of the CPUs on a cluster of high-end servers. In order to take advantage of this, you first need to set up SSH certificate logins for each machine (see this month's Masterclass for details). Once that's done, you just need to create a file with a list of all the logins in the form **user@host**, with one per line. You can tell **parallel** to use this with the **--slf** flag.

"Another advantage of parallel is that you can use it to parallelise any program you create."

Since **parallel** usually operates on files, you need to tell it what you want it to do with the files that are on the local machine when the processing is on remote machines. The most common options are **--transfer --return <filename> --cleanup**. The **--transfer** option copies the local file to the remote server before starting the computation; **--return <filename>** copies the file with the name **<filename>** back again afterwards; and **--cleanup** deletes the files from the remote server once the processing has finished. You can use the shorthand **--trc <outfile>** to use all these together. These are used when the filename being worked on is the argument passed into **parallels**, so won't work in a situation like our example.

```
bash parallels-out.sh | parallel --slf loginfile --trc {}.tar.gz tar
-czf {}.tar.gz {}
```

There are many more options to **parallel**, and many different ways of triggering the same options, but using the ones we've looked at here you should be able to significantly speed up many of your scripts.

LINUXVOICE TUTORIAL

PROFILING: MAKE CODE RUN MORE QUICKLY

BEN EVERARD

Optimise your code by finding out which parts of it are swift and which bits are sluggish.

WHY DO THIS?
- Write programs that run faster
- Gain a better understanding of how your programs run
- Use a 2000-year-old algorithm to find prime numbers

If there's one thing everyone loves in a program it's speed. Sitting around waiting for a computer to do something isn't fun, and now in 2015 when we have multi-core computers that can perform tens of thousands of millions of operations a second, it really shouldn't happen.

We're going to take a look at a simple Python program for calculating prime numbers. In case you're unfamiliar with these, they're numbers that are only divisible by 1 and themselves. So, 2, 3, 5 and 7 are prime numbers, but 9 isn't. These numbers have quite a few quirky mathematical properties, including some very obscure maths that makes them useful to encryption. Our task for today, though, is simply to calculate them as quickly as possible.

We'll start with a simple Python program that can calculate the prime numbers between 1 and 1000:

```
max_num = 10000
import math
primes = []

def run1():
 i = 2
 while i < max_num:
  prime = True
  y=2
  while y < int(math.sqrt(i)):
   if (float(i)/float(y)).is_integer():
    prime = False
   y = y + 1
  if prime:
   primes.append(i)
  i = i+1

run1()

print primes
```

This code works quite simply. It loops through every number between 1 and the maximum, which is stored in the variable **i**. The program then loops through every number between 1 and the square root of **i** (the square root because every non-prime must be divisible either its square root or a smaller number). It then divides the first iteration by the second iteration (**y**). If the result is a whole number, the program has found a divisor, so **i** can't be prime.

Yes, we know it's not perfect...

You may well have spotted some things that can be improved. But we won't just dive in and start changing things. We'll do this methodically. First, we'll get a benchmark to see how it runs. We can do this using the time program. You can do this with:

```
time python primes1.py
```

Where **primes1.py** is a file containing the above program. This will output three different times: **real**, **users**, and **sys**. The **real** time is the time between starting the program and it terminating. The **user** time is the amount of time spent in userspace, and **sys** is the amount of time in system calls. Since we're interested in the total time the program takes, we'll look at the **real** values in this tutorial. On our test system, this program took 0.589 seconds. That's not bad, but we can make it faster.

Before looking at making it faster, let's find out what's making it slow. We can use a tool called a profiler to analyse how our program runs and see which parts of the code are taking the longest. Python does have a built-in profiler, but the one in the module **line_profiler** does a much better job for code like ours.

First you need to install the Python header files through your distro's package manager. These are usually in a package called **python-dev**. Once you've done that, you can install the profiler with:

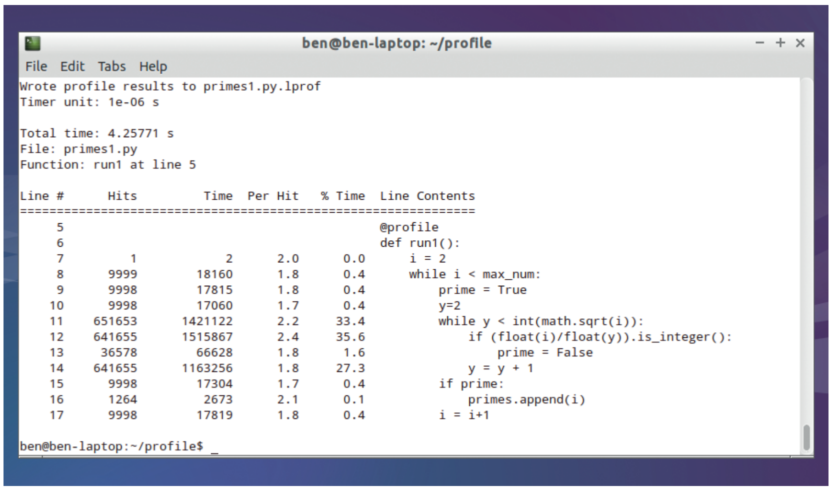
```
sudo pip install line_profiler
```

To use this, we first need to add a line directly above the function definition to tell the profiler that we want to investigate that function. It should now read:

```
@profile
def run1():
```

With this in place, you can profile the code with:

```
kernprof -vl primes1.py
```



```
Wrote profile results to primes1.py.lprof
Timer unit: 1e-06 s

Total time: 4.25771 s
File: primes1.py
Function: run1 at line 5

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
     5                                           @profile
     6                                           def run1():
     7         1            2      2.0      0.0      i = 2
     8      9999        18160      1.8      0.4      while i < max_num:
     9      9998        17815      1.8      0.4          prime = True
    10      9998        17060      1.7      0.4          y=2
    11    651653      1421122      2.2     33.4          while y < int(math.sqrt(i)):
    12    641655      1515867      2.4     35.6              if (float(i)/float(y)).is_integer():
    13     36578        66628      1.8      1.6                  prime = False
    14    641655      1163256      1.8     27.3              y = y + 1
    15      9998        17304      1.7      0.4          if prime:
    16      1264         2673      2.1      0.1              primes.append(i)
    17      9998        17819      1.8      0.4          i = i+1

ben@ben-laptop:~/profile$ _
```

The line profiler's output makes it easy to see which lines in our program are slowing us down.

