NEW TO LINUX? INSTALL IT NOW WITH OUR GUIDE P32

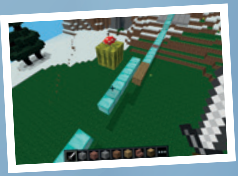
LINUXVOICE

BEST DISTROS 2015

Discover the best Linux for you right now with our pick of the distro landscape

GPX
Track your location
Collect and edit your own data for open mapping applications

RASPBERRY PI
Musical Minecraft
Enhance Minecraft with sound effects courtesy of Sonic Pi



ADMIN
Docker
Provision many servers at once, the easy way, with this CV-enhancing tool – Docker

36+ PAGES OF TUTORIALS

+ **LINUX IN THE MOVIES** Free Software on film, from Tron to The Matrix
IPYTHON Expand the power of Python with this superb interactive mode
ASSEMBLER Draw graphics on-screen with our homemade operating system

DEBIAN PROJECT LEADER
NEIL MCGOVERN
Inside the world's biggest, most important Free Software project

DESKTOP ENVRONMENT
KDE PLASMA 4.3
Tinker, tailor and modify to your heart's content with the best KDE desktop ever

FREE SOFTWARE | FREE SPEECH
ISSN 2054-3778
July 2015 £5.99 Printed in the UK
9 772054 377001 07

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

Distros, distros, distros!

The July 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.

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.

Thankfully, Linux is now very easy to install and use (if you want it to be). I've not counted how many clicks are required by the Ubuntu installer, but it's probably less than 10. That makes it considerably easier than installing one or two commercial operating systems I could mention. But one thing that doesn't necessarily get easier is finding the confidence to take that first step, whether it was Debian 1.1 in 1996 or Debian 8 in 2015. There's something empowering about installing your choice of operating system. But the knowledge that you're moving into new and unknown territory can also be a little scary.

We feel the solution to this problem is simple. It's support; to know that you're not alone. And it's this solution we're trying to nurture at Linux Voice, and why we try to cover as wide a range of subjects as possible, for all kinds of readers – from beginners (see p32), tinkerers (p84) and coders (p106), to advanced users (p68). Linux and open source is an adventure, and the more people who come along for the ride, the better that ride will be for all of us.

Graham Morrison
Editor, Linux Voice

SUBSCRIBE ON PAGE 66

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, Andrew Conway, Marco Fioretti, Josette Garcia, Juliet Kemp, Vincent Mealing, Simon Phipps, Gregory Pittman, Les Pounder, Mayank Sharma, Valentine Sinitsyn.

What's hot in LV#016

ANDREW GREGORY

"Until reading Ben's tutorial on Docker, I'd wrongly assumed it was yet another catchy buzword. But it seems rather ace." **p96**

BEN EVERARD

"Juliet Kemp's series on old programming languages is brilliant, especially as I've got a soft spot for Lisp." **p92**

MIKE SAUNDERS

"Our interview with the new Debian Project Leader is a great reminder of just how awesome the Debian project really is." **p42**

CONTENTS

July LV016

Rejoice, for Debian Jessie is here and the green shoots are showing.

SUBSCRIBE ON PAGE 66

TUTORIALS

REVIEWS

NEWSANALYSIS

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

Opinion

Free to scam

With great freedom comes great opportunity to swindle unsuspecting computer users…

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

Open source software is made available without any restrictions to everyone. Permission is granted in advance to do whatever you want with it. That's great! But that freedom also offers a resource for scammers to deceive and defraud the unwary. Freely licensed copyrights are freely licensed to the bad guys as well. There are plenty of them exploiting this gift to society for harm instead of for good as it's intended.

OpenOffice scams have been a sad fact of life for many years. Since users of office productivity suites are conditioned to expect substantial payments and regular updates, the scammers create professional-looking websites that offer a subscription to *OpenOffice*. They usually use the trademarked name and graphics in a way that would make anyone think they were the owner or licensee of the product. The most frequent pattern then offers "free" downloads, but require downloaders to pay a fee for an account to the download service. That fee is often modest, but hidden in the terms of use is an agreement that the fee is actually a recurring charge. Users who pay for the fee via credit card discover that they are charged the same amount of money monthly. The scammers often direct enquiries to the *OpenOffice* community public mailing lists, where confused users show up assertively demanding refunds believing they are addressing a private customer service address.

The same pattern also holds for Mozilla's *Firefox*. Mozilla's legal representative Anthonia Ghalamkarizadeh of law firm Hogan Lovells told me "Mozilla is regularly receiving reports through its online abuse reporting system from users. Most users won't be able to tell, or won't notice, that the download is not for the original version but either for one with modified default settings (such as different home page and different inbuilt search engine features) or comes with bundled third-party software."

VLC scams go even further. Using publicly available source code, the scammers build binaries that insert adware (or even malware) into Windows systems. They then buy Adwords credit from Google, placing advertisements for "VLC" and "Videolan" as well as the names of various hard-to-view video formats, so that people searching for a download of a media player or codec are deceived into downloading the trojan-infected rebuild of *VLC*. They then profit from the malware – injecting adverts, stealing credit card details and personal information and even participating in paid botnets – and use part of the profits to buy more adverts.

What can we do about this? Mozilla spends big money trying to protect us all, using trademark law as the lever. That led to a falling-out with the Debian project, which decided to make a running fork of *Firefox* rebranded as *IceWeasel* in protest at Mozilla's strong trademark policy. But it's paid dividends. Mozilla's (excellent) legal advocate has repeatedly shut down *Firefox* scams, in one case partnering with the German public prosecutor to secure both a conviction and damages against a substantial scam that had been preying on the public for years.

"The Mozilla Foundation spends big money trying to protect us all, using trademark law as the lever."

We are not alone

Not all projects can afford to instruct dedicated legal counsel to defend their good name and user community. The US-based Software Freedom Law Centre (SFLC) will usually help with the early stages of trademark enforcement for open source projects, but they do need to have clear trademark use and enforcement policies. But as open source grows ever stronger, we can expect the problem to grow, so now is the time to make sure your trademark policy is concrete if you are involved in a project.

Google also just accidentally helped matters get better. At the end of April, Google quietly changed its rules for Adwords advertisers who link to software. In the process, it has helped reduce the fraudulent abuse of open source software. The new rules disallow "Promotion of free desktop software, unless the ad includes the name of the specific software being promoted and leads to the authoritative online distribution source for the software. The authoritative source must not have a history or reputation of policy violations."

Maybe these changes by Google, intended to protect it from ad-injection malware, will make a difference. But the scammers will go where the money is to be made, and as open source grows so will their target. Take care out there, and help educate your friends and family to always download open source direct from the project and never from any other site.

Releases • Endless • Cyanogen • Mint • Xubuntu • Qt • Microsoft

CATCHUP

Summarised: the biggest news stories from the last month

1 New releases galore!

It's been a busy month for FOSS projects. Debian 8 was released after two years of development (and plenty of flamewars over *Systemd*). See page 52 for our review. Around the same time, Ubuntu 15.04 arrived (more on that in our cover feature), along with its variants in the form of Kubuntu, Xubuntu and Lubuntu. On the desktop front, KDE Plasma 5.3 was released, and we have the full lowdown on that on page 54. And by the time you read this, Fedora 22 will be close – turn over the page to find out what's new.

2 78% of businesses use open source software

That's what a survey by Black Duck Software, an open source consulting firm, says. However accurate it is, there's no denying that FOSS is playing an increasingly vital role in many businesses. In fact, we find it hard to imagine any business that doesn't use FOSS such as OpenSSH, *Apache*, PHP, Perl, *MySQL* and other well-known tools. Or perhaps those 22% of businesses that claim they don't use FOSS simply don't realise that it's inside many apps and products that they depend on...

3 Endless: a Linux computer for the whole world

This kickstarter aims to bring Linux to the 5 billion people who don't have a computer. It's a cheap and cheerful box that plugs into a TV and uses a mobile phone-type low-power CPU. **http://tinyurl.com/q76r5ng**

4 Cyanogen partners up with... Microsoft

This news had the internet exploding with rage, but it was largely misinterpreted. Cyanogen Inc., which makes mobile OSes based on Android, has announced that it will bundle Microsoft apps and services with its Cyanogen OS platform. But note: this is not the same as CyanogenMod, the Android variant that many of us run on our smartphones and tablets. The latter will remain "neutral on services", while Cyanogen OS will chase the big bucks. So breathe a sigh of relief...

5 Linux Mint gets cash boost from VPN service

Ubuntu spin-off Linux Mint has announced a new primary sponsor, PrivateInternetAccess. This company is one of the biggest VPN providers, and claims to offer extremely high levels of security, never logging anything that its users do. However, it's based in the USA and could be served with secret court orders to hand over any data it has, without telling the public. In any case, this is great news for the popular Mint distro, and should ensure a healthy future for a few years to come at least.

6 Xubuntu announces Core, unrelated to Ubuntu Core

Xubuntu is already one of the lighter mainstream distros, but the newly announced Core version will fit onto a CD and offer just the basics. Users can then download what they want, without having to remove bundled apps. The choice of "Core" as the name is crazy though – even the Xfce team admits that it could get confused with Ubuntu Core, a very different project! "Xubuntu Mini" would be better, we reckon. **http://xubuntu.org/news/introducing-xubuntu-core**

7 Qt slips up with installer account requirement

Qt, the framework used by KDE and other FOSS projects, has always occupied a difficult spot between the free and commercial software worlds. For the latest release, the Qt Company changed the installer so that it asked users to log in with a Qt account before proceeding. This smacked of "registration" or "activation" for many users, who complained bitterly. In the end, the Qt Company wisely backed down and removed the login requirement.

8 Microsoft in actually-liking-Linux shocker

Flying pigs have had to wrap up well during the cold spell in hell recently. Not only did Microsoft hold a launch party for Debian 8 at a Linux event in the US, the notoriously non-FOSS-friendly company has also released a code editor for Linux (see our review on page 53). Maybe this is down to the new CEO, who has a markedly different attitude to "Linux is cancer" Steve Ballmer, or perhaps it's just acceptance that yes, Linux is here to stay, so let's all get along nicely.

DISTROHOPPER

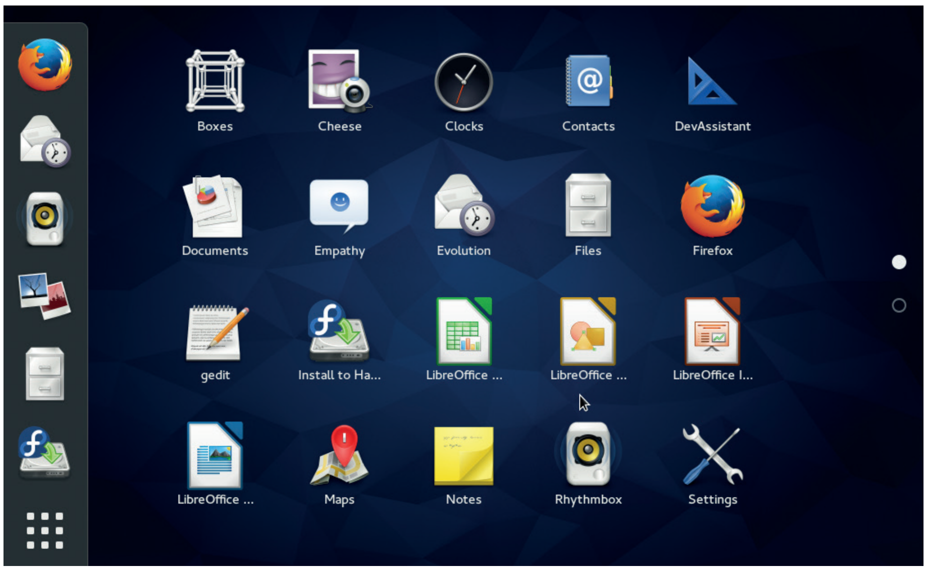
What's hot and happening in the world of Linux distros (and BSD!).

Fedora 22

Bleeding-edge Linux.

Fedora 22's final release date has slipped by a week, but it should almost be ready by the time you read this; it's due on 26 May. We've come to expect plenty of cutting-edge goodies in Fedora releases, but 22 goes a step further with major updates all over the distro. Wayland takes a prominent role, being the default display subsystem for the login screen, although X.org is still used for regular desktop sessions. It feels like we've been waiting for Wayland for aeons, so this is a small but significant step forward, and will provide the Fedora and Wayland teams with plenty of real-world feedback.

Then there's a big change to *Yum*, the package manager. This has been replaced by *DNF*, the "Dandified Yum", which offers better performance for such things as dependency resolution, but also maintains most of the same command line options. Then there's the new Gnome release with its improved notifications, plus enhancements to the bug reporting tool and other apps.



Fedora 22 showcases the latest Gnome, along with Wayland and the new *DNF* package manager.

On the server, there's a new Database role (utilising *PostgreSQL*) and improvements to the *Cockpit* web-based management tool. Fedora 22 will be available in three flavours – Workstation, Server and Beta – the first of which is provided as a 1.3GB DVD ISO download. All flavours are built from the same core, using the latest versions of the kernel, *Glibc* and *Systemd*, so it's pretty much the most bleeding-edge distro you can get.

If you try Fedora 22 and have any issues at the login screen, it's well worth letting the developers know (**http://bugzilla.redhat.com**). The more users that test it on a wide range of hardware, the closer we'll get to Wayland replacing X.org on the desktop.

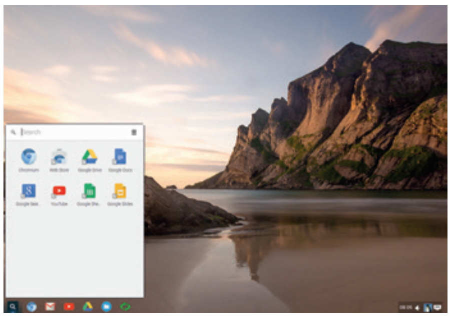
Chromixium 1.0

A web-centric OS based on Ubuntu's LTS releases.

Google's Chromebook laptops have taken off in a big way. They do almost everything that a typical non-technical user needs, and they have some benefits from being locked down and limited in scope, in that users can't double-click random **.exe** files in emails and clog the machine up with spyware.

Now, as Linux geeks we have our issues with such restrictive devices, and while Google has generally been a good ally of the free software community, its vast data collection mechanisms cause us a bit of concern. Chromixium looks like a healthy middle ground: it "combines the elegant simplicity of the Chromebook with the flexibility and stability of Ubuntu's Long Term Support release". By and large, it looks similar to Chrome OS, with a very minimalistic panel sitting on the bottom of the screen, providing access to various web apps (from Google, of course).

But it's still Ubuntu, so you can load it up with *LibreOffice*, *Skype*, *Steam* and other popular apps. Right now it's only available as a 32-bit ISO download, and the distro's website is rather lacking, but the team promises to do a lot more work on documentation. We're also unsure about the name: not only is it a mouthful, but it's close enough to Google's own wares (Chromium, Chrome OS, Chromebooks) that users could conceivably see it as an official project, and not something spun-off by the community.



Like the idea of Chrome OS, but want more Linux behind it? This could well float your boat.

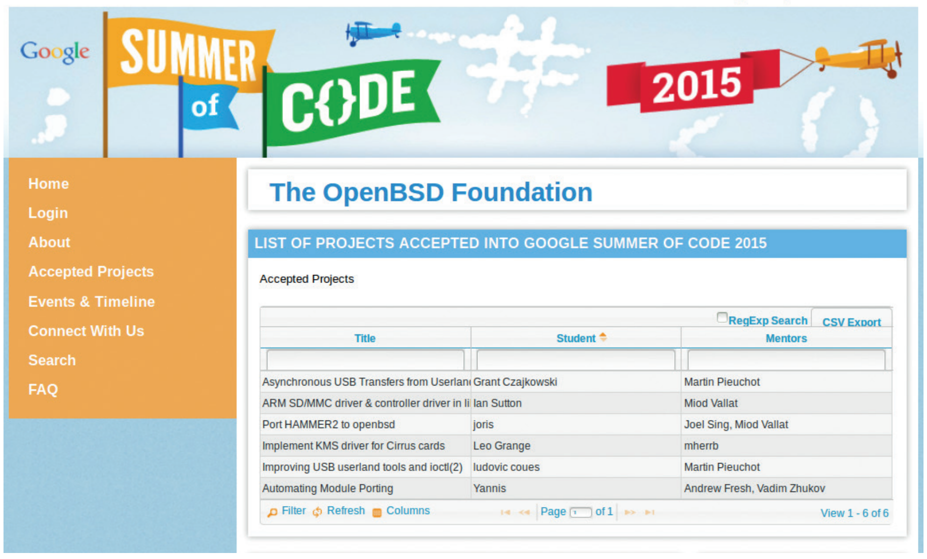
News from the *BSD camps

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

BHyVe, FreeBSD's native hypervisor for running virtual machines, is a relatively new project (having first appeared in FreeBSD 10.0) but it's making rapid progress. Until a few weeks ago, it was capable of running FreeBSD, OpenBSD and Linux as guests, and a recent update has added support for Windows as well. In the past, it was possible to run Windows on FreeBSD via *VirtualBox*, but the new *BHyVe* support should boost performance on the guest and means that extra software won't be required.

Also on the FreeBSD front, the project has issued its first quarterly status report for 2015, detailing work that's taking place all over the codebase. *LLVM/Clang*, the default compiler toolchain in recent FreeBSD releases (replacing *GCC*), has been updated to 3.6.0, while some developers have been beavering away on a new bootloader. There's also a push for ASLR – address space layout randomisation – which improves security by randomising the location of executables and libraries in RAM. With ASLR, crackers can't guarantee where certain bits of code are stored in memory, making it harder to perform exploits.

A number of OpenBSD projects have been accepted in the Google Summer of Code 2015. Projects include asynchronous USB transfers from userland, better support for SD card controllers on ARM devices, and a port of DragonFly BSD's Hammer 2 filesystem. Oh, and the EU published a study in favour of financially supporting open source projects – including OpenBSD – so maybe more funding will come from there.



The OpenBSD Foundation

LIST OF PROJECTS ACCEPTED INTO GOOGLE SUMMER OF CODE 2015

Accepted Projects

Title	Student	Mentors
Asynchronous USB Transfers from Userlan	Grant Czajkowski	Martin Pieuchot
ARM SD/MMC driver & controller driver in li	Ian Sutton	Miod Vallat
Port HAMMER2 to openbsd	joris	Joel Sing, Miod Vallat
Implement KMS driver for Cirrus cards	Leo Grange	mherrb
Improving USB userland tools and ioctl(2)	ludovic coues	Martin Pieuchot
Automating Module Porting	Yannis	Andrew Fresh, Vadim Zhukov

View 1 - 6 of 6

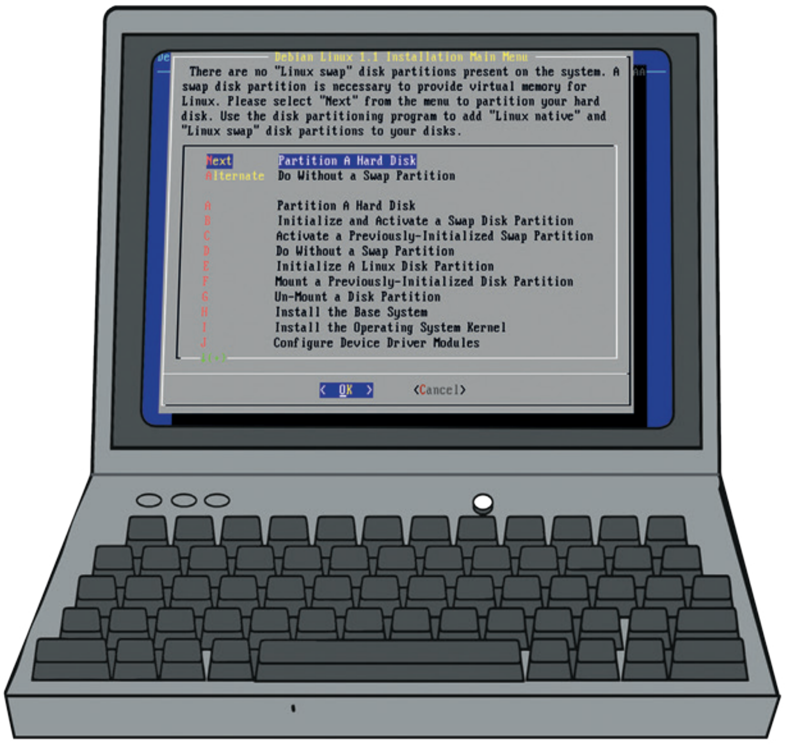
A few OpenBSD hackers will be financially rewarded for their work this summer thanks to Google.

Debian 1.1 – actually the first Debian release!

Yes, it sounds odd, but there was no such thing as Debian 1.0. At least, not officially. Back in 1995, the newly assembled Debian team was grafting away on the distro's first release, and then a CD vendor called InfoMagic took a development branch, accidentally labelled it "1.0", and started shipping it. This major slip-up could've ruined the fledgling project's reputation, but both Debian and InfoMagic quickly announced that the so-called 1.0 wasn't actually a proper release, and the first real version of Debian would now be called 1.1 to avoid any confusion.

Released on 17 June 1996, Debian 1.1 included Linux kernel 2.0 and a whopping 474 packages. (Contrast this with the newly released Debian 8, which has over 43,000 packages in its repositories – that's growth!) The distro also used the ELF binary format throughout, in comparison to **a.out** that some other distros were still struggling along with, so in that respect it was fresh and modern.

You might expect such an early Linux distro to lack a proper installer, providing instead just a list of steps to go through, but Debian's setup tool was impressive at the time. It's a colourful menu-driven program that's not a million miles away from what we have today, helping you to partition your hard drive and copy files over. If you want to give it a go, grab the floppy disk images from **http://archive.debian.org/debian/dists/buzz/main** – you may be able to get it up and running in *Qemu* or *VirtualBox*. Or if you have an ancient PC sitting around doing nothing useful, try that instead!


```
Debian Linux 1.1 Installation Main Menu
 There are no "Linux swap" disk partitions present on the system. A
swap disk partition is necessary to provide virtual memory for
Linux. Please select "Next" from the menu to partition your hard
disk. Use the disk partitioning program to add "Linux native" and
"Linux swap" disk partitions to your disks.

   Next        Partition A Hard Disk
   Alternate   Do Without a Swap Partition

   A           Partition A Hard Disk
   B           Initialize and Activate a Swap Disk Partition
   C           Activate a Previously-Initialized Swap Partition
   D           Do Without a Swap Partition
   E           Initialize A Linux Disk Partition
   F           Mount a Previously-Initialized Disk Partition
   G           Un-Mount a Disk Partition
   H           Install the Base System
   I           Install the Operating System Kernel
   J           Configure Device Driver Modules

              < OK >        <Cancel>
```

Cosmetically, the text-mode variant of Debian's installer has barely changed in the last 20 years.

GAMING ON LINUX

The tastiest brain candy to relax those tired neurons

THE SONS OF RAGNAR

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

At the time of writing, Steam has almost 1,200 Linux games in its store, and while this is far from a comprehensive figure of all the games on Linux, it's a pretty good indicator. To put this into perspective, the Xbox One has just under 200 games, while there are around 1,700 OS X games on Steam.

Linux gaming is becoming a big deal, and if Valve decides to market its Steam Machines as consoles, it will have the biggest launch lineup in history by a long, long way. If Valve's own figures are to be believed, there are currently between 1.25 and 1.5 million active Linux gamers on Steam, which is a pretty sizeable figure but again most likely falls short, especially considering users tend to steer clear of DRM.

All hail Linux!

Linux is becoming a force to be reckoned with and is an increasingly attractive market to developers now that porting and cross-platform development is becoming easier and cheaper. With this in mind, it seems likely that even if Steam Machines are a complete flop – though we may not continue getting some of the bigger titles which were ported based on its potential success – we'll still get a decent number of games. In the worst-case scenario, we'll have 1,200 games to play, plus the countless other great open source projects like *0 AD* and *OpenMW*, though it's more likely that little Tux icons will become a common sight at games vendors.

Pillars of Eternity

An epic CRPG that sucks you into its rich fantasy world.

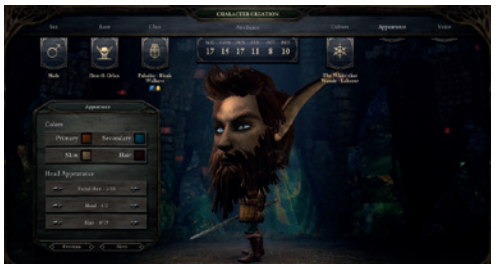
Despite all the vapourware that Kickstarter games give us, when something like *Pillars of Eternity* comes out it makes all that seem worthwhile. Obsidian Entertainment promised a classic CRPG experience, and that's exactly what they delivered.

Story-wise the game puts you in control of a character of your creation, and lets you choose their backstory, gender and race among other things, but the character is always a "watcher" – someone who can see past lives. The story unfolds in The Dyrwood, a land at the edge of civilisation and troubled by strange goings on which you have to get to the bottom of.

Pillars proves that the vapid eye candy of multi-million dollar graphics offered by the AAA game industry don't make a good game alone, and one that uses isometric graphics and has the player reading novels of text can provide a far more meaningful and captivating experience.

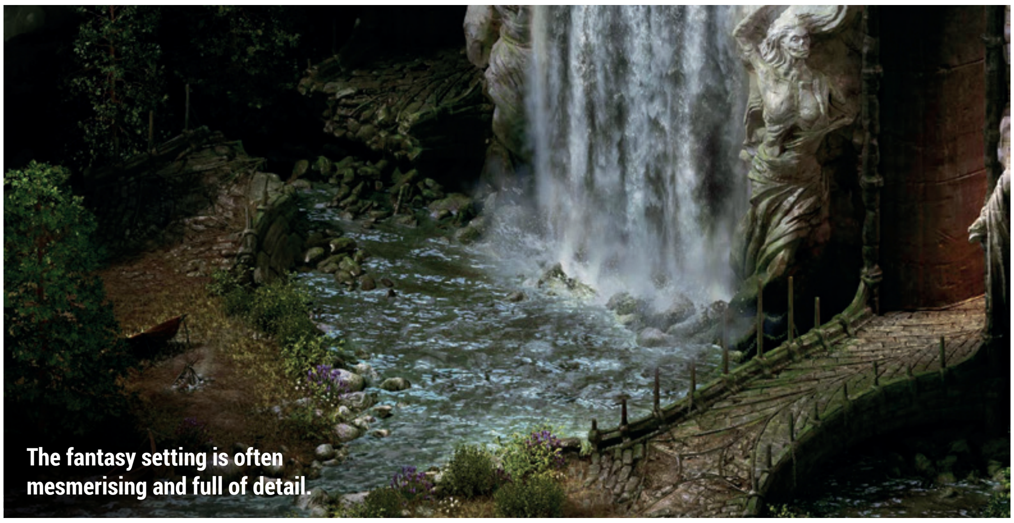
This doesn't mean that the game looks particularly dated, since some very pretty particle effects, solid voice acting in certain quests, a professional musical score and plenty of metagame extras remind us that this is a product of the 21st century rather than the 20th.

The stronghold sidequest, which allows the player to take over a castle, restore it, upgrade it and oversee its management is the cherry on the icing of this game, not to mention the vast multi-floor mega-dungeon underneath it, which again provide elements of digression which make the game more welcoming to newer players.



The character creation screen provides plenty of options – massive heads are ill advised.

Website http://store.steampowered.com/app/291650 **Price** £34.99



The fantasy setting is often mesmerising and full of detail.

"The story unfolds in The Dyrwood, a land at the edge of civilisation and troubled by strange goings on."

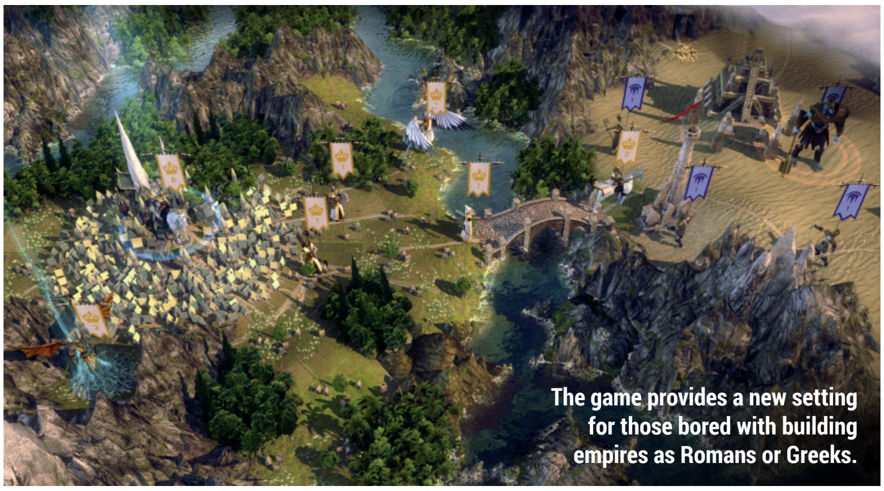
Age of Wonders III

A solid turn-based strategy game in a fantasy setting.

Age of Wonders III provides a welcome addition to the 4X genre (that's eXplore, eXpand, eXploit, eXterminate). In particular, it excels in "selling" the world to its players by creating a backstory and lore for the different characters and races; without this, a 4X game in an alternate world can often feel like micromanaging numbers (as it did with *Beyond Earth*), rather than a fun strategy game.

Players familiar to the genre will get to grips with it very quickly; others may find it a bit intimidating. However, this can be overlooked when taking into account the things the game does very well, such as heroes leading troops into battle or a vast underworld as big as the above-ground world. The battles are epic affairs with detailed character models and vast armies, which introduce an element of skill rather than the units with the most hit points being assured victory.

Website http://store.steampowered.com/app/226840 **Price** £29.99


The game provides a new setting for those bored with building empires as Romans or Greeks.

Out Of The Park Baseball 16

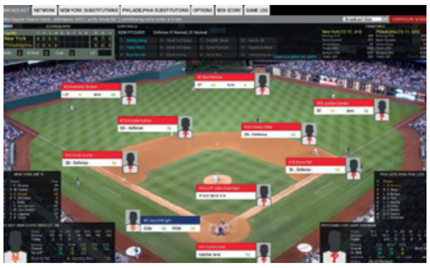
An addictive addition to the Linux management sim roster.

OTP 16 is the third iteration of the game we've seen on Linux, but unfortunately the franchise has gone largely unnoticed.

So should this game be discarded for those who know nothing about baseball? The short answer is no. The long answer is no, and you didn't need to be a GP to enjoy *Theme Hospital* or a theme park aficionado to enjoy *Roller Coaster Tycoon*.

The game is not only impressive for containing basically every baseball league on earth, but also for being able to play historical games going all the way back to 1871, making it in some ways a historical strategy game which lets the player change the course of history as well as a sports management sim.

It really is incredibly satisfying to see your little team climb the league as a result of your managerial skills, while on top of that learning a lot about baseball. Soon enough it turns into one of those games where every time you look at the clock a couple of hours have passed. Be warned if you still intend on being a productive and sociable person.


OOTP 16's level of detail won't disappoint those who obsessively go over statistics.

Website http://store.steampowered.com/app/333820 **Price** £29.99

ALSO RELEASED...

SuperTuxKart

Our favourite open source kart racing game has reached version 0.9, and has implemented a new rendering engine which makes the graphics look stunning and on par with many commercial games. This release candidate version may pave the way for releases on platforms like Steam, as developers have hinted, which would be a big breakthrough for open source gaming.
http://supertuxkart.sourceforge.net

StarDrive 2

In this strategy game you lead a spacefaring civilisation and form a galactic empire, taking part in epic space battles against pirates and rival civilisations with highly customisable ships along the way. The usual 4X tech trees, building mechanics and exploration are very well executed, not to mention an excellent tutorial which helps the player get to grips with things right from the start. Very nice musical score too!
http://store.steampowered.com/app/252450/

The Banner Saga

Trudging through the cold northern lands with a Viking caravan, in a world where the sun has stopped and falling apart at the hands of a terrifying foe, this game puts you in the thick of it right from the start. The story is well put together, the animation is spectacular and the tactical combat is extremely satisfying. *The Banner Saga* is a great RPG experience which also suits casual gamers.
http://store.steampowered.com/app/237990

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

LYX

Great to see the article on *LyX*; just one quibble though. Nobody seriously uses *Aspell* with *LyX*. The default is *Hunspell*, which uses the *LibreOffice* dictionaries.

One *LaTeX* feature not mentioned (I know you cannot mention everything in a short article) is *BibTeX*, the bibliographic database system which is second to none for anyone in academia.

There are some PDFs at **http://johnrhudson.me.uk/computing.html**, which may be useful for people wanting to go further with *LyX*. The *LyX* wiki **http://wiki.lyx.org** is also a great source of information, particularly on more specialist features.

John R Hudson

Andrew says: Much as my heart trembles at the thought of using a spellchecker originally written for the Hungarian language, it looks like this is indeed the default spellchecker for *Lyx*, to the extent that it's not compiled into the *Lyx* binaries for Windows. Not that we need spellcheckers anyway… Now *BibTeX*, that's a golden discovery. References are an absolute pain to get right, so anything that helps there is worth its weight in electrons.

BIBTEX

LyX and *LaTeX* give you much more control over your documents than a typical word processor.

SECOND HAND GAMES

I like to buy second hand PC games on disc, usually four or five years old and mystery or puzzle games. On the box it says they can be used with XP, Vista and Windows 7.

My questions are: will these games run if I use Linux as my OS; and if not, is it possible to buy mystery/puzzle games, on disc, that will work with Linux.

I saw in your most recent edition that more games are now available to use with Linux, including a city builder game which I would love. Do you have to download these games from websites or can you buy them on disc?

Sorry if my questions seem obvious, but I've only owned a computer for five years and I've got alot of catching up to do.

Mark Alexander

Graham says: PC games on disc are getting rarer and rarer. Most games are delivered as digital downloads through a website or application. Steam is the most popular, with over 1,100 Linux games now listed. **GOG.com** is another but with an emphasis on old games. This sounds much more your kind of thing (games are typically cheap too). You can run some old XP, Vista and Win 7 games discs on Linux, but it's a case-per-case situation and you'll need to run them through something called *Wine*. This is actually a really good idea for a tutorial, so keep an eye out!

Emulators (such as *Nestopia*) can bring old games to life on Linux.

THE LAUGHING GNOME

I seem to remember, years ago, a certain magazine ran with a cover asking “Has Ubuntu Lost it?”. This was in response to the dummy-spitting that went on when Ubuntu changed from Gnome to the Unity desktop. If the geeks don’t like it (and a lot of geeks didn’t like it), then it must be rubbish, right?

Well, no, not really. Old-time Unixers getting upset about change are the equivalent of comic geeks getting upset because the costume of their favourite character is a different shade of red than it was in the comics. It’s not about them; it’s about the kids who’ve come to see the Hulk and Thor and the boring one with the bow and arrow.

Anyway, I think we can all see with the benefit of hindsight that Unity was not meant to satisfy existing Linux users, but to bring Ubuntu and Linux to a wider audience, and if you think about it that way it’s been a massive success. It’s just a shame that more people can’t acknowledge this and give it the credit it deserves.

David Ross, Dundee

Andrew says: To be fair to the excellent editorial team who asked the question ‘Has Ubuntu Lost It?’, the answer to that rhetorical question was a firm ‘no’. Any change is always going to ruffle feathers (see below), but if you ask the people what they want, all they’ll say is ‘faster horses’. It’s to Canonical’s credit that it’s stuck to its guns and produced something so touchably lovely that, even without Android’s massive app ecosystem, the Ubuntu phones are destined to do well.

Traditional WIMP interface? No. Perfect for mobile, touchscreen computing? Yes.

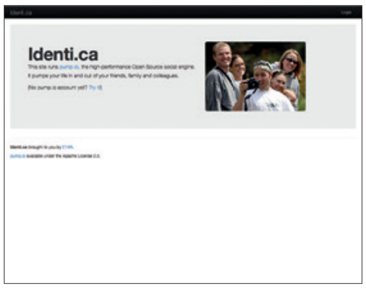
IDENTICA

Guys, why no Identi.ca presence? You’re on Twitter and Google+ – why do you support those big old corporations and not the free software equivalent. You’re supposed to be about Free Software and free speech, so do something about it!

Richard Hirst

Andrew says: You’re right. *Mea culpa*. If it helps us reach more people and supports free software, we should be doing it.

When Twitter goes the way of Facebook and starts charging you to send out messages to all your followers, when it’s invading your privacy, selling your details to advertisers, and constantly bugging you to get in touch with people you don’t like or have never met, we’ll all wish we did more to support Identica.

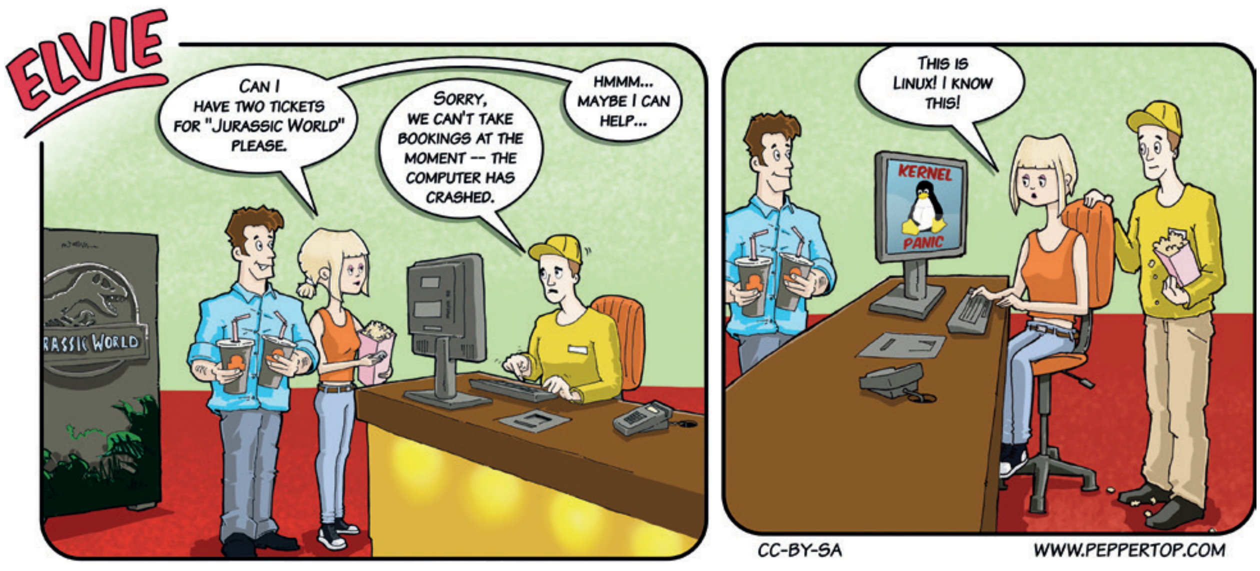

Identi.ca

This site runs pump.io, the high-performance Open Source social engine. It pumps your life in and out of your friends, family and colleagues.