This will run the program, then let you view how much time each line in the program took. The table that it outputs is quite revealing, and the % time column shows that it's the inner loop that takes the most time, specifically, the lines:

```
while y < int(math.sqrt(i)):
    if (float(i)/float(y)).is_integer():
        prime = False
    y = y + 1
```

Between them, these make up 33% of the number of lines in the program, but they take up 97.9% of the total running time. In other words, it doesn't matter what we do with the rest of the program – unless we can get these running faster, it's not going to make much difference.

Low-hanging fruit

There's a reason that these lines take so much time. They're run far more frequently than any other line because they're the inner loop. For every iteration of the outer loop, these run many times.

Of these lines, the slowest is the **if** statement. In order for it to run, it has to do four things: convert **i** from an **int** to a **float**; convert **y** from an **int** to a **float**; divide **float(i)** by **float(y)**; and see if the result is a whole number. That's quite a lot of processing for the simple goal of checking if one number is a divisor for another, and since this accounts for 35% of the total time, it's worth trying to find another way.

Mathematically, this operation is the same as dividing one integer by another and checking whether there's not a remainder to the division. This can be done using the modulo operator. This slightly obscure operator returns the remainder of a division, not the actual result, and it's performed with the percent symbol. The **if** line can be changed to:

```
if i % y == 0:
```

Running the profiler again, you should see that the speed has increased, but this line is still the source of a lot of the speed simply because it's run so many times. We can now turn our full attention to the **While** loop.

Generally, when programming, it's best to use the standard features of the language as much as possible. Doing this means you can program more quickly (because you have to write less code). It usually also makes your code faster and less error-prone, because the standard parts of most languages are written by people who are experts in the field, and they usually expend a lot of effort to make sure each part of the language runs as quickly as possible. In interpreted languages such as Python there is even more of an advantage, because the standard parts of the language are compiled into native code while any code you write is interpreted. The speed differences can be huge.

"In general it's best to use the standard features of a language as much as possible."

The code:

```
y = 2
  while y < int(math.sqrt(i)):
    y = y + 1
```

Is functionally equivalent to:

```
for y in range (2, int(math.sqrt(i)):
```

Both of these will iterate through the whole numbers from 2 to the square root of **i**, but the second one will do all the processing in highly optimised native code, while the first one does it in unoptimised interpreted Python.

As well as finding a way to calculate prime numbers, Eratosthenes calculated the earth's circumference without leaving Egypt (he was out by just 16%).

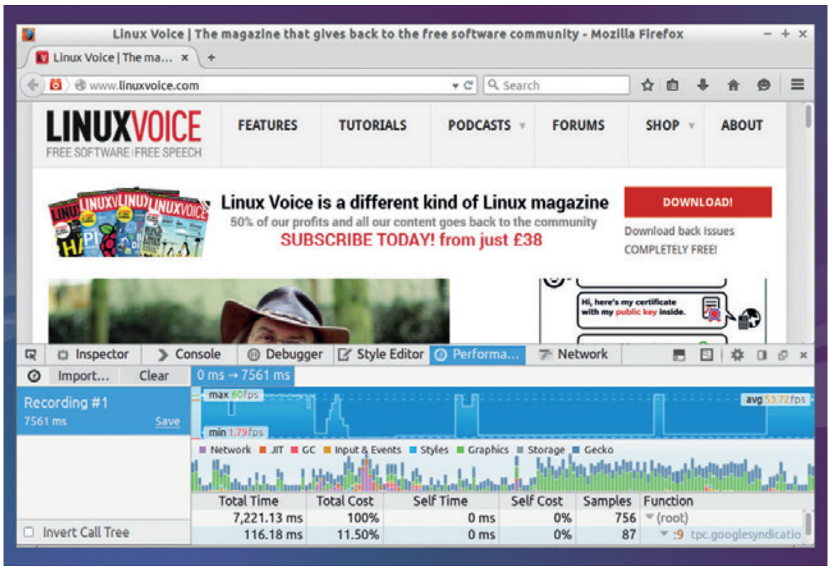
Profiling in other languages

In this tutorial, we've looked a profiling using the popular Python language, but almost all languages have at least one profiling tool to help you speed up your code.

Gprof is the most famous profiling tool for Linux (and other Unix-like systems). It's most commonly used with C and C++ code that's been compiled using *GCC*. You can instruct the compiler to insert profiling code into executables when they're built, and then use these to analyse the performance of the code in a similar way to the profiling we've done in this tutorial.

JavaScript is, like Python, an interpreted language, so you need a profiler that works with the interpreter. Most major browsers (including *Firefox* and *Chrome*) have profilers built in to their JavaScript engines so web developers can keep track of their pages performance from within the browser.

It's not just programming languages that benefit from profiling. Database engines often come with performance analysis tools. In the case of *MariaDB* and *MySQL*, this is the slow query log that can be used to track which queries are performing poorly.



```
  Total Time   Total Cost   Self Time   Self Cost   Samples   Function
 7,221.13 ms         100%        0 ms          0%       756   ▾ (root)
   116.18 ms       11.50%        0 ms          0%        87     ▾ :9 tpc.googlesyndicatio
```

The profiler in *Firefox* keeps track of the performance of JavaScript, and the web browser in general.

On our test machine, the version with the **while** loop ran in 0.33s, while the version with the **range** ran in 0.1s. That's running over 300% faster just by changing the type of loop!

Skipping steps

We've now made each line run much faster, but a major problem for our code is simply the number of times that the loops are running. Even a really efficient line can be a major slowdown if it's running millions of times. Here we have to look at the nature of the calculation that's going on to see where speedups may lie. The more you understand about the problem you're solving with your code, the better you will be able to optimise your algorithms.