(No pump.io account yet? Try it!)

Identi.ca brought to you by E14N.
pump.io available under the Apache License 2.0.

Perhaps one reason we haven’t joined Identica is that we’re put off by its dire website.


ELVIE
CAN I HAVE TWO TICKETS FOR "JURASSIC WORLD" PLEASE.
SORRY, WE CAN'T TAKE BOOKINGS AT THE MOMENT -- THE COMPUTER HAS CRASHED.
HMMM... MAYBE I CAN HELP...
JURASSIC WORLD
THIS IS LINUX! I KNOW THIS!
KERNEL PANIC
CC-BY-SA
WWW.PEPPERTOP.COM

WINNING

Just another conversion story here folks. My wife bought a laptop by accident; she needs a machine to do her marking on (she's a teacher), but it can't be 'mobile'. Never mind that a desktop tower is just as mobile as a laptop to anyone with the strength to pick one up, but the education authority has its rules that must be obeyed. Anyway, she needs Microsoft Windows to access the marking software, but as the laptop became surplus to requirements I quickly loaded it with Linux Mint Mate, installed all the codecs at boot time, and let her have it. No support, no guidance, no sitting down with her to show her what all the buttons do – she just gets it.

This is how things should be. It's not going to get slowed down by viruses, or weighed down by an increasingly bloated operating system, so I'm expecting the new machine to last a lot longer than its predecessor. Never mind next year being the year of Linux on the desktop – in our house, it's this year.
Brian Holgate

Graham says: Mint really is fantastic for a beginner-level Linux distribution, even though in our feature on page 32 Mike backs Ubuntu as the best for newbies. However, this is partly because Ubuntu has the biggest support community; if your wife has in-house support (ie you), she's winning already. Well done at keeping another precious computer away from Microsoft's clutches!

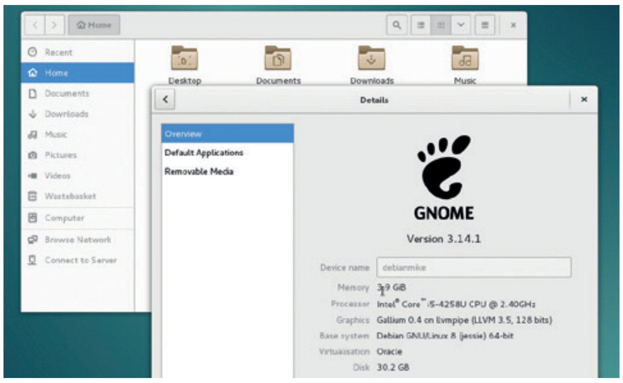
If you need to breathe new life into old hardware, there's a flavour of Linux out there for you.

SYSTEMD: WHAT'S IT ALL ABOUT?

What's all the fuss about Systemd? I've been listening to the Linux Voice podcast for years now (and the Tuxradar podcast that you chaps used to do), and you all seem to make a teeth-sucking noise whenever Systemd comes up, like a mechanic about to overcharge me.

Surely it's not the end of the world if your init system changes? Surely nobody even needs to know what in init system is if they're just a common or garden desktop user like most of us are? Do Windows users need to know what startup scripts are running on their machines? Of course not. This is quite apart from the abuse that the developers have been getting, which is another issue entirely. As civilised adults, we should be able to separate the technical discussion from personal attacks. Unfortunately many people are not civilised adults.
Robert O'Sullivan, Cork

Andrew says: I too find it baffling. But I think a lot of it comes back to David Ross's point, that any change is going to ruffle feathers. Imagine you've spent years learning how a system works, its every intricacy, and then someone comes along with an innovation that improves on that system, but makes all your years of learning worthless. That's the only way I can make sense of all the rage around a bit of plumbing on my Linux machine. The irony is that it's so contrary to the spirit of Free Software, where everyone contributes and the best code wins; clinging to the past is something you'd expect a near monopoly to do in order to stifle more advanced competition…



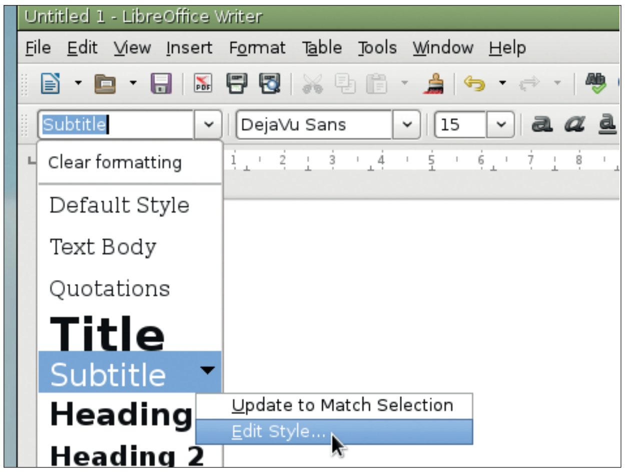
Version 3.14.1
Device name debianmike
Memory 3.9 GiB
Processor Intel® Core™ i5-4258U CPU @ 2.40GHz
Graphics Gallium 0.4 on llvmpipe (LLVM 3.5, 128 bits)
Base system Debian GNU/Linux 8 (jessie) 64-bit
Virtualisation Oracle
Disk 30.2 GB

The latest version of Debian has made the switch to Systemd, in the teeth of fierce opposition from a group of developers.

LIBREOFFICE

Can we have a beginner's tutorial on doing stuff with *LibreOffice* please? It's a huge piece of software, and I get the feeling that I only ever use a tiny slice of what it can do. There must be so many ways of saving time and effort that I don't know about – share the secrets with us!
Sarah McKie

Andrew says: There must be. I understand the importance of *LibreOffice* to a lot of people, but I've never really had to use it that much. In the days when it was bloated old *OpenOffice* I always used to use the *AbiWord* word processor, as it was much, much faster. Nowadays I use *InDesign* when I want words to look nice, and a text editor when I want to write something quickly – for my needs, *LibreOffice* falls in between the two use cases. That said, the development team is adding features all the time, so we should take a look at some of those soon. I don't think there's too much call for a beginner's tutorial though – does anyone out there disagree?



It's one of the flagship free software applications – *LibreOffice* is proof that you don't need to pay to get excellent software.

YOUR AD HERE

LV

Email andrew@linuxvoice.com to advertise here

LUGS ON TOUR

Pycon Sei

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

A few weeks ago, I received the most wonderful email – an invitation to attend Pycon Sei in Florence. Who could refuse such an invitation – not me for sure!

I could spend weeks there enjoying the views, the food and going mad at the number of tourists! Unfortunately Florence is loved by many people and sometime you feel that you are not going where you want to but you are carried somewhere.

In Italy the Pycon meetings started in 2007 in the centre of Florence and continued until Pycon Quattro – I fell in love with Florence during Pycon Due when Richard Stallman gave a talk at the Palazzo Vecchio. In 2011, Florence hosted EuroPython – a partnership that lasted three years (pretty good when EuroPython had to move on to another city after three years). Unfortunately the hotel in the centre could not cope with 1,000 delegates and the conference was moved to the Grand Hotel Mediterraneo, a few yards from the Arno and almost opposite the Piazzale Michelangelo. 2014 saw the return of Pycon with Pycon Cinque. During that year, the Associazione Python Italia with Tinker Garage APS also organised Django Village.

For 2015, they decided to combine the two conferences and produced Pycon Sei with a minimum of four tracks: Python, Django, PyData and Odoo.

Lots of the talks were in English. My favourite speaker, Alex Martelli, started the day with Modern Python patterns and idioms first in Italian and later on in English – I do not understand Italian nor Python but to see Alex's passion is pure magic – all of his body is moving. He is the epitome of Italians, the way we like them and sometime make gentle fun of them.

Over 1,000 delegates attended Pycon Sei – so many that the venue needed to be changed.

Other good presentations included Asynchronous Web Development with Python 3 by Anton Caceres; *PostgreSQL 9.4* for Devops by Gabriele Bartolini; Odoo disaster recovery con Barman by Giulio Calacoci; Does Python stand a chance in today's world of data science by Radim Rehurek; Packaging Django projects for PyPI by Roberto Rosario, and more.

"There are no techie conferences without networking, and the first event was PyBeer on Friday."

Saturday saw a recruiting session – it was fascinating to see the different ways that companies try to entice new recruits. It went from the big PR spiel to the down-to-earth approach: that's the job, that's what we want from you, that's what you get from us. The companies hiring included InfoCert, Zalando, Kuldat, Develer and 2ndQuadrant.

There are no techie conferences without networking, and the first event was PyBeer on Friday, which took place at the James Joyce pub. To consolidate our new friendships, on Saturday we met at the Ristorante Zazà for PyFiorentina. There you can taste the famous bistecca alla Fiorentina – a T-bone steak grilled over a wood or charcoal fire, to be eaten with a glass of wonderful Italian red wine. Just delicious!

Pycon Italy is organised by the Python Association of Italy, and Develer SRL (**www.develer.com**) spends a lot of time and resources putting the show together.

Despite our not understanding his Italian, Alex Martelli's talk on modern Python patterns looked fantastic!

Libre Graphics Meeting 2015

Greg Pittman reports on this year's finest open source graphics event

LGM 2015 has just taken place in surprisingly sunny Toronto, Canada, from 29 April to 2 May, at the University of Toronto. It was a smaller crowd than at some previous LGMs, yet a dedicated one, and we saw an impressive array of presentations. Not only did we find, as expected, that our major projects such as *Gimp*, *Inkscape*, *Scribus* and *Blender* are very much alive and still developing after this decade of LGMs, but we continue to see new ideas coming to the open source graphics community. Lively discussions occurred following presentations, and in various formal and informal meetings.

Early on we received some encouragement and prodding for the vector graphics artists out there to submit work to Wikimedia Commons in order to improve the quality of Wikipedia articles. We also learned about an Android app (*The List*) in development so that users can learn about image requests from Creative Commons, take photos with their phones, then automatically submit them. From the W3C we learned about developing guidelines for accessibility features for the web, even some ideas about considering how to make graphics accessible to the visually impaired. Also from W3C we learned that an effort has begun to create guidelines to encourage the ability to read ePubs and other ebooks within browsers.

Work on font creation, manipulation, and usage has become a major part of Libre Graphics Meetings in the past several years. Although work continues on FontForge, others are working in other directions, such as the ability to edit entire character sets at once by manipulating various font metrics (metapolator), and OSP continues their pioneering work on font creation, usage, and publishing. A concerted effort has begun to start the groundwork for making use of the advanced features of OTF fonts.

***Blender*, *Gimp*, *Inkscape* and other teams get together once a year to synergise their awesome visions!**

Animation and video editing

Blender continues to rise in its stature in the world of not only open, but also commercial animation. A new open source video editor, *Natron*, is working on closer connections with *Blender* for various video compositing tasks. *Pitivi* seems to have gotten some renewed energy in recent years, and is now approaching version 1.0. We also heard from a university educator using open source software, finding he not only empowers his students, but they seem more creative, and are encouraged to interact with the projects. A group (Theory Animation) has managed to create a virtual animation studio by regularly interacting online from numerous sites on three continents, using *Blender* to create animations.

"We continued to see new ideas coming to the open source graphics community."

We learned about glitch art, various ways to "mess up" image files for interesting effects – things like loading a JPEG into *Audacity*, editing there, and seeing what happened.

From some ideas coming out of steganography we get to new ways to compress 3D images into 2D image formats, which leads to ideas on image manipulation. And LGM wouldn't be LGM without some selected sessions on small bits of code having powerful effects, like graphically manipulating the code itself (Microraptor GUI). *GEGL* on a server can be used to automatically modify images on a web page (*imgflo*). *LibreOffice* is undergoing some major changes in its rendering scheme, promised to bring improvements in results as well as cross-platform economies of coding. The Documentation Liberation Project continues its vital work on unlocking proprietary and aging file formats, and managing colour in all of our various devices continues to become easier with the availability of open source software and hardware.

LGM 2016 will take place in London, at the University of Westminster (Harrow). Anyone interested in open source graphics is encouraged to attend. **www.libregraphicsmeeting.org/2015**

59 BEST DISTROS 2015

The Linux Voice team selects the cream of the Linux crop for 2015 – there's something here for everybody.

We don't need much of an excuse to start playing with lots of different Linux distributions. It's one of the best ways of spending a wet afternoon, especially if you've been using the same clutch of distributions for a while. Trying something different opens up all kinds of new perspectives on the software we all use and love, and that's why we've put so many together here.

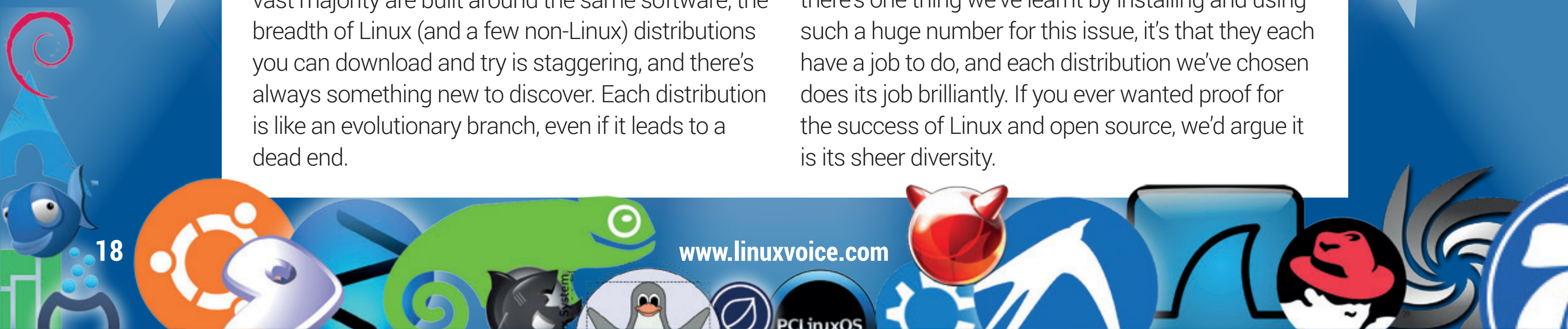
We've forgone any ranking of the distributions, instead focusing on a selection we find fascinating and worthy of your consideration, from distributions that take up less space than an email attachment to one that's helping decode the secrets of the universe.

Considering the vast majority are built around the same software, the breadth of Linux (and a few non-Linux) distributions you can download and try is staggering, and there's always something new to discover. Each distribution is like an evolutionary branch, even if it leads to a dead end.

"We've focused on a selection we find fascinating and worthy of your consideration."

Installing and playing with all these distributions has also reminded us about just how much has changed. Many distributions are now offering rolling releases, for instance, where you don't have to go through a monumental system upgrade just to get the latest version of *Firefox*. And there are now more desktop environments than ever before, with some distributions remaining the ideal choice for users of one desktop or another.

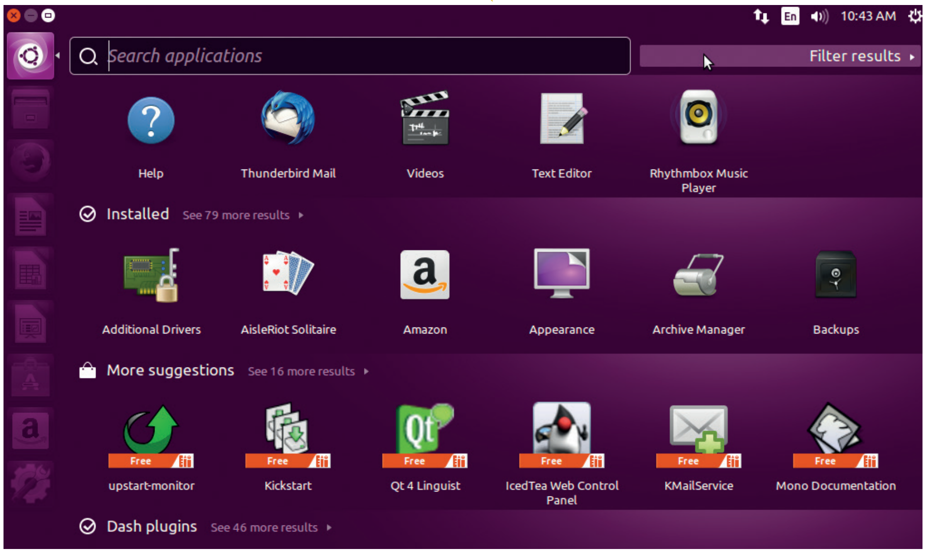
We often joke about the vast number of distributions there are. When you look at the huge list curated by **DistroWatch.com**, you can't help but wonder why so many have been created, especially when the effort involved in their creation is so huge. But if there's one thing we've learnt by installing and using such a huge number for this issue, it's that they each have a job to do, and each distribution we've chosen does its job brilliantly. If you ever wanted proof for the success of Linux and open source, we'd argue it is its sheer diversity.



1 Ubuntu 15.04

Ubuntu is a perfect place for us start our epic dive into the world of distributions because this is the one that's done more than most to make Linux accessible, easy to use and friendly. It pioneered installation from a live CD or DVD, and would even ship these out for free if you simply asked. In a recent post on **insights.ubuntu.com**, Canonical has also stated that Ubuntu now has more than 25 million users, making it the distribution most people are likely to have heard the most about.
The latest version of Ubuntu is codenamed 'Vivid Vervet' – a ververt being a kind of monkey. The word 'vivid' could also be a reference to the latest colour scheme, because Ubuntu is definitely getting less brown and more saturated purple with each release. Prince in his Purple Rain phase would be very pleased with the new backdrop, and we like it too.

There isn't a huge amount of difference in this release compared to either 14.10 or 14.06; it feels more like a staging environment for the migration to the *Systemd* boot system. Everyday users don't need to know about this, and won't see any operational changes, but for the engineering teams, there's a huge amount of effort involved migrating from the old system to the new, and 15.04 is proof that the team has done a good job. We experienced no problems on our installation, and the distribution upgrade even worked. Other than the background, visible differences are hard to see. As you'd expect, all the main applications have been updated, and the Unity interface has a few tweaks. It runs well and we're getting used to it, especially after using the Ubuntu Phone for a while now



With the new version of Unity, you can now place menus within an application's title bar.

Spin-off flavours

Of course, alongside the main distribution there's also the officially recognised Ubuntu spins. Kubuntu is one of our favourite KDE distributions, for example, and 15.04 is the first release of the newly ordained Ubuntu Mate. Mate is the continuation of Gnome 2, prior to the huge user-interface changes that came in with both Gnome 3 and Unity, as used in modern versions of Ubuntu. It's what helped give Linux Mint so much momentum, and it's great to see the work the team has done, as well as the acknowledgement that some people just prefer the old desktop metaphor, being recognised and supported by the original distribution.

The question of whether it's worth the upgrade or not has more to do with support than new features. And with the non-LTS releases getting only nine months of updates, we'd suggest holding back on upgrading unless you enjoy experimenting with new features and getting the latest applications. But as an indication of where Ubuntu is headed, this is still a great distro.
www.ubuntu.com

2 NIXOS 14.12

This is a fascinating distribution built around its own package manager that uses a declarative language to satisfy the desired configuration. Packages are then installed into their own isolated sandbox, making it ideal as an alternative to containers on a server. The downloadable *VirtualBox* image made testing much easier for us, before deciding on a fully fledged installation.
http://nixos.org

3 SCIENTIFIC LINUX 7.1

Produced by the Fermi National Accelerator Laboratory, Scientific is (like CentOS) built from the sources of Red Hat Enterprise Linux, which are only available as binaries with a subscription from Red Hat. This makes it incredibly stable, scalable and adaptable. It's used by many science labs around the world and is also a good choice for austere office environments and servers.
www.scientificlinux.org

4 LINUX MINT 17.1

The most successful offshoot of Ubuntu, Mint at times rivals its progenitor for popularity and influence. Despite early versions defaulting to KDE, it was Ubuntu's gamble on dumping Gnome 2.x for its own Unity desktop that gave Mint its biggest boost, and both the Cinnamon and Mate desktops offered and co-developed by Mint are still the best reasons for using it. In our opinion, they've even helped smooth the acceptance of the rather revolutionary Gnome 3.x by providing a well-supported alternative to the older version while the Gnome team got their act together.