One fairly simple fact about prime numbers is that no even number larger than 2 can be prime. This is simply by definition, since an even number has to be divisible by 2. Another slightly less obvious fact is that any odd non-prime number has to be divisible by an odd number. This is because of the property multiplication that says that an odd number multiplied by an even number is odd, while an odd number multiplied by an odd number is even and an even number multiplied by an even number is even. The result of all this is that we can cut into a quarter the number of times that the inner loop must run by eliminating any unnecessary loops through even numbers. The code can be changed to:

```
max_num = 10000
import math
primes = [2]

#@profile
def run1():
 for i in range(3, max_num, 2):
  prime = True
  for y in range(3,int(math.sqrt(i)), 2):
    if i % y == 0:
      prime = False
      break
 if prime:
   primes.append(i)

run1()
print primes
```

You can see here that we've also changed the first **while** loop to a **for** loop to increase speed as well, though this has a much smaller speed increase than the inner loop change. Another optimisation we've included is the break. This is because we can stop checking a number after we've found one divisor. It doesn't matter whether or not there are more: as long as there's one, it's not prime.

With these speedups in place, the code runs in 0.036 seconds. That's another 300% speed increase over the previous 300% speed increase. With just a few simple optimisations, we've been able to make our prime number calculator run faster by a factor of 10. That's pretty good going!

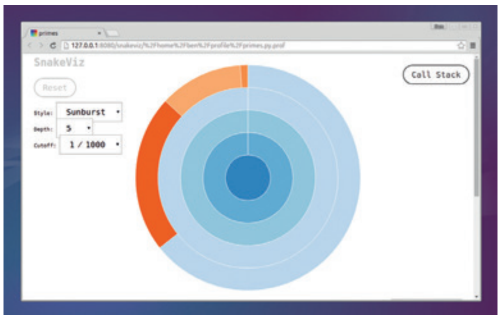
Excess speed

So far, our optimisations have led to our code becoming a bit more readable. However, this isn't always the case. You'll often find that pushing for gains can lead to code that's more complex and confusing. This can make it hard to maintain as you may not fully remember how the optimisations work if you have to come back to the code later.

In the previous example, complexity has crept in a little bit because, by optimising out the even numbers, we've introduced a special case for the number 2. The first prime now has to be added manually to the list of primes. This is a fairly simple case, and with the judicious use of comments, you should be able to explain it (optimisations are something that should always be commented).

cProfile

In this tutorial, we've used a line profiler, but that's not the only option. Python's default profiler, *cProfile*, is also a powerful tool for understanding your code. However, instead of working on a line-by-line basis, it works on a call-by-call basis, so it's great for working out which functions and methods are taking the most time to run. Take a look at the Python docs for details on how to use it (**https://docs.python.org/2/library/profile.html**).



The *SnakeViz* tool can be used to make diagrams out of the output from *cProfiler*.

Let's look if we try to optimise it further.

The algorithm we're using to find primes is probably the simplest one possible. It just checks every number against every possible divisor, and if it doesn't find a divisor, it classes the number as prime. It's so simple that a new programmer coming to the code should be able to tell what's going on almost straight away. It isn't however, the fastest method for calculating primes. Take, for example, the following code:

```
import math

max_num = 1000000
primes_list = [True] * max_num

for i in range(2, int(math.sqrt(max_num))):
  if primes_list[i]:
          for j in range (i*i, max_num, i):
                  primes_list[j] = False
                  break

out_list = []
for i in range(i,max_num):
  if primes_list[i]:
          out_list.append(i)

print out_list
```

If you run it, you'll find that it outputs all the primes between 1 and a million, and it's fiendishly fast. If we remove the **print** lines (which take up most of the running time), and compare it with the previous code (again with 1,000,000 as the maximum and the **print** lines removed), this code runs in 0.48s (on our test machine) compared to 5.67s for the other code. That's a massive speed up, but how on earth does it do it? If you fancy a maths challenge, try to work it out before reading on.

Ready? Let's carry on...

First, the algorithm starts by assuming each number is a prime until it finds that it's not. The **primes_list** variable holds a True or False value for each number, and we'll set them to False when we find out if a number is not a prime.

Starting with the number 2, and working up to the square root of the maximum, if the number is marked as a prime, it loops through the numbers **i*i**, **i*i+i**, **i*i+2i**, **i*1+3i** until this number gets larger than the maximum number. The program marks every number in this sequence as not prime since it's divisible by i.

It should be fairly obvious that every number marked as not a prime by this isn't a prime, but what's not immediately clear is why we can be sure that every number that's not a prime is marked as such, leaving only the prime numbers remaining. It's crucial to realise two things: every non-prime has at least one prime divisor (actually, they have at least two, but that's not necessary here), and we start our loop with the smallest prime, 2. We need to find any numbers that are multiples of 2. However, since we are looping up from the smallest, we only need concern ourselves with numbers of the form 2*x where x itself is 2 or larger. This is why we start our inner loop with i*i. Any numbers of that form where x is smaller than the first number can be ignored, because we will have found them (if they exist) in an earlier iteration of the loop.

Then, we take the next smallest prime, 3. Anything smaller the 3*3 that's not a prime must have already been marked (for example 6, which was marked off in the first iteration since it's a multiple of 2), so the algorithm then checks off all the multiples of 3 larger than 9. The algorithm then continues until it's reached the square root of the maximum, since no number smaller than the maximum can have a smaller smallest divisor than the square root of the maximum.

This method is called the sieve of Eratosthenes, and was discovered by Eratosthenes of Cyrene circa 200BC. It's much faster (around 10 times) faster than our original method, but it's also much harder to understand. Indeed, even after reading the previous explanation, you may still have difficulty fully understanding what's going on. There are some even more efficient methods of calculating primes that are even more complex.

The problem we have here is that we've optimised for speed, but in doing so have sacrificed readability and the understandability of the code. If you do this at every stage in your programs, you'll very quickly end up with an unmaintainable mess. You need to target just the parts that are most worth optimising for speed, and optimise the others for readability instead.

"The problem here is that we've optimised for speed, but in doing so have sacrificed readbility."

Keep it sensible

Donald Knuth, legendary programmer and author of the epic book *The Art of Computer Programming* advises, "We should forget about small efficiencies, say about 97% of the time: premature optimisation is the root of all evil. Yet we should not pass up our opportunities in that critical 3%."

Note the term premature optimisation. Back at the start of this tutorial, we used a profiler to find the parts of our code that were running the slowest. You can do the same with whole projects. If we did this on our program that included the prime generation code, and discovered it only needed the first 100 primes once and never again, it would be foolish to include the sieve of Eratosthenes, as doing so would save your code fractions of a second, at a huge cost to maintainability. On the other hand, if it's spending 10% of its time calculating prime numbers, then it would be equally foolish to throw away a significant speedup for a few lines of complicated code. Just make sure they're properly commented.