Mint installs easily and complements Ubuntu's packages with its own apps and design, and it also runs well on older hardware. Support comes from an excellent community and the distribution is well funded, which means you can rely on a Mint installation providing a straightforward productive desktop environment that's going to last. It would be unfair to describe it as the Windows XP of Linux, especially as XP is no longer supported, but it's close. The Debian edition is also worth a try.
www.linuxmint.com

5 SOLYDXK 201501

There are probably more Debian-derived distributions in this feature than any other, especially if you include Mint and Ubuntu. Solyd's (the X or K above is for Xfce or KDE) killer feature is that it's a rolling release distro. You install once and simply upgrade to get the latest packages. It accomplishes this brilliantly, making it an easy alternative to something more technical, such as Arch.
http://solydxk.com

6 KNOPPIX 7.5

Revolutionary from the beginning, Knoppix was one of the first distributions to run in a live environment directly off a USB stick or CD/DVD. It used to default to a KDE desktop, but the move to LXDE was a sensible one, as it runs much better from slow media. The latest release is excellent, but you'll need to wait for a distribution exclusivity agreement to expire before being able to download it.
www.knoppix.org

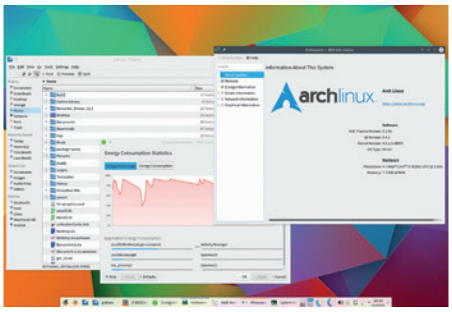
7

Arch Linux

We're big fans of Arch here at Linux Voice but it's not for everybody. For a start, in an era where distributions are often installed with one or two clicks of the mouse, installing Arch needs time and patience. You'll need to read the installation guide carefully (or take a look at our own guide in our very first issue) and adapt the installation for your own hardware. It's not difficult but it requires a different kind of mental approach, and the same could be said for running and maintaining the system after you get it installed.

There are several huge advantages to explain why you might want to do this. The first is that you'll learn a lot about how Linux works. Just connecting to the internet, for example, will teach you about your hardware, the kernel drivers and the commands needed to configure and connect them. It's the same for installing a graphical environment or building your own packages. And you'll be left with a system fine-tuned to your exact requirements, which means the typical Arch installation will occupy less space and boot faster.

Another huge advantage is that Arch is a rolling release distribution. Packages are constantly updated and can be constantly upgraded. Even hourly, if you so choose. You often need to proceed with caution, but it makes Arch the best distribution for cutting-edge releases and upgrades. The package manager is also very powerful, enabling you to roll back and forward through your locally cached downloads, and the user repository of unofficial packages has the largest breadth of choice we've seen, with even the most esoteric of projects getting a relatively painless installation package.


Arch is one of the only distributions that has the latest and greatest KDE 5 packages.

If you like building and modifying, the source and build trees that can be created are also very powerful, enabling you to automagically handle dependencies and patch only specific files.

Finally, the quality of the documentation is unparalleled, whether it's explaining how to get networking working or the differences between the various audio layers. This is a huge part of the project but it requires some skill to find the pieces relevant to your installation and requirements, as well as navigating the occasional contradictions or out of date material.

Every upgrade needs preparation, which means Arch needs some serious commitment. But get past the relative complexity and labour of running an Arch installation and we feel you have one of the best distributions available, whether that's on a Raspberry Pi or the latest PC hardware. If all this sounds too complex, there are some easier-to-maintain alternatives, such as Manjaro or KaOS. These can offer many of the advantages without the install and update hassle. But for purists like us, nothing quite beats the real thing.

www.archlinux.org

8

CRUNCHBANG PLUS PLUS 1.0

CrunchBang, the name for the # and ! symbols you typically find at the beginning of a script, was a hugely popular Debian-based distribution. It didn't need much in the way of system resources, and its minimal Openbox desktop attracted many users who wanted function over form. But when its creator, Philip Newborough, decided to call it a day, we all thought CrunchBang's days were over.

Amazingly, CrunchBang's ethos has been preserved, with new developers and a new version of Debian from which to draw packages. After negotiating the standard Debian installer, version 1.0 looks fabulously minimal, from the login screen to the desktop; everything looks like a shinier version of the command line. Many things can be launched with a simple keyboard shortcut with everything else though a right-click of your mouse. And because this release is built on the latest Debian, CrunchBang is now bang up to date with all the latest packages, making it perfect for power users who want a quick installation and a desktop with minimal distraction or interference.

https://crunchbangplusplus.org

9

CHAKRA 2015.03 "EULER"

This is a great KDE distribution that's low on system resources and provides one of the easier ways of staying on top of KDE 5 development through a specific repository. Another great feature is that while the foundation packages are updated only every six months, other packages for many applications are updated quickly, making this a great half-rolling release distro.

http://chakraos.org

10

PORTEUS 3.1

Because Porteus is a minimal, fast booting live distro, we love the way you select what you want before you even download, from EFI to desktop. You then get a personalised image. Our no-frills LXQt build was just 150MB and booted to a fully functional desktop within just 12 seconds, making it our go-to distribution when we need something quick and clickable.

www.porteus.org

11

STEAMOS

Valve's reputation is mythical. It's a games company that has not only created some of the best games ever made, but also the games platform at the heart of PC gaming. When Valve announced it was moving to Linux, bringing its games, APIs and gaming partners with it, there was a huge wave of expectation. SteamOS is its Linux distro, created to deliver seamless integration between your PC, your games and the community. It's perfect on a powerful box tucked beneath your TV, and is now relatively easy to install. It streams games from other machines, and can be augmented with standard Debian repositories. We can't wait for commercial boxes to appear.

http://store.steampowered.com/steamos

12

PCLINUXOS 12-18-2014

As old Mandrake users, will still remember Texstar's brilliant set of KDE packages that transformed the appearance and capabilities of that old distribution. And it was these packages that led to the creation of PCLinuxOS over 10 years ago. The package manager may have changed, and its old muse has fallen on hard times, but PCLinuxOS is still a distribution with the same emphasis on a great-looking desktop.

There's now a choice between KDE, Mate and LXDE, all of which feature similar styling and all the packages you'll ever need. KDE 5 has yet to make an appearance, which we're surprised at considering the distro's heritage, but it will be a great upgrade when it does.

www.pclinuxos.com

13

ANDROID X86 4.4-R2

Why include a Linux 'distribution' developed by Google to run on smartphones? It runs surprisingly well and is useful even without a touchscreen (although this does help). Android's full-screen apps and streamlined task switching makes it a great option for distraction free working, and if you get Google Play working, there are many different apps you can install and play with.

www.android-x86.org

14 OPENELEC 5.08

This is a distribution with only one job – to run the *Kodi* media centre, and it does this job perfectly. It's ideal if you want to turn your low-end box or Raspberry Pi into a movie, music and photos powerhouse, and its minimal install and automatic updates means you can stick it onto your computer and forget about it. If you need some help, see our comprehensive guide in issue 12.
http://openelec.tv

15 SYSTEMRESCUE CD

Sooner or later, something is going to go wrong with your Linux installation, and you'll need a distro with lots of tools to get you out of trouble. Ubuntu is a good option, but SystemRescueCD is our favourite. We've used this after deleting files by mistake, and even repartitioning the wrong drive. It's quick to download, fits on almost any USB key. (435MB) and is crammed full of hope.
www.sysresccd.org

16 FEDORA 22 WORKSTATION

Fedora should need no introduction. It's a candidate for being one of the most fun distributions to try, mainly thanks to the cutting-edge nature of many of its packages. It also does a brilliant job of remaining true to the projects it packages, meaning the Gnome experience is very close to what the Gnome team see themselves, with the exception of the traditional change in background. By the time you read this, Fedora 22 should be released, but at the time of writing, we could only play with early-May beta version. This included the latest Gnome 3.16 and the new kernel, as well as Fedora's updated fork of the *Yum* package manager, *DNF*. *DNF* should feel almost identical to *Yum*, but there's a lot going on in the background, including a much more efficient approach to dependency resolution.

DNF has been available for some time, but this is going to be the first release where it's the default tool for package management, and we experienced no issues with the beta. If you've yet to try Fedora, version 22 is shaping up to be a great update.
https://getfedora.org

17 PUPPY TAHRPUP 6.0 CE

Despite being based on Ubuntu 14.04, Puppy is so tiny it should even fit on a 256MB USB drive. 256MB is not a typo – with that you get a fully functional desktop, a startup wizard and barking samples. There are many apps including *AbiWord* and the *Pale Moon* browser, based on *Firefox*. If you ever need a desktop on a USB stick, this should always be in your bag.
http://puppylinux.org

18 DAMN SMALL LINUX 4.11RC2

If you think Puppy's 256MB footprint is small, wait until you try this – a full Linux environment from 50MB. Sadly, DSL hasn't been updated since 2012, but it still deserves a place here simply because of its size and functionality; MP3 playback, web browsing, SSH, email and text editing. It's not pretty, but what do you expect for a distro smaller than the PDF version of this magazine? **www.damnsmalllinux.org**

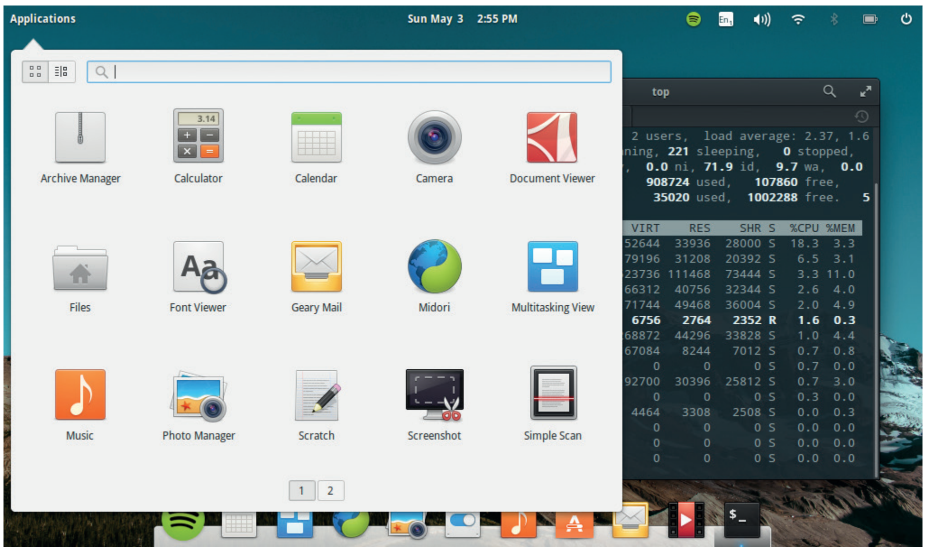
19 Elementary OS Freya

Elementary is an Ubuntu LTS (14.04) based distribution that's trying to do things a little differently. This is obvious when you take a look at its home page, as it proudly lays down the gauntlet by stating it's "A fast and open replacement for Windows and OS X." There's only a single mention of Linux on this page, and that's in the small paragraph headed "Safe & Secure."

The team has also courted controversy recently. To get to the SourceForge hosted download link, you need to navigate through a payment box that defaults to $25. If you want the download link without paying anything, you need to escape this box, choose 'Custom' and enter $0 manually. This is a process the developers originally described as 'cheating' in a part of a blog post now deleted.

One of the reasons for this is that Elementary OS is attempting to be a commercial success, using paid developers to create a custom desktop environment and user interface, complete with its own modified window manager and desktop interface. This desktop is definitely one of the best reasons to install Elementary OS because it looks beautiful. And unlike the good looks that come with something like KDE, appearances in Elementary are mostly good design choices rather than eye candy.

We installed the latest release on a low powered (Atom-based) Samsung NC-10 netbook and performance was better than running Mint on the same machine. The same applications loaded faster and were slightly more responsive.



```
 2 users,  load average: 2.37, 1.6
nning, 221 sleeping,   0 stopped,
,  0.0 ni, 71.9 id,  9.7 wa,  0.0
  908724 used,   107860 free,
   35020 used,  1002288 free.    5

 VIRT    RES    SHR S  %CPU %MEM
52644  33936  28000 S  18.3  3.3
79196  31208  20392 S   6.5  3.1
23736 111468  73444 S   3.3 11.0
66312  40756  32344 S   2.6  4.0
71744  49468  36004 S   2.0  4.9
 6756   2764   2352 R   1.6  0.3
68872  44296  33828 S   1.0  4.4
67084   8244   7012 S   0.7  0.8
    0      0      0 S   0.7  0.0
92700  30396  25812 S   0.7  3.0
    0      0      0 S   0.3  0.0
 4464   3308   2508 S   0.0  0.3
    0      0      0 S   0.0  0.0
    0      0      0 S   0.0  0.0
    0      0      0 S   0.0  0.0
```

Freya looks beautiful, runs well on older hardware and offers access to Ubuntu's repositories.

Custom applications

Applications themselves are launched from a simple panel that hides itself at the bottom of the screen. The top panel is equally unobtrusive, defaulting to transparency and showing just the time, status icons and the applications launch menu. It all looks fantastic and is very easy to use.

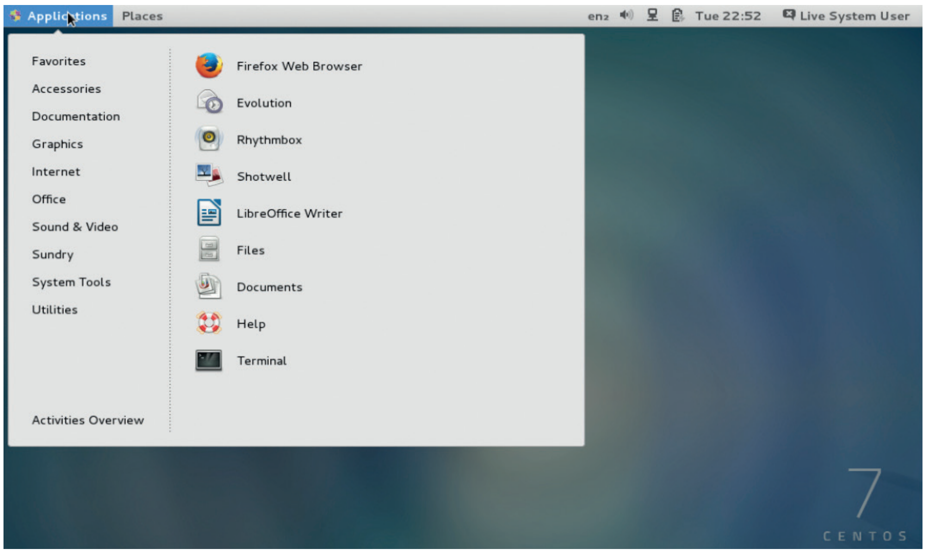
OS X migrants in particular are going to feel at home with the grey window decoration, settings panels and task switcher. The choice of applications is also good, with *Geary* making a well deserved appearance for email, and the terminal emulator is one of our favourites. Everything about Freya is easy.

Thanks to its Ubuntu heritage, installation is a breeze. You can connect to your wireless network and download updates, and Freya detected our Mint installation with a warning about not overwriting it. Package repositories are also Ubuntu's, giving plenty of choice for software and PPAs. Freya is a seriously strong contender for non-Linux users and Linux users with an appreciation for aesthetics – if you're prepared to forgive the developers for their occasional lack of tact in getting people to pay for their work.
https://elementary.io

20 CentOS

CentOS, or the Community Enterprise Operating System, is a distribution built on the code to Red Hat Enterprise Linux (RHEL). That means it's just as secure and reliable as RHEL, and also gets updates for 10 years, but unlike RHEL, CentOS is available for free. This combination of features means it's ideal for servers for any organisations that don't need the level of support that Red Hat offers. In fact, we trust CentOS so much, it's the distro we use for our Linux Voice servers. We're not alone in this. According to **w3techs.com**, CentOS is the third most popular Linux distro for web servers (after Debian and Ubuntu) and 20% of all Linux webservers run CentOS.

The only real differences between CentOS and RHEL is in the branding (the Centos team have to remove all trademarks before they recompile the source of RHEL and release the binaries). The two distros behave the same in almost all technical areas, so CentOS is also an excellent choice for any aspiring sysadmin. This compatibility also means that almost all Linux server software is supported on, and packaged for, CentOS. Whether you're running the latest-and-greatest open source code, or some closed-source proprietary server, you can guarantee that if it runs on Linux servers, there'll be a version for RHEL and Centos. The Extra Packages For Enterprise Linux (EPEL) repositories that are provided by the Fedora project to provide additional software for RHEL are compatible with CentOS.



CentOS 7 comes with a Gnome 3 desktop that's been customised to the Gnome 2 style.

Originally, the CentOS project ran entirely separately from Red Hat. However, in January 2014, CentOS announced that it was joining Red Hat (although it will still remain separate from RHEL). This should mean that updates continue to come in a timely manner, and it guarantees that the project will continue and not wind up leaving CentOS installs unsupported.

Enterprise features – for free!

The most recent version (7) only supports 64-bit x86 processors, but anyone still running 32-bit servers can continue to run CentOS 6, as support will continue until 2020. There are plans to support ARM processors as well (both 32-bit and 64-bit), but at the time of writing, there were no official version for this architecture.

While CentOS is often seen as a server OS, it can also be used as a desktop. The long, slow release cycle means that you won't have the latest software (including drivers for newer hardware), but once it's installed, you don't need to worry about upgrading the distro for a long time. Leaving a desktop operating system installed for over five years might seem anathema for most geeks, but for ordinary desktop users, this isn't unusual. We wouldn't recommend CentOS for everyone, but it certainly has a place.
www.centos.org

21 TINY CORE

Tiny Core's special feature isn't just that it's really small, it's that it's small enough to run entirely from memory. This means that the entire OS is loaded into RAM at boot time, and the result is a blazingly fast system. Even on a slow computer, Tiny Core can start in just a few seconds, and applications open almost instantly. By keeping things really simple, Tiny Core speeds up old PCs.
http://distro.ibiblio.org/tinycorelinux

22 CLONEZILLA

Clonezilla only does two things: it makes clone images of hard drives; and it re-images hard drives. It may seem strange to build an entire distro just to do this, but that's because to work with hard drives at this level, they have to be unmounted. By running as a live distro, Clonezilla allows its tools full access to the hardware. It's a great example of how the flexibility of Linux pays off.
http://clonezilla.org

23 BODHI

Bodhi Linux is a distro for the Enlightenment desktop. Or at least it was until 28 April 2015, when the developers decided to switch to the Moksha desktop. This isn't really much of a change, as Moksha is a fork of the E17 Enlightenment desktop. Moksha is born out of a frustration with a deterioration in the stability in more recent version of the Enlightenment desktop, so it aims to go back to simpler, more stable times and provide users with what they really want: a desktop that looks incredibly pretty and that just works.