Ben Everard always comments his code, though as we can't read his handwriting, it's often not much help.

LINUX VOICE MASTERCLASS

Ferry files like a master with an oft-overlooked protocol that changed the nature of the web when it debuted.

LEARN THE SECRETS OF REMOTE ACCESS WITH SSH

The ubiquitous login tool has more than a few tricks up its sleeve to help experienced users.

BEN EVERARD

SSH is one of the most familiar Linux commands to many people. In its simplest use, it's a way of getting command line access to a remote computer, however this isn't all it does. Even if you're familiar with its basic use, it's still worth taking a little time to investigate it fully.

In order to use **ssh**, the remote host needs to have the server software installed (you can also log into localhost using **ssh** if you want to try out some options). This varies between distributions, so check the docs for details. On Ubuntu-based systems, it can be done with:

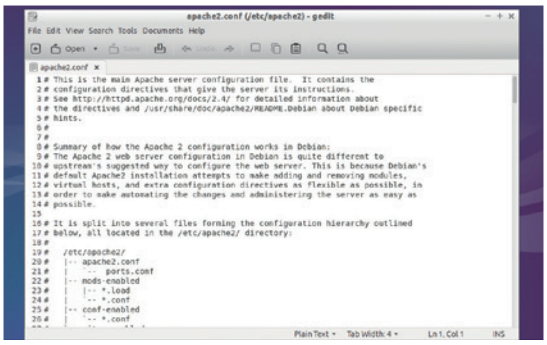
```
sudo apt-get install openssh-server
```

You'll also need the **ssh** client on the machine you're logging in from, but this is almost always installed by default (Arch users should read the wiki).

The most basic usage of the **ssh** command is:

```
ssh user@host
```

Where **user** is your username on the remote host, and host is the IP address or hostname of the machine you want to log into. If you have certificates set up for that use on that host, it will log you in automatically (see boxout), if not, it will prompt you for a password. Once it's connected, it will present you with command


apache2.conf (/etc/apache2) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
apache2.conf
Plain Text
Tab Width: 4
Ln 1, Col 1
INS

```
# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See http://httpd.apache.org/docs/2.4/ for detailed information about
# the directives and /usr/share/doc/apache2/README.Debian about Debian specific
# hints.
#
#
# Summary of how the Apache 2 configuration works in Debian:
# The Apache 2 web server configuration in Debian is quite different to
# upstream's suggested way to configure the web server. This is because Debian's
# default Apache2 installation attempts to make adding and removing modules,
# virtual hosts, and extra configuration directives as flexible as possible, in
# order to make automating the changes and administering the server as easy as
# possible.

# It is split into several files forming the configuration hierarchy outlined
# below, all located in the /etc/apache2/ directory:
#
#       /etc/apache2/
#       |-- apache2.conf
#       |       `--  ports.conf
#       |-- mods-enabled
#       |       |-- *.load
#       |       `-- *.conf
#       |-- conf-enabled
#       |       `-- *.conf
```

Need to do some remote admin but don't like command line text editors? **ssh** with X forwarding to the rescue!

Certificates

When you first install an SSH server, you'll be able to log in with your system password. However, passwords are fundamentally limited by our ability to remember them. Our weak, squishy brains struggle to hold enough information for properly secure passwords, and attackers routinely scour the internet for servers with SSH open and try to guess these passwords.

The best solution to this is to remove the squishy part of the problem. Rather than having to remember passwords, we can store the details needed for authentication on our computers. We don't just mean that you should copy and paste your password into a text file, but create a certificate that enables you to securely log into an SSH server. This requires a little bit of setup on both the client and the server.

The first thing you need to do is on the client (the machine you log in from). You need to generate a new SSH identity with:

```
ssh-keygen -t rsa
```

This will create key files in the **.ssh** directory in your home folder. In order to be able to log in to the remote machine, you just need to copy your newly created public key into the **authorized_keys** file on the remote host. You can even use SSH to make this easier!

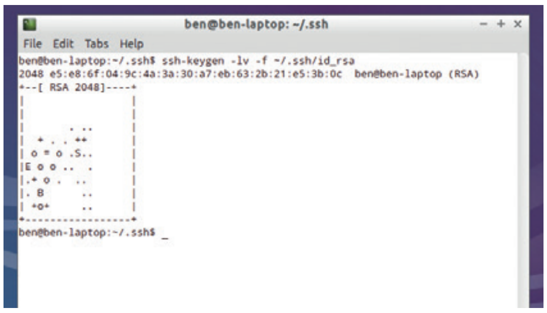
```
cat ~/.ssh/id_rsa.pub | ssh user@host "mkdir -p ~/.ssh && cat >>  ~/.ssh/authorized_keys"
```

With this set up, you'll no longer be asked for a password when you log in. Remember, though, that the client now has access to the server, so if it's a laptop and it gets stolen, you'll need to remove the key from the **authorized_keys** file to lock out the thief. If the remote server is important, you should also consider encrypting every machine that has certificates to access it.

You can now disable password logins to make the system more secure. This will block everyone from logging in via SSH using a password, so make sure none of your users rely on this before moving on.

Just insert the following line in the **/etc/ssh/ssh_config** file, and restart the SSH service (or reboot the computer):

```
PasswordAuthentication no
```



```
ben@ben-laptop:~/.ssh$ ssh-keygen -lv -f ~/.ssh/id_rsa
2048 e5:e8:6f:04:9c:4a:3a:30:a7:eb:63:2b:21:e5:3b:0c  ben@ben-laptop (RSA)
+--[ RSA 2048]----+
|                 |
|                 |
|          . ..   |
|   + . .  ++     |
|  o = o .S..     |
|E o o ..   .     |
|.+ o .   ..      |
|. B        ..    |
|  +o+      ..    |
+-----------------+
ben@ben-laptop:~/.ssh$ _
```

Each **ssh** key has an associated bit of art. This is to make it easier for people to tell if the key of a server has changed, but we don't know anyone who would notice.

line access to the remote machine. This standard method is great when you're sitting at a computer, but you can't use it easily in scripts. The solution here is just to add a new argument that's the command you want to run on the remote machine. For example, you can find out the disk usage of a remote machine with:

```
ssh user@host "df -h"
```