The great feature of Enlightenment (and we hope Moksha) is that it provides a good-looking interface without overly taxing the CPU. The amount of eye candy Bodhi manages to pack in while barely taxing the CPU is truly impressive, especially on older computers that struggle to run KDE with any form of effects enabled. It goes to show that clever coding, not more graphical processing power, is the best way to get a great desktop experience. While other distros have Enlightenment in their libraries, Bodhi is built specially for it.
www.bodhilinux.com

24 YGGDRASIL

Between 1992 and 1995 Yggdrasil was one of the most popular Linux distros, but there hasn't been a release in 20 years. Back then, Linux was a bit rougher around the edges, and supported less hardware, but the brave soles who ran Yggdrasil despite its many limitations, fixing bugs as they came up against them, are the pioneers of the Linux systems we have today.
No website

25 KAOS

Linux is about choice, right? Not according to this distro. The Kaos developers have selected what they view as the best applications available for each task and only packaged them. For example, KDE is the only desktop. This means that the repositories contain much less software than most distros, but it's all the best of its type, leading to a very focused distro.
http://kaosx.us

26 CYANOGENMOD

Android is the most popular Linux distribution, but like all popular Linux distributions, it has derivatives. The best known of these is CyanogenMod. This is built from the source code from the Android Open Source Project (AOSP), which contains the code to the core of Android. Not everything in the commercial version of Android is in the AOSP; many of the default apps such as the mail reader and the camera are proprietary, so new versions of these have been written for *CyanogenMod*, and in most cases they're more powerful than the default Android ones.

There are community builds of *CyanogeMod* for a huge variety of devices (see **http://wiki.cyanogenmod.org/w/Devices** for details). There is often support for more recent version of Android than the device manufacturers provide, so it can be a great way of getting the latest-and-greatest features on older phones. It's also a good way of getting a version of Android that doesn't have all of Google's services syphoning off data for their advertising engine (although you can install Google services if you wish).

www.cyanogenmod.org

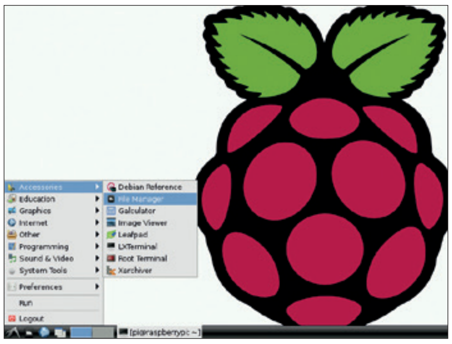
27 Raspbian

Most Linux distros are designed to run on a variety of computers, but not Raspbian: this distro is carefully crafted for just one device, the Raspberry Pi. Actually, that's five devices (the Raspberry Pi models A, B, A+, B+ and version 2 model B), but they all share a lot of common hardware. Some other manufacturers have managed to persuade Raspbian to run on their hardware, but the distro is so closely tied to the Raspberry Pi that this is never going to be a good idea.

By focussing on just a single hardware platform, the distro can take full advantage of it. Since the Pi has, by modern standards, quite a low-power processor, Raspbian is adept at squeezing as much power out of it as possible, and runs more efficiently than most distros. It contains drivers for the VideoCore graphics processor; much of the software contains platform-specific optimisations; and the repositories contain utilities for controlling the GPIO pins.

Raspbian's web browser is a perfect example of how targeting just a single hardware platform benefits the distro. Raspbian comes with a fork of the *Epiphany* browser (aka *Gnome Web*) that's been modified run far better on the Raspberry Pi's unique hardware. This means it is optimised for ARM v6 and v7 processors, while also offloading graphical processing to the VideoCore. The result is a far nicer web experience than you should really expect given the rather limited hardware.

The Raspberry Pi is designed for education, and Raspbian comes with some software that's custom-made for this. There's a special Raspberry Pi build of *Minecraft* that enables users to interact with the world through Python (and other



The LXDE desktop in Raspbian is simple, easy to use and runs well on low-powered hardware.

languages). This is particularly appealing to children starting programming.

Another child of Debian

Raspbian also comes with a special build of Wolfram's *Mathematica* (a mathematical programming system). This is widely used in schools because of its ease of use, and the amount of data included as well as mathematical functions.

The Raspberry Pi Foundation continue to push more optimisations into Raspbian, so more recent versions run faster than older ones even when running on the same hardware. Raspbian isn't built from scratch. As the name suggests, it's based on Debian, so as well as all the additional Raspberry Pi software, there's also the full set of Debian software. You should find whatever you need in the repositories.

There are other distros available for the Raspberry Pi, but for most users, most of the time, Raspbian is the best option. In fact, it's so good, it's arguably the best reason to choose the Raspberry Pi over other small ARM-based computers.

www.raspbian.org

28 ROSA

Rosa is developed by the Russian company Rosa Labs as a cutting-edge KDE distribution. Its best feature is that it has a really good-looking desktop right out of the box, including a large number of customisations to the default KDE look. It's a great option if you like the idea of KDE, but struggle to find a setup that you like.

www.rosalab.com

29 CORE OS

The future of the data centre is in containers – at least, it is if you believe the hype. Core OS is a distro built from the ground up for this new world of Docker (or Docker-like) applications running in isolation from each other (see the tutorial on Docker on page 96). CoreOS is designed to run in large groups of containers all intercommunicating to make the cloud run smoothly.

https://coreos.com

30 OPEN MEDIA VAULT

A Network Attached Storage unit (NAS) is a device that you plug into your house's network, sharing some hard disk space with all the other computers that are connected. This makes it easy to back up and share files between all the computers in the building. Open Media Vault (OMV) is a distro that converts a regular PC into a NAS with an easy-to-use web interface.

www.openmediavault.org

31 TAILS

We're being spied on. Our internet providers are keeping detailed records of what websites we visit. Our governments are monitoring who we're contacting. Advertising companies know more about us than our closest friends. It's time to fight back. Tails is a distro built around *Tor* to make it easy to browse the web without being tracked. We like to keep a copy of the latest Tails ISO in a virtual machine when we want a private web browsing session.

Tails includes software for encryption and private instant messaging, so it's got a full suite of privacy tools to keep any communications secure. Download Tails and protect yourself from prying eyes.

https://tails.boum.org

32 TRISQUEL

Most distros include some non-free software, often for device drivers or media players. Trisquel is different. It includes only 100% free software, and doesn't have any non-free repositories. This does mean it works with less hardware than other distros, but you're fully in control of your computer.

http://trisquel.info

33 ZORIN OS

Zorin is a distro built to ease the journey into Linux from Windows. It focuses on providing a familiar interface so that new users aren't put off, and using the Zorin Look Changer application, you can customise the distro to match different versions of Windows.

http://zorin-os.com

34 Kali

Computer security is possibly one of the most important issues facing the digital world. Governments are attacking each other; companies are attacking each other; and bands of digital bandits are attacking regular web users. If you work in computing in any way, it's important that you understand the issues in computer security, and to do that, you need a distro built for IT security professionals. That distro is Kali Linux. Kali's main function is to help penetration testers – the people who try to break into an organisation's computer systems in order check the defences.

Kali includes the *Metasploit Framework* (which we looked at last issue) for attacking servers, the *Social Engineer's Toolkit* (*SET*) for attacking personnel, tools for making malicious hardware, and dozens of other pieces of security software, all set up and ready to go. Most useful tools come installed by default, so you just need to browse the menus to find the best tool for the job and you're ready to start attacking. In fact, browsing the menus of Kali is a great way of seeing all the open source security tools that are available.

Industry standard

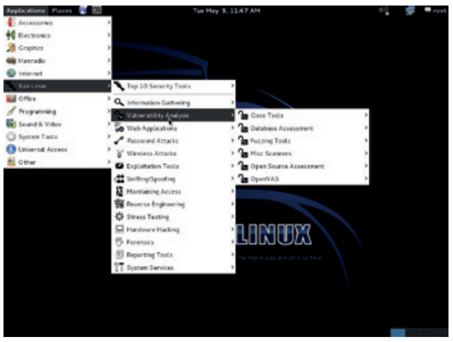
Since it's so widely used, most books and other resources on penetration testing use Kali (or BackTrack as it used to be known) in their examples, so it's the easiest penetration testing distro to get started with.

There are builds of Kali for quite a lot of different platforms, including small ARM-based computers such as the Raspberry Pi, the Odroid U2 and the Beaglebone Black. These small devices are particularly interesting from an attacker's point of view because they're small enough to be hidden inside an office and will run for a considerable amount of time on battery power alone. Using Kali in this way enables penetration testers to make devices that will capture information about wireless traffic, or attempt to break in from inside a building with no one physically present. There have even been some projects that have performed penetration tests by putting these small computers inside remote control planes, and launching them from a distance.

Kali also runs on mobile devices such as tablets and Chromebooks. These give attackers the power of a full pen testing environment while maintaining the appearance of performing some mundane task. After all, who would suspect that someone tapping away at a tablet has access to a range of sophisticated pen testing tools?

Kali Linux is a distro that every security-minded Linux user should try out – if only so they can see the range of tools available to attackers

www.kali.org



Kali Linux provides everything a penetration tester may need in a live-booting distro.

35 MIKE OS

There are many reasons to create an operating system: a desire for freedom, a belief that you can do it better than the existing options, and as a hobby. MikeOS was created to demonstrate the usage of x86 assembly language. The standout feature of MikeOS isn't any technical aspect, or a flashy user interface, but the fact that the source code is clear and well commented.

Linux Voice's Mike Saunders created Mike OS. You may have read his series on assembly programming (continued in this issue), where he goes through the techniques he used to develop Mike OS. If you have, and you want to learn more about how to program on bare metal, then MikeOS is the best place to start.

As operating systems go, MikeOS is quite limited: there's a text editor, and a few other utilities, but not much in the way of general productivity software. Mike tells us that networking is possible, but it's not easy. In other words, don't install it and expect it to replace a general purpose OS (even Mike uses a Linux distro for most of his work).

http://mikeos.sourceforge.net

36 MINIX

Originally, Minix was a purely academic distro created by Andrew Tanenbaum to help teach students about OS kernel design. However, in recent versions (starting with Minix 3 in 2005) the aim has shifted to creating a real-world OS. The main advantage of Minix over other Unix Kernels is that it's designed using the microkernel, model which should, theoretically, lead to a far more stable and self-healing OS.

The code is now available under the BSD licence, and version 3.3 (released in september 2014) came with support for the ARM architecture as well as x86.

www.minix3.org

37 OPENINDIANA

OpenIndiana is a distribution of Illumos, which is itself a fork of Open Solaris. This means it's a Unix OS that's from a separate lineage to Linux and the BSDs. Perhaps the two best reasons for using OpenIndiana (aside from academic interest) are that it's Solaris that gave us the ZFS filesystem and Solaris Jails (a sandboxing environment). Other free Unixes are catching up in this area now, but Solaris and derivatives were for a long time significantly more advanced than their cousins. All these technical differences aren't immediately apparent as OpenIndiana boots up to a Gnome 2 environment that will be familiar to many Linux users, so it's easy to get started.

http://openindiana.org

38 GNU HURD

When GNU first launched its project to re-implement UNIX in the 80s, it needed a kernel. At this point the Linux kernel didn't yet exist and the BSD kernels were legally uncertain, so GNU set out on a new project: Hurd. It's taken a long time to become useable, but version 0.6 (released in April 2015) brought in more stability.

Hurd isn't the kernel itself, but a set of servers and protocols that sit on top of the Mach microkernel. This microkernel model is often cited as the cause for delays in the kernel's release, since it has led to more complexity than the monolithic architecture of Linux. However, you can now try the Hurd in a port of Debian.

www.gnu.org/software/hurd

39 SABAYON

If you like the idea of Gentoo, but don't want to install everything from scratch, Sabayon is for you. It sticks close to the bleeding edge of software development, but also tries to be stable. The team pride themselves on how good the distro looks out-of-the-box, so just install and go.

www.sabayon.org

40 DEEPIN

Deepin isn't well known in the English-speaking world, but it's popular in China. As well as a customised desktop, it features the *Deepin Software Centre*, which enables users to rate and comment software. Most of the comments are written in Chinese.

http://planet.linuxdeepin.com

41

Debian GNU/Linux

Virtually everyone has heard of Debian, and most Linux users have tried it at least once, but what makes it so successful? It's not backed by a big company, it's rarely the starting point for new Linux users, and it lacks the snazz and pizazz that many other distributions have.

Well, we'd say this: Debian is the closest thing we have to a "standard" among distributions. It's one of the longest-running distros out there, it's incredibly well respected, and along with CentOS it's the go-to distro when you want to put Linux on a box and then forget about it for the next five years – updates included.

Debian is conservative and slow moving, but you know that it has been extremely well tested and won't break with the next round of updates. A Debian release will stay the same across its lifespan; patches are provided to plug security holes and fix critical (ie data-loss) bugs, but you won't have to deal with software versions changing under your feet.

Debian also prides itself on being a "universal" operating system. In other words, it's designed for everyone, and not just a niche of users. This manifests itself in multiple ways: Debian puts great effort into supporting users with disabilities, so that people with vision problems, for instance, can still install it. Similarly, Debian has goals beyond merely providing a Linux distro, and tries to be a framework for other operating system projects, such as Debian GNU/kFreeBSD (the Debian and GNU userland combined with the FreeBSD kernel). Many pundits have criticised Debian for putting time and effort into these niche projects, but we think they're important.

Some Debian developers have welcomed the switch to *Systemd*. Others fear a total-control 1984-like scenario, engineered by Red Hat. (Image credit: **http://blog.desdelinux.net**).

Freedom is strength

Sure, very few people are going to use Debian GNU/Hurd or the FreeBSD variant. But consider the IBM vs SCO lawsuit back in the 2000s, when the latter company insisted that proprietary Unix code had somehow found its way into the Linux source tree. SCO effectively tried to own Linux and get users to pay licence fees. Those were bad times, but IBM came out on top.

Now imagine such a scenario happens again, with a much bigger company trying to assert ownership of Linux, or even have Linux declared as a copyright-infringing work. This is very unlikely, but crazier things have been known to happen...

If the Linux kernel somehow gets tangled up in these legal wranglings, and they take years to sort out, thanks to the other Debian offerings we'll still be able to enjoy the benefits of a free operating system. The FreeBSD kernel is very well regarded, and combined with the GNU userland you have an extremely impressive OS. Bring on more projects like that, we say!

www.debian.org

42

MANJARO

Many Arch fans claim that the distro is easy to install, thanks to its extensive and superb documentation, and there's some truth in that. But you still need a good grounding in Linux and related technologies, and if you just want to set up a box with a rolling release distro in 10 minutes, the process is somewhat long-winded.

Manjaro is one of many Arch forks that aim to preserve the best aspects of the distro, but make them more accessible to new and intermediate Linux users. So Manjaro has a graphical installer, hardware detection tools, and other features that simplify and accelerate the process of getting the distro installed.

Although Manjaro hasn't hit version 1.0 yet, it's already a very polished and usable linux distribution, and well worth a try if you're tempted by Arch Linux but want something easier.

https://manjaro.github.io

43

GENTOO

Back in the early 2000s, Gentoo was the absolute darling of power users – and for good reason. It was immensely configurable, as you were encouraged to build everything from scratch. You could change CFLAGS to compile packages with specific optimisations for your CPU, and you could enable or disable custom features for your setup. On top of this, it was one of the earliest rolling-release distros: you got new software as it came down the pipeline from upstream developers, instead of waiting for another big distro upgrade in six months.

So what happened? Why isn't Gentoo dominating today? Well, a big chunk of its userbase moved over to Arch Linux. While playing around with CFLAGS was fun, many of the optimisations made very little difference, and the most passionate Gentooers were mocked as "ricers" – in other words, the Linux equivalent of car modders who add go-faster stripes. Still, Gentoo lives on and is one of the few distros that hasn't adopted *Systemd* by default, so we're sure it'll be around for a while.

www.gentoo.org

44

HAIKU

This isn't Linux-based, but it's a project with bags of potential nonetheless. Haiku is an open source reimplementation of BeOS, a desktop OS that gained mild popularity in the late 90s. BeOS was strikingly fast and multimedia-friendly back in the day, and while Haiku is still undergoing heavy development, it could prove to be a great lightweight OS one day.

www.haiku-os.org

45

DEVUAN

Debian's switch to *Systemd* wasn't received well by the whole distro community. A few disgruntled developers left the Debian project to start Devuan, a distro for "init freedom lovers". Whether this will turn out to be a serious project – or just a knee-jerk reaction that leads to nothing – remains to be seen, but there has certainly been plenty of chatter on the mailing lists.

https://devuan.org

46 ELIVE

Before the KDE and Gnome dektop environments took off in the late 90s, Enlightenment was the top-tier window manager in Linux, with impressive effects such as window translucency and gorgeous themes. Enlightenment is still going today, and Elive uses it on top of a Debian base to create a distro that's good looking, featureful and relatively low on system requirements.

www.elivecd.org

47 PEPPERMINT OS

Chromebooks are increasingly popular among non-technical users, offering a stripped-down operating system that provides most of its functionality via web apps. Peppermint does a similar job as a "web centric operating system", except you don't have to submit to Google's giant data collection machine. It's based on Ubuntu's Long Term Support releases.

www.peppermintos.com

48 LXLE

Talk about standing on the shoulders of giants. LXLE is based on Lubuntu, which is based on Ubuntu, which is based on Debian. It might seem crazy to have so many spin-offs in the distro world, and we'd agree that some distros offer little more than wallpaper changes, but LXLE is well worth looking at. The developers have put great effort into a consistent theme and layout for the LXDE desktop, while the default app selection has been carefully chosen.

Most importantly, LXLE is great for reviving older machines that have limited RAM: it will run pretty well in just 512MB. So if you have an old netbook lying around doing nothing especially useful, pop LXLE on it and bring it back to life. Or if you know someone still running XP on an older box, you could install LXLE for them and bring them up to date. There are plenty of other lightweight distros out there, but LXLE is one of the most polished we've come across, and as it's based on the LTS releases of Lubuntu (for long term support), you know it will receive security patches for many years to come.

www.lxle.net

49 ORACLE LINUX

Ask database giant Oracle what the best OS is for running its products, and you'll be told: Solaris, of course! But Oracle has had to accept that its proprietary Unix isn't the be-all and end-all on servers, and many people want to run Oracle's DB on Linux. So the company forked Red Hat Enterprise Linux (see opposite page) and added kernel tweaks to handle huge workloads.

www.oracle.com/linux

50 FREEBSD

It's not Linux, but it's a free and open source Unixy operating system – very much like Linux. In FreeBSD, the whole operating system is developed inside a single source code tree, in contrast to Linux where the kernel, C library and base utilities are from separate projects. FreeBSD sees plenty of use on servers, and it's fairly good on the desktop with the right hardware.

www.freebsd.org

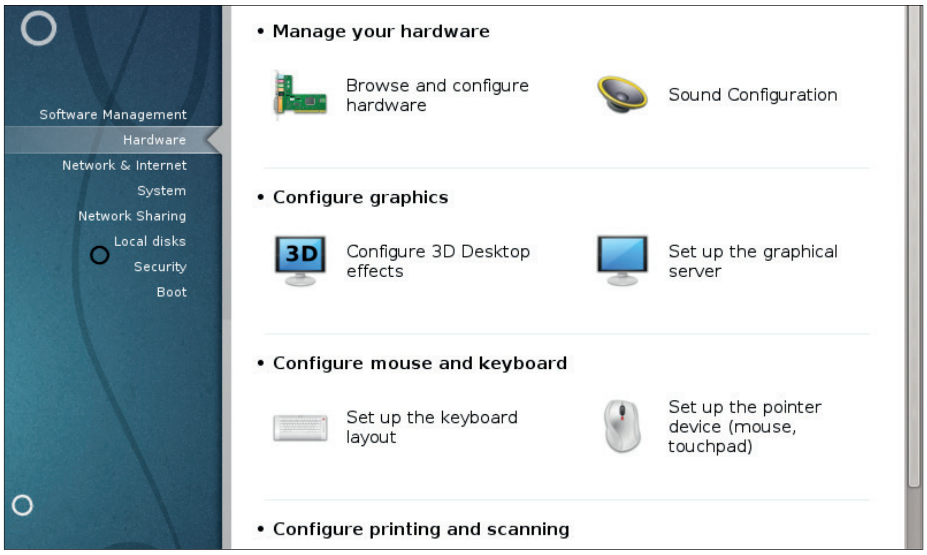
51 MAGEIA

Mah-jee-a? Ma-gay-a? Mah-gaia? There's no official way to pronounce this distro's name, which we regard as something of a marketing fail, but we'll let that minor grumble pass. Mageia is one of the most prominent forks in Linux distro history, and unlike many projects, it didn't just happen because of a squabble between developers.