X forwarding

You don't have to restrict yourself to just command line programs. SSH also enables you to run graphical programs on the remote system but display the output locally with the flag **-**X. For example, you could edit a configuration file on a remote server using the graphical *Gedit* text editor with:

```
ssh -X user@host gedit
```

Fail2ban

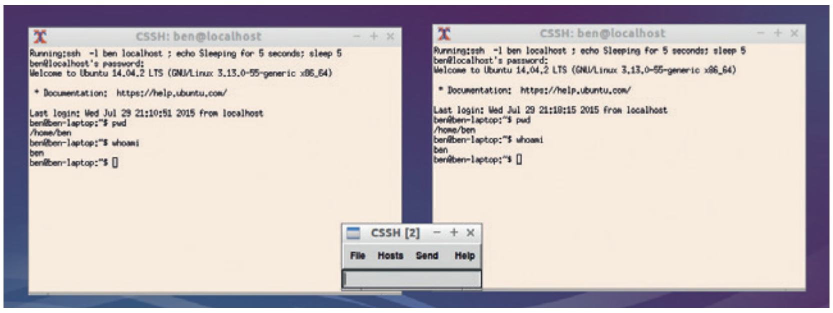
If you can't disable password SSH logins (see the box on Certificates), then it's a good idea to add a little more security to SSH through *Fail2ban*. This tool monitors SSH connection attempts, and if it finds someone repeatedly trying to log in, it blocks them for a specified length of time. Although this doesn't give as much security as certificates, it should prevent an attacker brute-forcing any but the very weakest of passwords (though this is no excuse not to use secure passwords!).

You should find *Fail2ban* in your distro's repositories, but it can be a little tricky to configure (although depending on your distro, the defaults may be fine). The file **/etc/fail2ban/jails.conf** holds all the options. *Fail2ban* monitors services that are set up with jails. These jails are specified through sections in the **jails.conf** file. For example, the following is the default jail for SSH on Debian:

```
bantime  = 600
findtime = 600

[ssh]
enabled  = true
port          = ssh
filter   = sshd
logpath  = /var/log/auth.log
maxretry = 6
```

This will monitor the log for anyone who tries and fails to log in six times in 600 seconds (set in the **findtime** option). Anyone who triggers this will be banned for 600 seconds (the **bantime**). After this time they can try again.

Fail2ban isn't just for SSH, and can be used to protect many services. Take a look at the documentation at **www.fail2ban.org/wiki/index.php/MANUAL_0_8** for more.



```
Running:ssh  -l ben localhost ; echo Sleeping for 5 seconds; sleep 5
ben@localhost's password:
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-55-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Last login: Wed Jul 29 21:10:51 2015 from localhost
ben@ben-laptop:~$ pwd
/home/ben
ben@ben-laptop:~$ whoami
ben
ben@ben-laptop:~$ []
```

```
Running:ssh  -l ben localhost ; echo Sleeping for 5 seconds; sleep 5
ben@localhost's password:
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-55-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Last login: Wed Jul 29 21:18:15 2015 from localhost
ben@ben-laptop:~$ pwd
/home/ben
ben@ben-laptop:~$ whoami
ben
ben@ben-laptop:~$ []
```

Using *Cluster SSH*, you can run the same command on multiple machines (or the same machine multiple times if you must).

A side advantage of running commands via SSH is that you can use it to stream files between machines. For example:

```
cat myfile | ssh user@host "cat - > myfile-remote"
```

Of course, it's usually best to use SCP to securely copy files between computers, but there are occasions where SSH can be a better choice. Usually, this is when the file itself doesn't yet exist, and using SSH rather than SCP, you can avoid creating the file locally in the first place. For example, you can stream a hard drive image directly to a backup server with:

```
dd if=/dev/sda | ssh user@host 'dd of=sda.img'
```

Similarly, data can be streamed the other way, so you could restore this image with:

```
ssh user@host 'dd if=sda.img' | dd of=/dev/sda
```

SSH can also be used to connect computers for purposes other than running commands. It is, at its heart, just a system for securely sending data between two computers. So far, we've seen how this can be used for commands and files, but it can be anything, and SSH has a whole host of options to make it easy to set up. The **-D <portnumber>** option is used to set up a SOCKs proxy, which can be used to channel web browsing data through another machine. To do this, first create the SOCKs proxy with:

"SSH can be used to connect computers for purposes other than running commands."

```
ssh -D 12345 user@host
```

Then you can connect your web browser to this by changing the proxy settings to **localhost** and port **12345**. Once this is done, your browsing will appear to be coming through the remote machine. This can be a useful way of accessing an intranet, or accessing a web page from a different part of the world.

We'll finish our tour of SSH with something that's not a core part of SSH, but a useful feature of some implementations: the ability to run commands on more than one machine a once. There are a few SSH clients that enable you to do this such as *Cluster SSH* (**https://github.com/duncs/clusterssh/wiki**). Using this tool, you get a terminal window for every machine you're logged into, but also a cluster window, and anything you type into the cluster window gets sent to every machine. If you need to upgrade lots of machines, you can simply enter it once rather than having to log into each machine and perform the operation manually.

REMOTE MANAGEMENT WITH WEBMIN

Connect to and administer remote machines quickly and securely.

BEN EVERARD

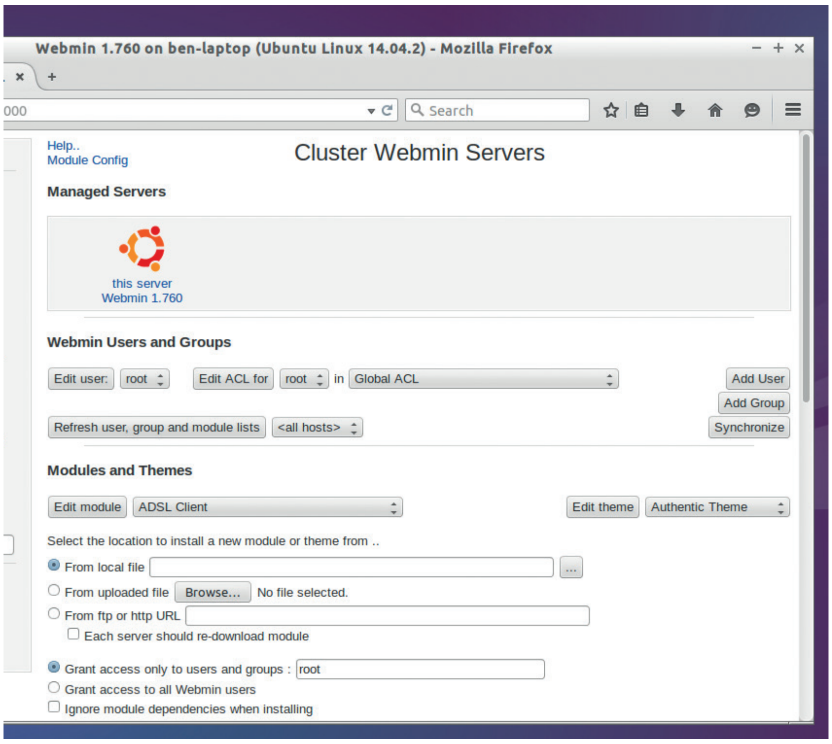
Webmin is a great tool for remotely administering a server without having to go near the command line. Unlike SSH, which just provides access to the remote machine, *Webmin* is as the name cunningly suggests, a web-based administration tool.

Webmin is available for most distros, but it's not always in the repositories, so if you can't find it via your package manager, you'll find RPM and Deb files on the project website, **http://webmin.com**.

Webmin comes bundled with its own web server and runs on port 10000 by default, so it shouldn't interfere with any other services that are running. Once you've installed it, you can log in by pointing your web browser to **https://localhost:10000** and entering the credentials of any user that has sudo privileges. It will use a self-signed SSL certificate, so you will get an SSL warning. If you're using *Webmin* on an untrusted network, it's a good idea to set it up with a proper certificate so you can be sure that there's not a man-in-the-middle attack, since that could give attackers complete control over your server. Once in, you'll see a web page with a menu on the left-hand side that contains a wide range of areas that can be administered. *Webmin* is module-based, so the options you see will depend on the modules you have installed. By picking the modules you need, you can customise *Webmin* to your taste. The list of modules is ordered into categories to make it easier to find the right module for a particular task.

"By picking the modules you need, you can customise Webmin to your taste."

The clustering feature of *Webmin* can make administering groups of servers as easy as administering one.


Webmin 1.760 on ben-laptop (Ubuntu Linux 14.04.2) - Mozilla Firefox

Help..
Module Config

Cluster Webmin Servers

Managed Servers

this server
Webmin 1.760

Webmin Users and Groups

Edit user: root Edit ACL for root in Global ACL Add User Add Group

Refresh user, group and module lists <all hosts> Synchronize

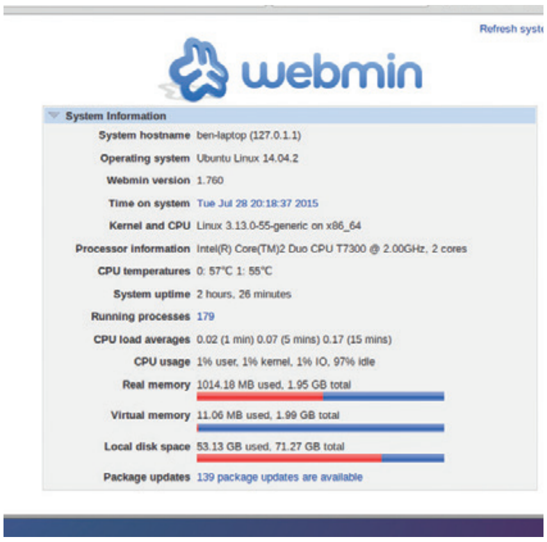
Modules and Themes

Edit module ADSL Client Edit theme Authentic Theme

Select the location to install a new module or theme from ..

- (●) From local file
- ( ) From uploaded file [Browse...] No file selected.
- ( ) From ftp or http URL
  - [ ] Each server should re-download module

- (●) Grant access only to users and groups : root
- ( ) Grant access to all Webmin users
- [ ] Ignore module dependencies when installing



System Information

System hostname	ben-laptop (127.0.1.1)
Operating system	Ubuntu Linux 14.04.2
Webmin version	1.760
Time on system	Tue Jul 28 20:18:37 2015
Kernel and CPU	Linux 3.13.0-55-generic on x86_64
Processor information	Intel(R) Core(TM)2 Duo CPU T7300 @ 2.00GHz, 2 cores
CPU temperatures	0: 57°C 1: 55°C
System uptime	2 hours, 26 minutes
Running processes	179
CPU load averages	0.02 (1 min) 0.07 (5 mins) 0.17 (15 mins)
CPU usage	1% user, 1% kernel, 1% IO, 97% idle
Real memory	1014.18 MB used, 1.95 GB total
Virtual memory	11.06 MB used, 1.99 GB total
Local disk space	53.13 GB used, 71.27 GB total
Package updates	139 package updates are available

The welcome screen to webmin shows you how your machine's resources are currently being used.

Performing backups

Let's take a look at one of the most important aspects of running a system: backups. There are loads of backup systems available, and if you've already got a

Usermin

Usermin is a stripped down version of *Webmin* designed for non-root users. It lets people perform all the usual tasks that they may need to perform for a regular user rather than the system-wide changes that a root user may need to make. *Usermin* follows the same setup as *Webmin*, but with far fewer modules, and none of the modules can make system-wide changes.

For example, a few of the default *Usermin* modules are:

- **Disk Quotas** where you can view the amount of disk space you're currently allowed to use.
- **SSH Configuration** allows you to add authorised keys to your SSH account.
- **Scheduled Cron Jobs** is used to edit the tasks that the cron daemon runs at set times.

The intended setup is that both *Webmin* and *Usermin* are on the same system, and any particular user has access to the one that suits their access level.

Virtualmin

Don't be confused by the name: *Virtualmin* has little to do with virtualisation. It's an extension to *Webmin* designed for setting up and running virtual servers in the sense that *Apache* uses the term 'virtual server', not in the sense of a server running on a virtual machine. This is where different users have accounts on the same machine that they can use to host their own websites, usually with associated FTP accounts and other such access tools. Unlike a virtualised server, all virtual servers run on the same distro. We don't know who decided to name two similar technologies the same, but we really wish they hadn't.

Virtualmin is more specialised than either *Webmin* or *Usermin*, and it's mostly for people administering large numbers of websites, especially if each website is being run by a different person.

system you're happy with, there's no reason to change it. If, however, you don't, then *Webmin*'s backup and restore is one of the easiest to set up, especially for people who aren't comfortable on the command line.