No, Mageia came about from necessity. In the 2000s, one of the most popular distributions among new users was Mandriva, formerly known as Mandrake Linux. This was a long-running distro originally based on Red Hat Linux, and was noted for its slick desktop, excellent hardware detection and newbie-friendly Windows Control Panel-esque setup tool. Also, it was one of the few distros to offer snazzy boxed sets with CDs, printed manuals and customer support.

We were big fans of Mandriva back in the day, but unfortunately the company wrestled with financial problems for many years and ended up laying off most of its distro developers. A bunch of them wanted to carry on with their work, however, and Mageia was formed. Initially, many observers suspected that the Mageia project would go nowhere, but time has proven them wrong and it's a very fine distro today.

Like its predecessor, Mageia excels with its *Mageia Control Center* configuration tool and novice-friendly installer. Users can choose between KDE, Gnome and Xfce desktops, all of which are given Mageia-specific theming, and the development team has done a good job pumping out regular releases over the last few years. In a fascinating turn of events, the Mandriva company, which still exists (albeit in a much smaller form than in the past), is now using chunks of Mageia in its own Business Server product. Who knows – maybe the projects could join together again one day...


Software Management
Hardware
Network & Internet
System
Network Sharing
Local disks
Security
Boot
• Manage your hardware
Browse and configure hardware
Sound Configuration
• Configure graphics
Configure 3D Desktop effects
Set up the graphical server
• Configure mouse and keyboard
Set up the keyboard layout
Set up the pointer device (mouse, touchpad)
• Configure printing and scanning
Mageia's Control Center is a one-stop shop for all things system configuration.

Mandriva lives on!

There's another spin-off of Mandriva called, innovatively enough, OpenMandriva. Initially the goals of that distro and Mageia were so well aligned that it seemed crazy to have two distros doing exactly the same job. Sure, forks happen in the free software world, and often for good reason, but is it ever possible for two projects to merge?

We asked the Mageia team about this at the FOSDEM conference in February, and yes, there had been talks in the past about merging the projects, but today their goals are increasingly diverging. Mageia is largely focused on being the spiritual successor to Mandriva, targeting new and intermediate desktop users, while OpenMandriva is doing much more experimental work by switching to the *LLVM/Clang* compiler and targeting ARM devices. Don't expect a merge any time soon, but both projects are doing great work.

www.mageia.org

52

ANTERGOS

"Your Linux, always fresh, never frozen." That's the motto for Antergos, and reflects that it's a rolling-release distro based on Arch Linux. But unlike Arch, it tries hard to attract new and intermedite Linux users, with an attrative website and polished desktop environment configurations. Antergos has its own installer, *Cnchi*, and could be a big player over the next few years.

http://antergos.com

54

NETRUNNER

There are a million and one KDE-based distros out there, but Netrunner distinguishes itself by actively supporting KDE with financial help – a rarity in Free Software. There are two flavours of Netrunner: one is based on Kubuntu and has regular six-monthly releases, while the other is based on Arch and is therefore a rolling-release distro.

www.netrunner.com

55

AROS

We have to include this here (even though it's not based on the Linux kernel), just because we still get misty-eyed when thinking about the Amiga. Yes, this is an open source implementation of AmigaOS, running on modern PCs. It's lacking a lot of hardware support compared to Linux, but it's impressive and brings back great memories of the glory days of Workbench.

http://aros.sf.net

56

SLACKWARE

Slackware is the oldest Linux distribution that's still running, and is largely the project of one man: Pat Volkerding. This might seem like an impossible feat given the huge teams behind Debian, Fedora and other big-name distros, but Slackware is very simple under the hood. Its packages are tarballs with some metadata, its boot scripts are short and BSD-like, and the whole distro has an air of simplicity and elegance that makes it a joy to work with. It's not easy going for newbies, but because it doesn't make big changes to upstream code, you get a very vanilla Linux experience without distro-specific "features" causing trouble.

Indeed, one of the favourite sayings among users of this distro is: "With Red Hat you learn Red Hat. With Ubuntu you learn Ubuntu. But with Slackware, you learn Linux." Slackware fans are noted for their eclectic taste in humour (see the Church of the Subgenius) and general apathy towards converting others. Have you ever seen a foaming Slacker trying to win over users on an internet forum? Of course not – that simply ain't the Slackware way.

www.slackware.com

53

OpenSUSE

If you were using Linux in the late 90s and early 2000s, you may recall the chunky boxed sets you could buy from various distro vendors, containing CDs/DVDs, books, stickers and other goodies. OpenSUSE (known back then as just SUSE) was the best distro in this respect: you'd get a shiny card wallet jammed with discs containing thousands of packages, along with three thick manuals teaching you everything you needed to know about the OS. Getting one of these boxed sets through the door was bliss, especially if you had only a dialup modem connection.

OpenSUSE is one of the longest running distros, coming to life in Germany in 1992 as "Software und System Entwicklung" (software and systems development), and originally based on Slackware. The SUSE company grew quickly, establishing itself as the main competitor to Red Hat, and was eventually bought by Novell in 2003 for a cool $210m. Novell itself was later snapped up by The Attachmate Group, and today SUSE operates as a subsidiary, selling enterprise Linux solutions to big business.

Amongst hobbyists, however, the OpenSUSE distro is still going strong. Its flagship feature is *Yast* (Yet another Setup Tool), a graphical and command line program which handles virtually every aspect of system administration. While most distros have a disparate bundle of tools for handling such things as user management, startup services, package installation and so forth, *Yast* provides all of these facilities – and much more – from the same place. Yes, some would argue that such a large, monolithic program goes against the philosophy of Unix, where small

13.2
openSUSE
13.2 openSUSE
umfassend · einfach · stabil
Mit gratis E-Book und weiteren Extras!
Das umfangreiche Linux-Paket mit mehr als 1000 Anwendungen für Einsteiger und Geeks

OpenSUSE is still available in boxed set form – at least, for the German market.

tools work together, but we can see the attraction in *Yast*. Whenever you need to do some kind of administration work on your installation, you know exactly where to start.

Roll your own

Another ace that OpenSUSE has up its sleeve is its Open Build Service. This is an online resource where developers can upload code and build packages for OpenSUSE and other distributions including Fedora and Debian, without having to install the distros and the (often rather complicated) set of build tools manually. It's great to see the OpenSUSE team playing such an active role in distro cross-pollination efforts – and making life easier for app developers who want to package up their work for as many distros as possible.

Along with the regular release versions of OpenSUSE, there's also a "tumbleweed" flavour that's a rolling release, much like Arch Linux. It's not guaranteed to be as stable as the well-tested releases, but it's great if you want to live on the bleeding edge and get the latest and greatest applications.

www.opensuse.org

57

RED HAT ENTERPRISE LINUX

Here's a distro that gets relatively little coverage in Linux Voice, as we tend to focus on home users and tinkerers, but it's the biggest flavour of Linux in the corporate world. Red Hat Enterprise Linux (RHEL) is based on work done in the Fedora project, and comes with support contracts that can cost a bunch of bucks.

But why pay for Linux when you can get it for free? Well, with RHEL you know there's someone on the phone to fix any problems you may come across. Businesses need this, or at least someone they can point the finger at. Red Hat Enterprise Linux gets 10 years of support as well, so while costly, it's a sound investment.

www.redhat.com

58

SLITAZ

Desktop Linux in just 35MB – is that even possible? With SliTaz, yes. You get a very trimmed-down GUI with a handful of apps such as the *Midori* web browser, and it all runs in RAM at lightning speed. It's great to carry on a USB key and boot up on someone else's PC to show Linux's awesomeness.

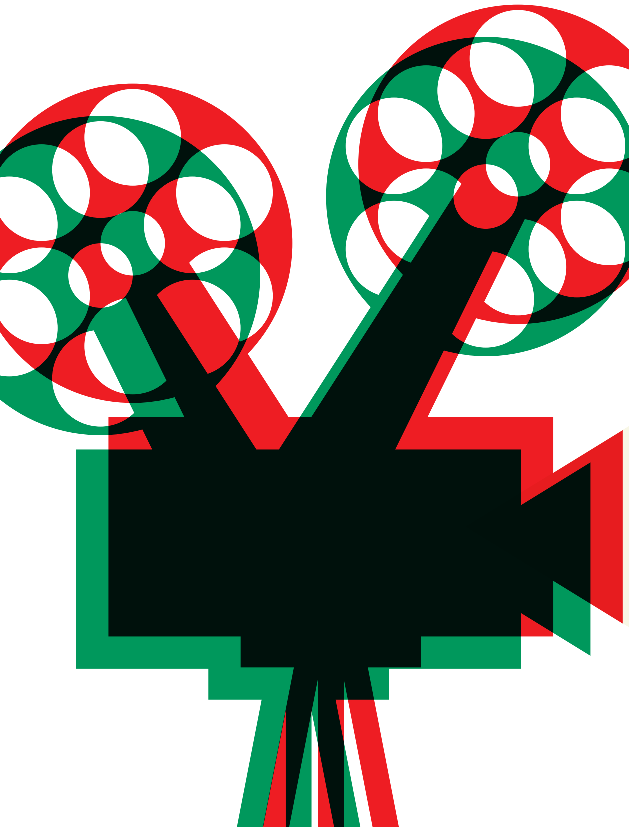
www.slitaz.org

59

LINUX FROM SCRATCH

So those are 58 distros – and now it's time to create your own to add to the pantheon! Linux From Scratch is a guide and repository of source code, explaining in detail how to install Linux by hand, piece by piece. Give it a go – you learn a lot...

www.linuxfromscratch.org



LINUX ON FILM

Graham Morrison spends a week on YouTube to discover the ghostly corners where Linux has appeared on the big screen.

Back in the late 1980s, I worked in what used to be known as a 'Video Shop', or a 'Video Rental Store'. This was a real, physical place you could visit by walking down your High Street, or by taking a short drive to your local strip mall, and entering an actual door. These places would let you browse and borrow the latest films for a single night, usually for around £2.99, with a discount for a second title on Sunday. You'd be handed the film and expected to return it the following morning, either by going back to the shop or by slotting it through an appropriately sized hole in the door.

We issued fines for people who didn't rewind their films (they were all on bulky VHS cassettes), or were late, or who failed to collect a reserved title.

Despite a salary that barely covered Thursday's night's Pernod and black, this was one of the three coolest jobs in town (the other two being similar positions at the other video rental stores in Ashby-de-la-Zouch). You could watch whatever you wanted, usually as soon as it was released, and spend quiet days drinking tea and watching child-friendly titles on the in-store screen; *Labyrinth*, *Short Circuit*, *The Neverending Story*, *Indiana Jones*, *Twins*... I can still remember the rental number attached to the spine of each case. It was a time before the pressure of online piracy brought the consumer releases of films far closer to the theatrical release, which made videos feel rather exclusive for those who may have missed the cinema release.

"Some computers, such as IBM's AN/FSQ-7, bridged decades with its on-screen menace."

Stars of screen and stage

For those of us into computers, conversations around video often surround the depiction of computers on the screen. This being the late 80s/early 90s, there were an increasing number of films about and featuring computers, and it was also obvious that computers and the way they were portrayed in film was becoming a reflection of how computers were perceived and how that perception was changing. A couple of decades before, that meant a background of vacuum tubes and blinking lights, or a room full of clinical white hardware. HAL in Stanley Kubrik's 1969's science fiction classic *2001*, for example, was famously authoritarian and represented a scary dystopia for the new machines taking over payroll and banking. Some computers, such as IBM's AN/FSQ-7, bridged decades with its menace – from Fantastic Voyage in 1966 to Dr Evil's submarine lair 2002's *Austin Powers*.

But the microcomputer changed all that with its increasing affordability. 1983's *War Games* , for

MAY CONTAIN SPOILERS!

instance, bridged the gap between the old menace of technology and the rise of this new subculture. Its virtual protagonist was WOPR, a pseudo-intelligent computer system designed to predict war strategies while having a Telnet connection to the nuclear button. Mutually assured destruction was a simulation away. But its depiction of Matthew Broderick's hacker, using an IMSAI 8080 to dial into random computers, is relatively accurate and more representative of what other crackers were doing. This focus on what the population were doing with their new computers was the beginning of a new age.

The early microcomputers, there was no choice of operating system: you got whatever came with your Apple II, Commodore Amiga or Macintosh. But by the 90s, that the operating system was becoming a choice, and a choice that filmmakers could use within their films. Windows, for example, was often the backdrop for business, whereas Mac OS was seen with designers. Linux and UNIX, when they appeared, were for geeks. And its appearance over the years says something about how Linux and open source has been viewed by outsiders and how it's finally being represented as a technology core to their use and development. This is partly because many people are now aware of what Linux does and what it is, but it's also because Linux and open source has for a long time been used in film production itself, from 3D modelling and the complex calculations used to generate the scenes and animation, to the user-interfaces behind the custom software for studios that would rather not mention it.

Fame at last

There are lots of brief appearances on screen though. It's seen in 2010's film about the creation of Facebook, *The Social Network*, for example, and even features in the official trailer (skip to 1:04). The character based on Mark Zuckerberg uses KDE 3 pimped out with the Keramic theme to browse the internet. And Linus Torvalds was rumoured to have bumped up the kernel version to 4 in preparation for meeting the exact version number, 4.1.15-1.1381_SKYN12nnmp, spotted running on the T-800 in the film *Terminator: Salvation*. Plus there's been plenty of mentions on geek-centric TV, in particular "They even had Gnome 2.0 the day Warty Warthog came out." on *Veronica Mars,* and Sheldon Cooper in *The Big Bang Theory*: "Oh Ubuntu, you are my favourite Linux-based operating system."

We're going to take a look at some of our favourite Linux-appearances, but to set the scene, we're going to choose a film where it doesn't. Before Linux could become usable, the operating system it was based upon – UNIX – was already being associated with the geeky side of computers, preparing the stage for a time when Linux itself could make an entry. And its most mention has become both a meme and a subreddit, and it comes from the biggest film of 1993.

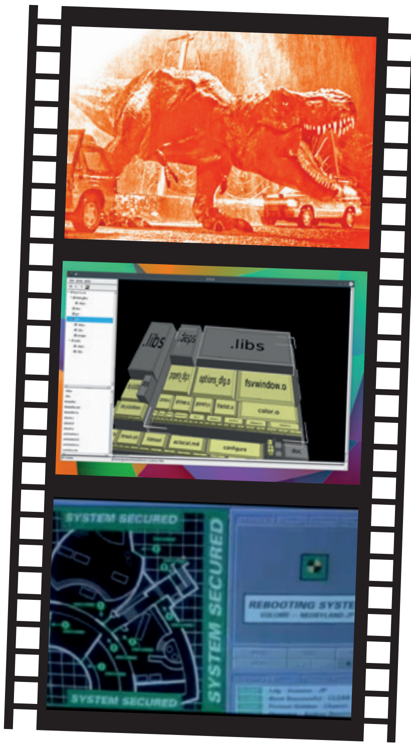
Jurassic Park (1993)

Surely everyone knows the plot to this blockbuster, based on a novel by Michael Crichton. The film itself is set on a fictional island called Isla Nublar where a bioengineering company has created a unique theme park. Instead of roller coasters and over salted fries next to expensive drink stands, this theme park is inhabited by a variety of genetically cloned dinosaurs built from DNA extracted from ancient mosquitoes caught in amber. What could possibly go wrong? Of course, it all goes wrong and the dinosaurs begin to rampage, spoiling the trip for a number of actors.

The clip we're after appears towards the end of the film, where one of the young people in the surviving group recognises the computer sitting in the room by saying, "It's a UNIX system, I know it." She's then able to awkwardly use the mouse to navigate a 3D map of the filesystem, switching to a Motif-like GUI that displays the local area in 2D and allows the locks to be activated, saving the group moments before an extra wearing claw gloves can get into the room. The machine itself is a Silicon Graphics Crimson, and befitting a system with the best graphics of the time, the 3D rendering on the screen has that lovely rendering quality typical in the early 90s. What's unusual is that while it's easy to scoff at this 3D representation of a filesystem and accessing locations in this way, the tool they're using was real and contemporary to the SGI machine being used in the movie.

The velociraptors are coming!

The tool was called *File System Navigator*, abbreviated to *fsn*, and was developed by Silicon Graphics as a proof of concept. What's particularly awesome is that there was a port of this to Linux and an updated version of that port that still compiles today. It's called *3D File System Visualizer*, or *fsv*, and the latest version is *fsv2* (see **http://fedorchenko.net/fsv2.php**). We needed to build the executable from the source code and because of its age, we needed to add **-lGLU** to the next but last build step, but otherwise, it's still fully functional. You can't quite lock the doors with it, but *fsv2* still does a good job of showing you which folders have the most files and presenting your filesystem in a nicely drawn OpenGL window.



A version of the 3D filesystem viewer used in *Jurassic Park* can be built and installed on a modern Linux system.

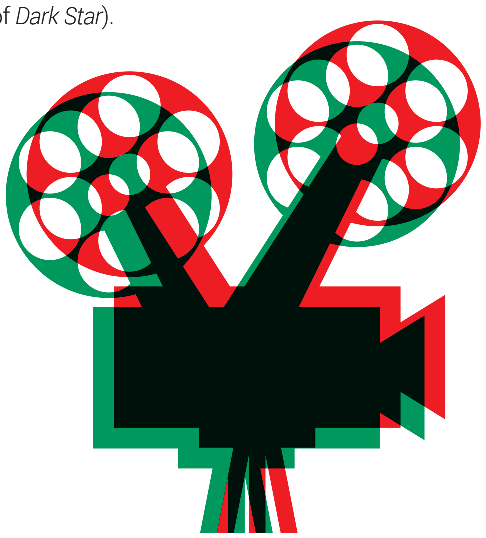
Perhaps the most remarkable thing about *Antitrust* is that it's directed by Peter Howitt who mid-life UK readers will know as Joey Boswell in 80s sitcom, *Bread*.

Antitrust (2001)

Antitrust is a big-budget film with a ridiculous script, hammed-up acting and a plot that combines pre-dot-com bubble startup culture with a Microsoft-gone-bad alike corporation and a megalomaniac CEO.

Unlike the majority of films, where Linux is but a brief screenshot and a prop for a hacker's credentials, however, the story manages to portray some understanding of what open source is about, even squeezing in a few cameos from prominent open source figures of the timesuch as Miguel de Icaza (next to Sun's Scott McNealy no less). Open source was also talked up as part of the pre-release marketing, perhaps in an attempt to generate interest within the Linux communities.