The backup options are in the System > Filesystem Backup module. On the first screen, you just select the directory that you want to backup. There's a box you can check to store the backup in TAR format, which will make it easy to use with other tools should you wish. On the second screen, you can set a wide variety of options to make sure your backup is running just how you want it. You can automatically transfer the backup to another server via FTP or SSH; you can schedule the backup to take place at a certain time; you can email someone once the backup has completed; and more. Experienced admins will probably realise that all this is possible via a few commands, but the point of *Webmin* isn't that it makes things possible, but that it makes things easy, and avoids the need to memorise command options if you don't want to.

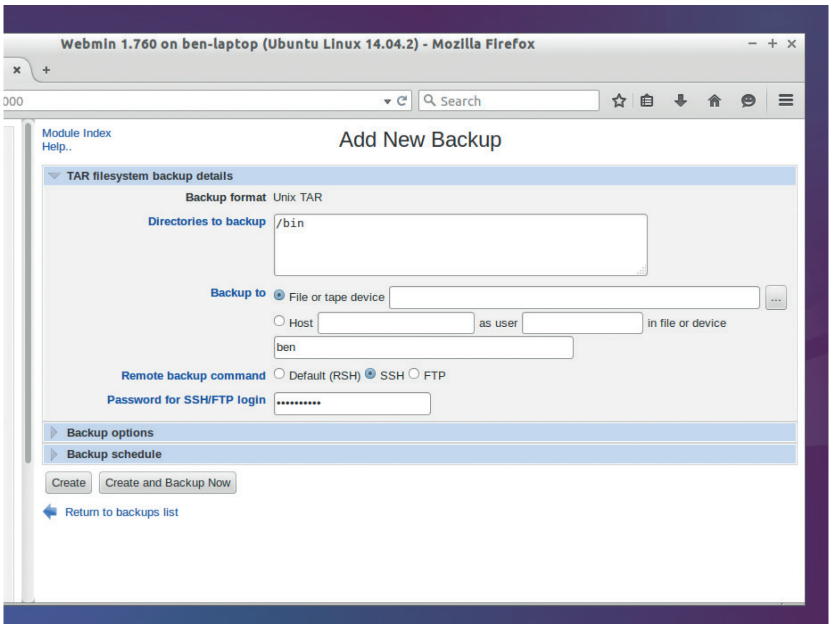
Webmin is designed for an administrator remotely managing servers, but it's actually quite good as a general computer admin tool as well, and you can use it to perform day-to-day tasks, for example, installing software via your package manager.

Package management

The package management module is at System > Software Packages (there's also System > Software Package Updates for grabbing the latest versions of software). Here you can do all the usual package administration tasks like installing new software, removing software, and even a distribution upgrade.

There are already graphical front-ends to *rpm* and *apt-get*, but *Webmin* brings them into a centralised system that you can use for much of your system maintenance. Any OpenSUSE or Mandriva users will already be used to having such a centralised system, but it can be quite a novelty to those of us accustomed to different distros.

One advantage of *Webmin* over standard command line tools is the ability to cluster servers together.


Webmin 1.760 on ben-laptop (Ubuntu Linux 14.04.2) - Mozilla Firefox

Add New Backup

TAR filesystem backup details

Backup format: Unix TAR

Directories to backup: /bin

Backup to: [✓ File or tape device] / Host ___ as user ___ in file or device: ben

Remote backup command: Default (RSH) [✓ SSH] FTP

Password for SSH/FTP login: ••••••••••

By using the TAR format for backups, you can be sure that you can restore them even if *Webmin* isn't working.

Using this feature, you can quickly configure a large number of servers without having to repeat the configuration steps on each machine. A similar, but slightly more complicated approach can be done with *Cluster SSH*, as seen on the previous page.

In the Cluster menu, you'll see options to copy files, run commands, update software and more on groups of machines at the same time. Before you do that, though, you have to connect your current instance of *Webmin* to the machines you want to administer. To do this, go to Cluster > Cluster Webmin Servers and enter the details of the machines. Once they're all connected, you will see them appear in the lists on the various cluster modules. For example, in Cluster > Cluster Software Packages, you can install the same software on a range of servers with just a few clicks.

The main benefit: discoverability

One big advantage of *Webmin* for novice system administrators is that it's easy to discover what's possible. If you're faced with just a blank command line, it can be hard to work out what you can achieve with which commands. For example, if you don't know how to change groups, man pages are a good start, but they only work if you know what the commands are. Using *Webmin*, you just need to navigate through the menu to System > Users And Groups, and you can quickly see all the groups, users and actions. Of course, it's a good idea to learn the commands as well, as they often have more flexibility than *Webmin* as well as the ability to script, but that doesn't mean that there's not a place for a web-based system in your system administration toolkit.

Ben Everard is the Ed Sheeran lookalike behind all of Linux Voice's tech infrastructure.

/DEV/RANDOM/

Final thoughts, musings and reflections

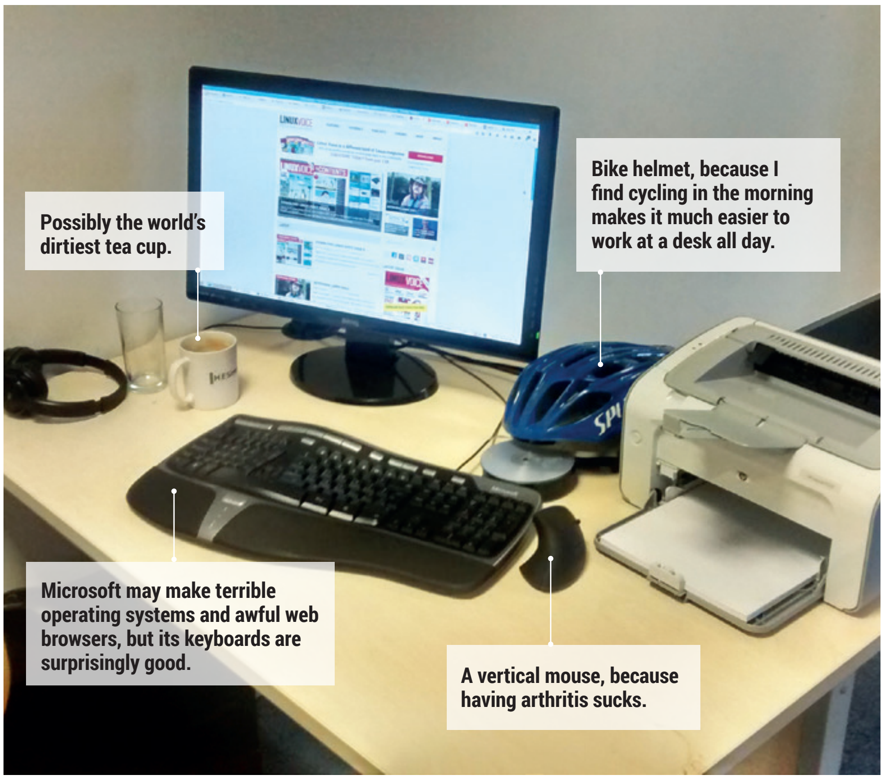
Nick Veitch was the original editor of Linux Format, a role he played until he got bored and went to work at Canonical instead. Splitter!

A few months ago, my boiler broke. In the grand scheme of hardships, this is not a great one, particularly as I have a service contract: the duly appointed engineer arrived, limboed under the beams in my attic and replaced the errant component – he actually had one in the van, for once. We had a brief discussion about it, because my boiler breaks about once a year (I must have upset the gods of central heating somehow) and I likened it to the fabled ship of Theseus, as I believe the front cover is the only part which is original.

On this occasion, it was the main circuit board (the third incarnation, I recall), which apparently would have cost about £100. It looks like a loose collection of relays, a few logic ICs and some power circuitry, with a few sensor circuits thrown in. To be fair, in limited numbers, it probably is expensive to build. But why in limited numbers? Why do things such as this, and a whole collection of other devices littering my home, have bespoke "brain" circuitry which does the same job? Dishwashers, washing machines, microwaves, tumble driers – they all have some 'brain' circuit connected to a handful of actuators and sensors to perform their servile tasks.

If you abstract the I/O part, they could probably all run comfortably from something mass produced and perhaps ruggedised into the bargain for an attractive fraction of the cost. Why not go further and invest in some sort of open hardware standard? Imagine a world where all the brains of all the things were the same!

It may be an overspecced part for some applications, but the ubiquity would mean everything. So if anyone wants to convince the appliance makers of the world of open hardware, they have my support. And the boiler repairman's too – his van is only so big...


Possibly the world's dirtiest tea cup.
Bike helmet, because I find cycling in the morning makes it much easier to work at a desk all day.
Microsoft may make terrible operating systems and awful web browsers, but its keyboards are surprisingly good.
A vertical mouse, because having arthritis sucks.

My Linux Setup Ben Everard

Linux Voice's technical editor shares his open plan blandness.

Q What version of Linux are you currently using?

A Across my various devices, I use Arch, Ubuntu, ChromeOS, Cyanogenmod and Android. All of them do a particular job really well.

Q And what desktop do you currently use?

A On Arch, it's a bizarre blend of Mate and Gnome 3, while on Ubuntu it's an early build of LXQt that I've never quite gotten around to upgrading. Nothing's ever quite normal on my machines.

Q What was the first Linux setup you ever used?

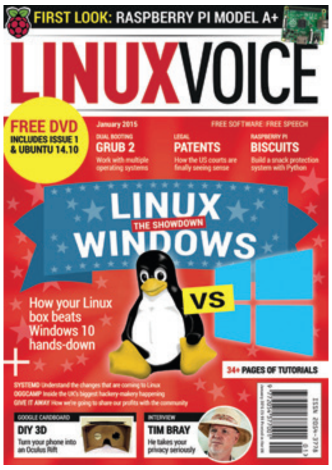
A I think it was Slackware, but I only used it once and couldn't get graphics working. I tried again with SUSE a year or so later, and that's when I started using Linux properly.

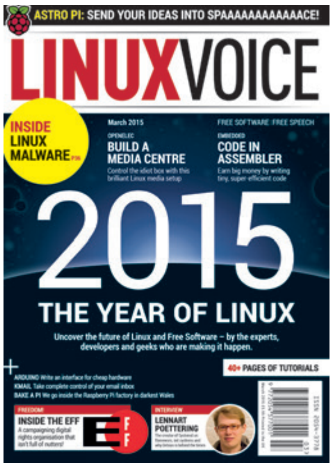
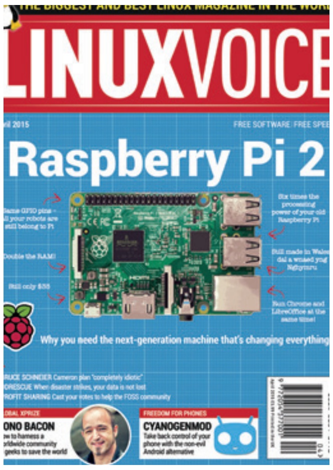
Q What Free Software/open source can't you live without?

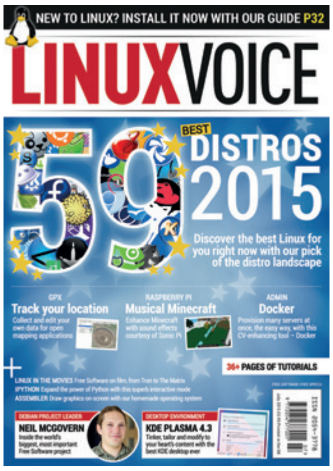
A *VirtualBox*. It is absolutely amazing for testing things out, and I seem to spend half my life in a virtual machine.

Q What do other people love but you can't get on with?

A C++. I program in a handful of languages, including some object-orientated ones, but for some reason I just don't like C++. I'm fine with C though.





O'REILLY®

OSCON

26–28 OCTOBER, 2015
AMSTERDAM,
THE NETHERLANDS

"I'm a believer! The breadth and depth of technical know-how is incredible. This conference pays for itself."

— TRACY LEFFLER,
OSCON 2014 ATTENDEE

Save 20%
on your ticket
(use code **AFF20**)

The power of open source

OSCON in Amsterdam celebrates, defines, and demonstrates the best that open source has to offer. From small businesses to the enterprise, open source is the first choice for engineers around the world.

osconeurope.com | @oscon | #oscon