The film also has plenty of Linux screen action, and as you might expect from a film showing Miguel set in a period where the Eazel company was raising millions of dollars to help develop Gnome's file manager, *Nautilus*, this screen action mostly shows Gnome. What's great about the depiction in *Antitrust* though isn't that it's being used to lend credibility, but because it's actually what the people depicted in the movie would use to work on the ideas they worked on. It's the desktop those people actually used, and seeing those scenes now is quite nostalgic if you happen to remember the Gnome of the day. Also, in our opinion, the film has its faults but it's also quite good fun to watch, especially after all this time. Microsoft is no longer using the same strategies it did 15 years ago and the world of big data and governmental snooping makes our old worries about corporate monopoly seem rather quaint. (Also, Mr Veitch, if you're reading this. I've still got the copy you lent me almost 10 years ago along with your copy of *Dark Star*).



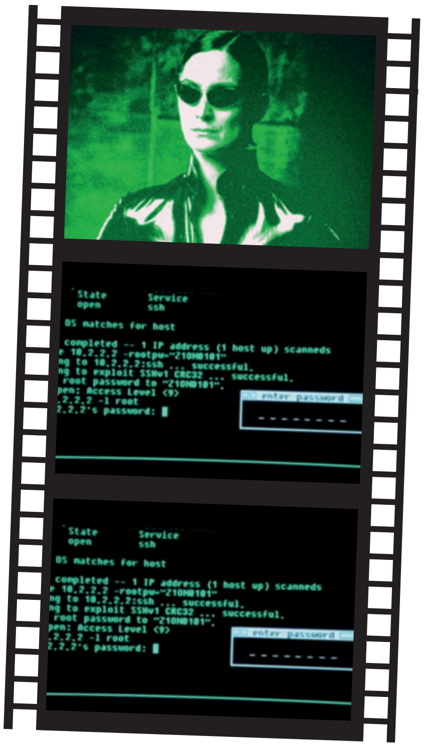
The Matrix Reloaded (2003)

Like many of the films here, *The Matrix* is likely to need little introduction. A total of three films were made, and the impact the first had in both its visual style (the ultra-slow bullet time), and its mind bending narrative has been seen in countless films since. Written and directed by the Wachowski Brothers, the first film eventually became part of a trilogy that explored our influence over the nature of reality while a war rages between the tiny percentage of humans who escape slavery and their machine-like controllers.

"What we see has all the hallmarks of Linux with a tiling or minimal window manager."

Linux makes an entry around two thirds of the way through the second film in the trilogy, *The Matrix Reloaded*. Trinity, played by Carrie-Anne Moss, needs to break into a supercomputer, and she does so by using an old-school green phosphor terminal. The display you see on the film shows the results of a command you don't see her type, but the output is what you'd get from the network portscanner *nmap*. This output includes the discovery that the server she wants to access is running SSH on port 22, the default port for SSH. Trinity then runs a fictitious exploit called 'sshnuke' against the server, presumably setting the root password 'Z10N0101' as this is the value she passes in the argument. This exploit then outputs that it's successful taking advantage of 'SSHv1 CRC32' before enabling Trinity to issue a normal SSH command to access the root account of the server.

Nearly everything covered in this brief sequence is very close to being realistic. And while the operating system could feasibly be something like Mac OS X, which was released a year or two earlier, what we see has all the hallmarks of Linux with a tiling or minimal window manager. The display could even have been constructed using a text interface such as *curses* on the command line, for instance.



```
 State        Service
 open         ssh

OS matches for host

 completed -- 1 IP address (1 host up) scanneds
e 10.2.2.2 -rootpw="Z10N0101"
ng to 10.2.2.2:ssh ... successful.
ng to exploit SSHv1 CRC32 ... successful.
 root password to "Z10N0101".
pen: Access Level <9>
.2.2.2 -l root
2.2.2's password: █
```

```
 State        Service
 open         ssh

OS matches for host

 completed -- 1 IP address (1 host up) scanneds
e 10.2.2.2 -rootpw="Z10N0101"
ng to 10.2.2.2:ssh ... successful.
ng to exploit SSHv1 CRC32 ... successful.
 root password to "Z10N0101".
pen: Access Level <9>
.2.2.2 -l root
2.2.2's password: █
```

Could the password, Zion, followed by the binary value for 5 be a coded message?

Tron: Legacy (2010)

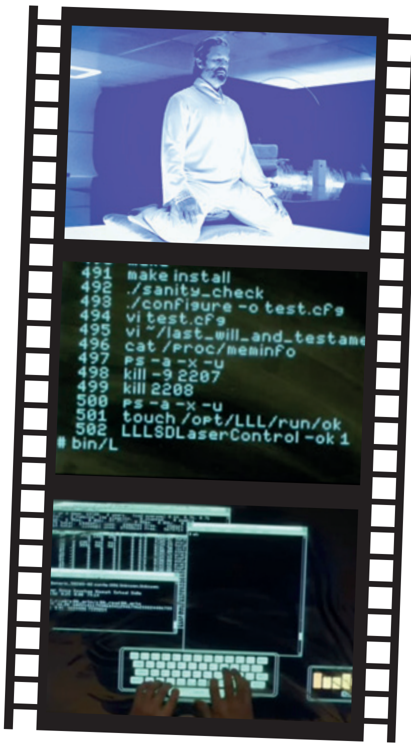
The first *Tron* film was made in 1982 and with it, Disney become one of the first studios to successfully blend live action with computer-generated graphics. The scene in which Jeff Bridges races his Light Cycle across the neon vector wasteland is burned into our collective memories. *Tron* also had a compelling plot, involving a malevolent computer system going out of control and a brilliant soundtrack by synth pioneer Wendy Carlos. The film's slow burning success led to an eventual sequel in 2010, *Tron: Legacy*.

The sequel isn't as good, but it does feature our favourite operating system, in a clip where Sam, the son of the protagonist in the the original film, uncovers his father's large touchscreen computer. He removes dust from the screen with his hand to reveal a relatively monochrome X11 environment. In the film, there are a few windows open, including a terminal showing a process list thanks to the **top** command, and another terminal that the protagonist types in using the on-screen keyboard. His interaction with the terminal is relatively realistic, first typing **whoami** to return the name of his father, and **uname -a** to return the kernel and operating system being used. For the film, this is SolarOS 4.0.1, which probably a reference to Solaris, hinted at with the sun4m kernel running surprisingly in 32-bits (i386), also seen in the output from **iostat** running in another terminal.

"The sequel to Tron isn't as good as the original, but it does feature our favourite OS."

Sam then uses some neat *Bash* skills by typing **bin/history**. This isn't perfect, but the output is the slowed-down output of the real **history** command, which will list all your previously logged commands. Those displayed on screen include **vi ~/last_will_and_testament.txt**, **ps -a -x -u** for listing processes and the utterly feasible **kill -9 2207**. Sam then manually types the output from the final command output in the history, the implication being this was the last thing his father typed on the computer.

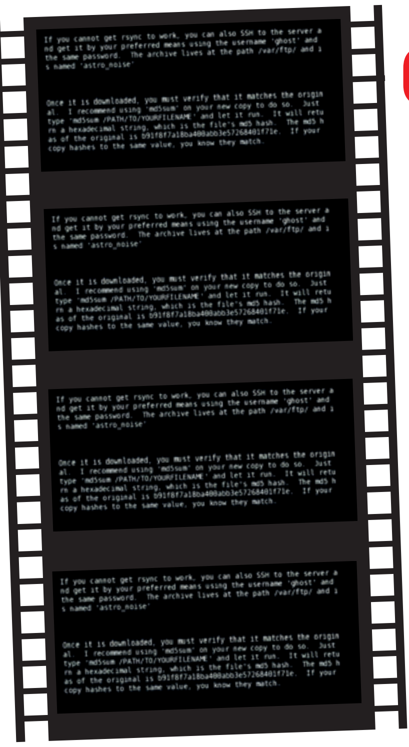
The only thing we could have improved from a Linux point of view – and this would admittedly look less cool on screen – is that Sam could have done the same thing much more quickly by typing **!** followed by the number of the command he needed from the history output.



```
491  make install
492  ./sanity_check
493  ./configure -o test.cfg
494  vi test.cfg
495  vi ~/last_will_and_testame
496  cat /proc/meminfo
497  ps -a -x -u
498  kill -9 2207
499  kill 2208
500  ps -a -x -u
501  touch /opt/LLL/run/ok
502  LLLSDLaserControl -ok 1
# bin/L
```

This may not have been a real theme at the time the movie was made, but you can now get the *Tron Legacy* look on both Gnome and KDE.

Citizenfour (2014)



```
If you cannot get rsync to work, you can also SSH to the server a
nd get it by your preferred means using the username 'ghost' and
the same password.  The archive lives at the path /var/ftp/ and i
s named 'astro_noise'


Once it is downloaded, you must verify that it matches the origin
al.  I recommend using 'md5sum' on your new copy to do so.  Just
type 'md5sum /PATH/TO/YOURFILENAME' and let it run.  It will retu
rn a hexadecimal string, which is the file's md5 hash.  The md5 h
as of the original is b91f8f7a18ba408abb3e57268401f71e.  If your
copy hashes to the same value, you know they match.
```

```
If you cannot get rsync to work, you can also SSH to the server a
nd get it by your preferred means using the username 'ghost' and
the same password.  The archive lives at the path /var/ftp/ and i
s named 'astro_noise'


Once it is downloaded, you must verify that it matches the origin
al.  I recommend using 'md5sum' on your new copy to do so.  Just
type 'md5sum /PATH/TO/YOURFILENAME' and let it run.  It will retu
rn a hexadecimal string, which is the file's md5 hash.  The md5 h
as of the original is b91f8f7a18ba408abb3e57268401f71e.  If your
copy hashes to the same value, you know they match.
```

```
If you cannot get rsync to work, you can also SSH to the server a
nd get it by your preferred means using the username 'ghost' and
the same password.  The archive lives at the path /var/ftp/ and i
s named 'astro_noise'


Once it is downloaded, you must verify that it matches the origin
al.  I recommend using 'md5sum' on your new copy to do so.  Just
type 'md5sum /PATH/TO/YOURFILENAME' and let it run.  It will retu
rn a hexadecimal string, which is the file's md5 hash.  The md5 h
as of the original is b91f8f7a18ba408abb3e57268401f71e.  If your
copy hashes to the same value, you know they match.
```

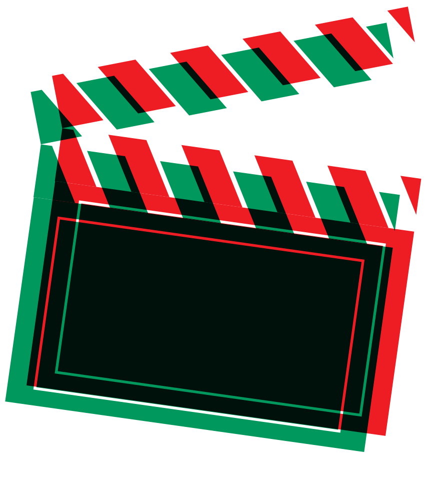
```
If you cannot get rsync to work, you can also SSH to the server a
nd get it by your preferred means using the username 'ghost' and
the same password.  The archive lives at the path /var/ftp/ and i
s named 'astro_noise'


Once it is downloaded, you must verify that it matches the origin
al.  I recommend using 'md5sum' on your new copy to do so.  Just
type 'md5sum /PATH/TO/YOURFILENAME' and let it run.  It will retu
rn a hexadecimal string, which is the file's md5 hash.  The md5 h
as of the original is b91f8f7a18ba408abb3e57268401f71e.  If your
copy hashes to the same value, you know they match.
```

While the other films are primarily entertainment, *Citizenfour* uses Linux to make a very serious point.

Citizenfour is a vital documentary that follows Edward Snowden in the days up to and immediately following his whistleblowing on the activities of the NSA and the extent of its global surveillance. It's filmed in Hong Kong before he fled to Russia and before he knew what the world's response would be to what he reveals.

The big difference in *Citizenfour*, of course, is that while nearly every other film uses Linux, UNIX or the command line to imbibe the viewer with a sense of computing literacy, in *Citizenfour* Ed Snowden is simply using Linux and open source for real. It's not a set up. It's not there to add pseudo credibility. It's there because it's the best tool for the job. He uses it for *GPG* email encryption when contacting the film's director, Laura Poitras, and he uses the Tails distribution to connect to the *Tor* network. There's also plenty of *Rsync* action on screen – used to transfer an archive securely, and SSH is mentioned as an alternative method in an email. We even see how difficult it is trying to tutor Guardian journalist, Glen Grenwald, into how to use all this technology.

Linux and open source in *Citizenfour* is the only trusted method for sending this information, and for that reason, this is the best film with which leave the story arc of this feature. From the fiction of a dinosaur theme park to the reality of governmental data collection and spying, the portrayal of UNIX/Linux has come a long way. And now that open source has effectively won in the world of development, mobile and servers, it much more likely to be realistically portrayed in the future. LV



SWITCH TO LINUX

New to Linux? Or want to convert your friends and colleagues? Our guide has everything you need to know.

Linux Voice is unashamedly a geeky magazine. We don't shy away from advanced topics such as assembly programming and kernel hacking. But we were all beginners once, so this month we've decided to help newbies get into Linux with a special guide. If you've never used Linux before but want to dip your toes into its glistening waters, we'll get you started over the next six pages. Or if you're a regular Linux Voice reader who already knows his or her way around the operating system, cut out this guide and give it to friends, family and colleagues – or make photocopies and convert everyone you know!

Before we get started, though, what exactly is Linux? Where did it come from? Well, Linux is an operating system, much like Windows and Mac OS X. It runs on your computer, acting as a middleman between your hardware and your applications. It manages your computer's memory, helps different programs to run together, and has drivers for your hardware.

Linux has many strengths, such as security and performance, as we'll see in a moment. It runs its own software, although it can also run a selection of Windows programs. Linux is great as a secure and reliable desktop operating system, but it also powers the internet: Google runs Linux on tens of thousands of servers, for instance. And you might not know it, but Linux forms the basis of Android, the operating system that totally dominates smartphones and tablets.

What we call "Linux" today is the work of multiple projects that have been running since the 1980s, all of which have worked together to create a free, open and shareable computing platform. The GNU project played a huge role in this, which is why you sometimes see Linux referred to as GNU/Linux, and today the operating system has hundreds of thousands of developers around the globe.

"Linux is a secure, reliable desktop operating system, and it also powers the internet."

Why use Linux?

The four big reasons why you should make the switch.

1 It's free

Linux is free (as in zero-cost). That's right, you don't have to pay a penny to use it. But how on earth is such a large body of software completely free – who pays for its development?

Much of the work on Linux is done by volunteers around the world, working over the internet. But an increasing number of contributions come from large companies such as IBM, Intel, Red Hat and Canonical. They don't make money from selling the operating system itself, but they generate revenue by offering support contracts, services, documentation etc.

So if you just want to use Linux on your home computers, you can download and use it for free. If you want to deploy it across 5,000 PCs in an enterprise, and need someone on the end of the phone who will fix any potential problems, you can pay IBM, Red Hat or other companies to provide support. Many volunteer Linux developers and projects also raise money via donations or selling merchandise such as T-shirts and mugs.

2 It's open and secure

This is hugely important. Linux is open source, which means anyone can study its inner workings. You can download the source code (the original human-readable recipe) of Linux, change it, and recompile it to run on your computer. Now, few people have the technical nous to do this, but it's essential nonetheless: you have full control over your computer.

With Windows, Mac OS X or iOS, you can never be sure what the software is doing – you can't get the source code, and you can't fix it yourself. It's like buying a car with the bonnet welded shut.

With Linux, anyone can look inside and improve it. The knowledge encapsulated in its millions of lines of source code is there for the whole world to benefit from. And because its code is in the open, it's almost impossible for government agencies or other nefarious types to sneak in back-doors or other ways to monitor you. With many thousands of people studying new code as it's added to Linux, security holes are usually spotted very quickly. Who knows what's lurking inside Windows and Mac OS? You can't find out. In short: with Windows and Mac OS X, someone else is in charge of your computer. With Linux, you have total control.

3 It's reliable

Linux is well known for its reliability and general crash-proofness. Its overall design is based on that of Unix, a family of operating systems that goes back to the 1970s, so it's built on mature and well-established foundations. With developers all over the world working on its codebase, bugs are found very quickly, and because it's open source, anyone can fix a bug. Even if you're not a programmer yourself, you can pay someone to fix an issue or add a feature you need. With Microsoft? Good luck, unless you have tens of thousands of pounds to wave around...

A properly set up Linux system simply won't crash unless something is wrong with your hardware. We know people who've been running Linux servers for several years without a single reboot. Linux is designed in such a way that its various components are well isolated from one another, so if there's an issue with one part of the operating system (such as the graphical user interface), the rest of it carries on chugging away.

4 It's compatible

Linux may be a different operating system to Windows and Mac OS, and doesn't run all of the same programs, but it's the most compatible OS in existence. You can open your *Microsoft Office* documents in *LibreOffice*, you can play all your videos and music in the *VLC* media player, and there are Linux equivalents for pretty much every application in the Windows and Mac OS worlds – we'll explore the best software later on. Whereas paid-for software often tries to lock you in to closed file formats, Linux applications respect that we all like freedom of choice. And Linux happily co-exists with Windows or Mac OS X, so you can have both on your computer and choose when you power it on.

While many Linux developers are volunteers, the system is also backed by some of the biggest companies in IT.

Which distribution?

Linux comes in many flavours – here are the biggest names.

Because the Linux source code is free and open for everyone to share, anyone can also package it together and make their own "distribution" (aka "distro"), which is an installable version of the operating system. There are hundreds of distributions out there, some made by big companies and some made by small groups of volunteers, but they're all Linux at the core and compatible with each other. Some distributions are geared towards new users, others for servers and software development workstations – everyone has their favourite. Here are the main ones you need to know about.

Ubuntu www.ubuntu.com

This is by far the best-known distribution, and is a great all-round operating system. Ubuntu is primarily geared towards desktops and laptops, although it's making gains on tablets and phones as well. With Ubuntu, you can get a modern, shiny and well-tested version of Linux on your PC within just a few mouse clicks.

Fedora www.getfedora.org

This is from Red Hat, the makers of Red Hat Enterprise Linux, a distribution focused on servers and business usage. Fedora is a community-supported distribution known for incorporating cutting-edge technologies, and makes new releases every six months. Like Ubuntu, Fedora focuses on having an attractive and versatile interface.

OpenSUSE www.opensuse.org

Developed in Germany, OpenSUSE is one of the longest-running Linux distributions, having started life in the mid 90s. It's popular among intermediate Linux users, sporting an excellent configuration tool called *Yast* that lets you tweak all aspects of your system from within a single program. Great for control freaks.

Debian www.debian.org

Debian is released rather slowly (once every two years), but is well known for its stability and is therefore used on tens of millions of servers around the world. Debian is compatible with many kinds of computer hardware, and provides the basis for many other distributions, such as Ubuntu.

Arch Linux www.archlinux.org

Arch can be complicated to install but it teaches you a huge amount about how Linux works under the hood. Arch Linux is a rolling release distro, so instead of having big updates every six months like most other distros, it's constantly changing with the latest software.

Linux Mint www.linuxmint.com

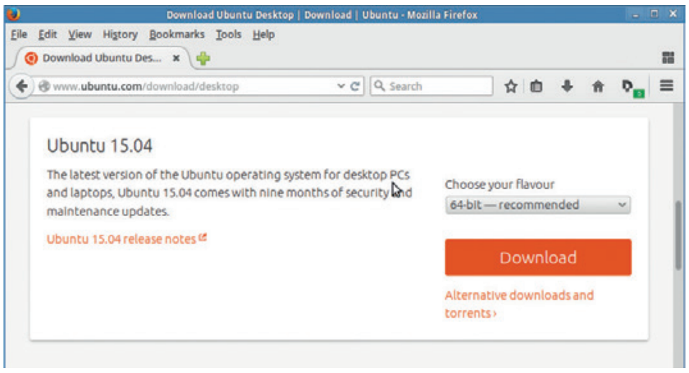
Mint is based on Ubuntu, but provides a different interface and set of default software. It's popular among new users and has a very helpful, supportive community. Various versions of Mint exist with different interfaces – the most popular at the moment is the traditional *Mate* version.

So which one should I choose?

We'll make this simple: Ubuntu. You'll see people recommending other distributions as well, but Ubuntu is the best known, is very polished, and has a huge supporting community on the web (eg **www.askubuntu.com**). The Ubuntu team puts a lot of effort into its interface and makes sure that the operating system works well out of the box, so we think it's the best way to start. After a few months with Ubuntu, you'll be confident enough in Linux to try other distributions and expand your horizons.

Install Linux! Follow our step-by-step guide and get Ubuntu onto your PC.

Requirements: 1GHz Intel/AMD CPU, 2GB RAM, 10GB drive space

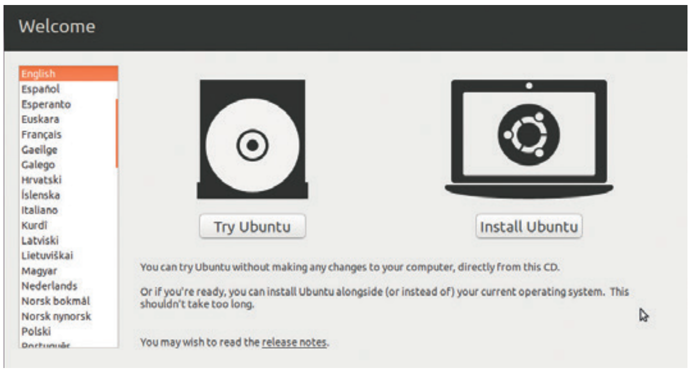

Download Ubuntu Desktop | Download | Ubuntu - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Download Ubuntu Des...
www.ubuntu.com/download/desktop
Search
Download
Ubuntu 15.04 release notes
Alternative downloads and torrents ›

Ubuntu 15.04

The latest version of the Ubuntu operating system for desktop PCs and laptops, Ubuntu 15.04 comes with nine months of security and maintenance updates.

Choose your flavour: 64-bit — recommended

1 Download
Go to **www.ubuntu.com/download/desktop** and get the latest version (15.04 at the time of writing). You'll download a **.iso** file, which is a disc image that can be burned to a DVD-R using your regular disc burning software. If you want to use a USB key to install Linux, follow the instructions at **http://tinyurl.com/ubuntukey**.

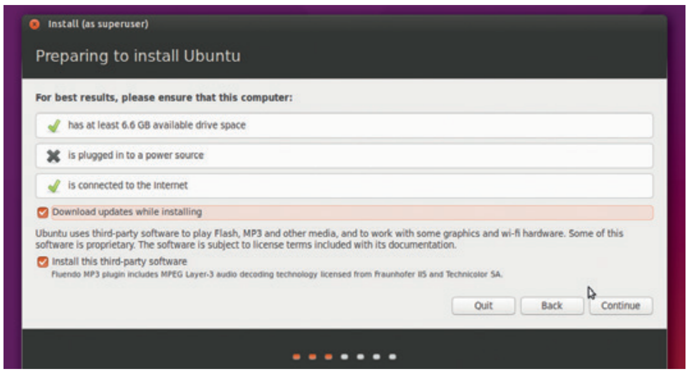


You can try Ubuntu without making any changes to your computer, directly from this CD.

Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.

You may wish to read the release notes.

2 Boot
Boot your PC from the DVD-R or USB key; you normally need to press a key on your keyboard when your computer starts to do this, so consult your PC's documentation to find out how. After a few moments, Linux will run from the DVD and this screen will appear – click Install Ubuntu to begin the installation process.



For best results, please ensure that this computer:

- ✓ has at least 6.6 GB available drive space
- ✗ is plugged in to a power source
- ✓ is connected to the Internet

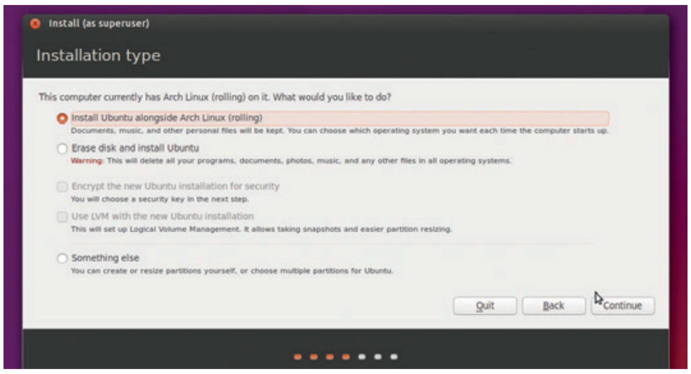
[✓] Download updates while installing

Ubuntu uses third-party software to play Flash, MP3 and other media, and to work with some graphics and wi-fi hardware. Some of this software is proprietary. The software is subject to license terms included with its documentation.

[✓] Install this third-party software

Fluendo MP3 plugin includes MPEG Layer-3 audio decoding technology licensed from Fraunhofer IIS and Technicolor SA.

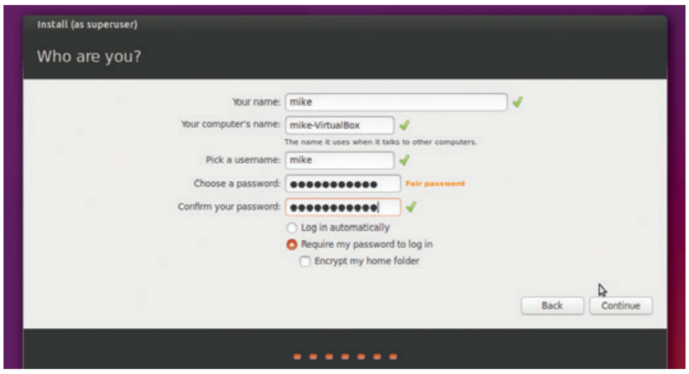
3 Settings
The Ubuntu installer will check that your machine has sufficient hard drive space to install Linux. If you're connected to the internet, you can download updates and extra drivers and media file codecs (recommended) during installation; click on the network icon in the top-right to set up a Wi-Fi connection if necessary.



This computer currently has Arch Linux (rolling) on it. What would you like to do?

- [✓ Install Ubuntu alongside Arch Linux (rolling)]
  Documents, music, and other personal files will be kept. You can choose which operating system you want each time the computer starts up.
- Erase disk and install Ubuntu
  Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.
- Encrypt the new Ubuntu installation for security
  You will choose a security key in the next step.
- Use LVM with the new Ubuntu installation
  This will set up Logical Volume Management. It allows taking snapshots and easier partition resizing.
- Something else
  You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.

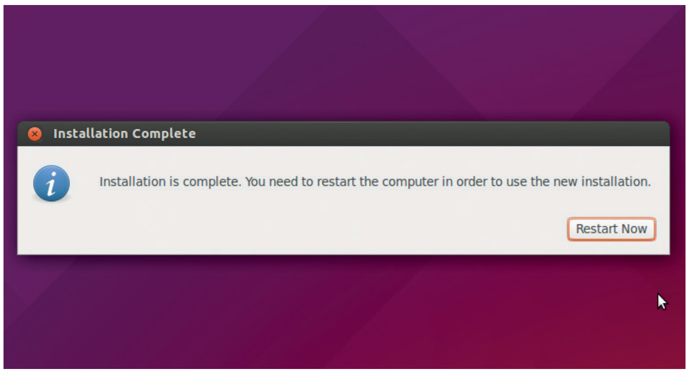
4 Partition
Now choose where to install Linux on your PC's hard drive. You can install it alongside Windows, and have a menu when you start your PC to choose your operating system, or you can dedicate the whole hard drive to it. Choose "Something else" if you're experienced with partitioning and want full control of where the systems end up.



Your name:	mike ✓
Your computer's name:	mike-VirtualBox ✓
	The name it uses when it talks to other computers.
Pick a username:	mike ✓
Choose a password:	●●●●●●●●●●●● Fair password
Confirm your password:	●●●●●●●●●●●● ✓

- Log in automatically
- [✓ Require my password to log in]
  - Encrypt my home folder

5 User account
Now the Linux files will be copied to your hard drive, and you'll be asked to set your location and keyboard layout. You will also be prompted to set up a user account so that you can identify yourself to the operating system and log in – don't forget your password! You can also choose to encrypt your personal files here.



Installation is complete. You need to restart the computer in order to use the new installation.

6 And you're done!
Once all the files have been copied over (depending on your PC, this can take a few minutes), the installer will prompt you to reboot the machine, so click on Restart Now and remove the DVD or USB key once the PC restarts. Then you can choose Ubuntu from the boot menu that appears, and turn over the page to start exploring.

Explore Linux

Discover your new operating system and its included software.

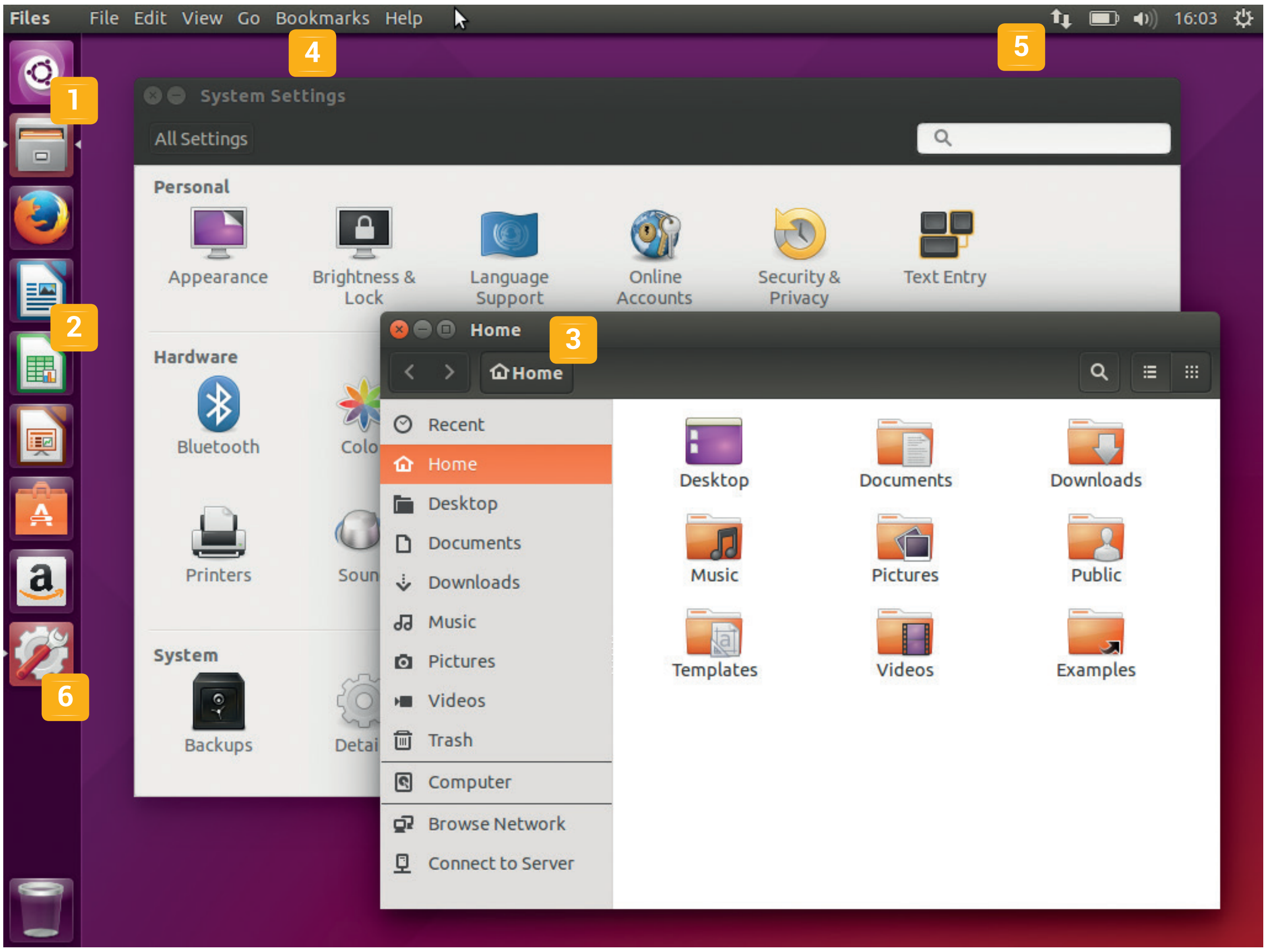
Congratulations – you now have Linux installed! Log in with the username and password that you specified during the installation, and the Unity desktop will appear. You'll notice that it looks quite different to Windows and Mac OS X – but it's also very easy to pick up.

To access your personal files, click the drawer button underneath the Ubuntu button on the left-hand panel. Your "home" directory is like My Documents in Windows – it's where your personal files are stored. If you insert a DVD or plug in a USB key, a window will pop up showing its contents, and on the left-hand panel of the file manager, you can also access resources on the network.

Underneath the drawer button you'll see an icon for *Firefox*, a web browser you're probably familiar with from Windows or Mac OS X. *Firefox* is arguably the best browser out there, combining good performance and thousands of extensions with excellent privacy settings. And underneath *Firefox* you'll see three icons for *LibreOffice*, opening the word processor, spreadsheet and presentation tool respectively.

LibreOffice is the flagship office suite on Linux, and is tremendously capable, having seen decades of development in its

Exploring the Ubuntu desktop



1 **Ubuntu button** This is similar to the Start button in Windows. Click on it to browse included software (go to the Applications button at the bottom of the window after clicking it, and then Installed to see what's included by default). You can also type to search and run programs.

2 **Applications** These buttons are shortcuts to useful programs. When you start a new program, its icon will appear on this bar; right-click it and choose "Lock to Launcher" to keep it there after closing the app.

3 **Windows** Click and drag the title bars to move them, and use the edges to resize them. The red button on the top bar closes windows, while the other buttons minimise and maximise respectively.

4 **Menu bar** Ubuntu has a global menu bar, like in Mac OS X; when using an application, move your mouse pointer to the top bar to show menu entries.

5 **System tray** This is where you'll find icons for audio levels, power management and networking. Click on the cog icon on the far right to log out or shut down the machine.

6 **Settings** Click this cog-and-spanner icon to open up the Systems Settings window, from which you can configure your installation, manage your hardware, and add new user accounts, if more than one person will be using the PC.

previous incarnations as *OpenOffice* and *StarOffice*. *LibreOffice* does a great job of opening *Microsoft Office* documents – although there can be slight formatting issues with some very complicated documents. Still, if you open an *Office* doc from one version of the suite in a different version, you'll likely experience the same thing, so this is something even Microsoft gets wrong!

For email, click on the Ubuntu button and search for *Thunderbird*. This is an email client from the makers of *Firefox*, and is mature and very stable. Other pre-installed software worth exploring is *Rhythmbox* (a music player), *Empathy* (for instant messaging) and *Shotwell* (a photo manager). Of course, you'll find plenty of small tools such as a calculator and text editor as well.

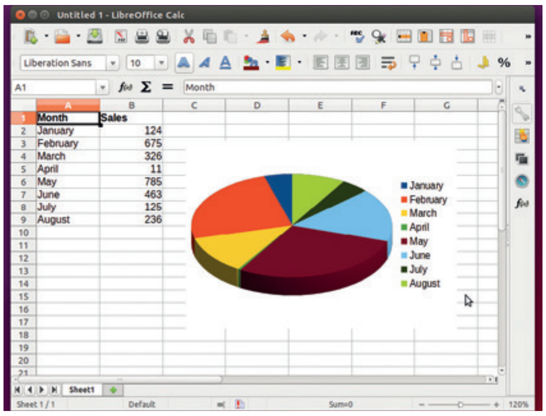
Command line basics

While it's perfectly possible to use Linux without the command line, many advanced features and shortcuts are accessible through it. So it's worth learning the basics, even if you spend 99% of your time doing things via the graphical user interface. In Ubuntu, you can open up a command line by clicking the Ubuntu button and searching for Terminal. Click this, and a window with a prompt will appear.

By default, this starts in your home directory (aka home folder), which as mentioned is akin to My Documents on Windows. This is in **/home/<username>**, where **<username>** is replaced by the name you gave during the installation. Enter **ls** (list files) to see files and directories (coloured in blue) in the current directory. To switch into a directory, use **cd**, eg **cd Downloads**. To switch back into the previous directory, use **cd ..** (note the two full stops).

To get a detailed list of files, use **ls -lh**, while to delete a file use **rm filename** (or **rm -r dirname** for a directory). You can copy files with **cp file1 file2**, and rename with **mv oldname newname**. To see the disk space usage in a directory, enter **du -h**, while the contents of text files can be read with **less filename** (press Q to quit).

The Linux command line has many shortcuts to save you time. You can use the up and down cursor keys to cycle through previously entered commands,



Month	Sales
January	124
February	675
March	326
April	11
May	785
June	463
July	125
August	236

LibreOffice is compatible with Microsoft's suite, and has excellent word processing, spreadsheet and presentation tools.

for instance, while the Tab key auto-completes file or directory names. Say you have a directory called **MyPhotosFrom2007**, and you want to switch into it with **cd**. Instead of having to type the full directory name, enter **cd My** and hit Tab – it will auto-complete the directory so you can just hit Enter.

In guides and tutorials on the web (and in this magazine), you'll often see commands beginning with **sudo**. This performs the command as the "root" user, which is the administrator, so it should only be used for commands that make important system changes. After hitting Enter, you will be prompted to enter your password before the command is executed; in this way, random pieces of software can't change your operating system without your permission.

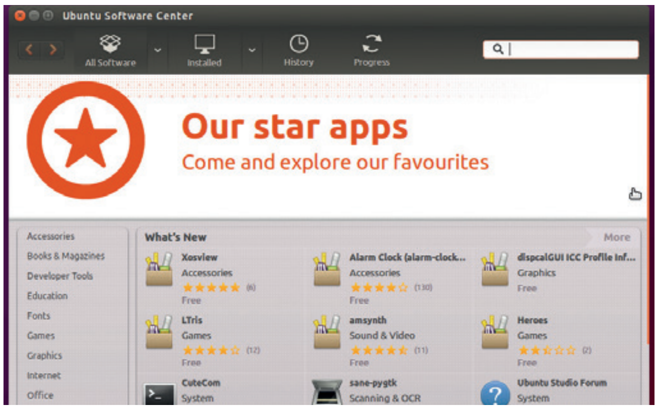
Where to go from here?

So, you have Linux installed, you've explored the desktop and supplied software, and you've learned the command line basics. Good work: you're now a Linux user! From here you'll want to add more software (see the boxout) and become more proficient with your Linux skills. Of course, we recommend a subscription to Linux Voice for this, because every subscription includes access to all back issues in DRM-free digital (PDF and ePub) formats. So from just £38 you get a mammoth compilation of over 1,500 pages of Linux tutorials and features – see **http://shop.linuxvoice.com**. And if you need any help, pop by our forums at **http://forums.linuxvoice.com**. Happy Linuxing! LV

Adding more software

Ubuntu comes pre-installed with many top-class applications, but thousands more are available too. Click on the Ubuntu button, type "software" and choose the *Ubuntu Software Centre* to explore programs available to download – most of them free and open source. You can browse categories down the left, and explore desktop productivity programs, multimedia apps, games, software development tools and much more.

Some of our recommendations include *Gimp* (an image editor), *Audacity* (for editing audio files), *OpenShot* (a movie maker), *VLC* (a media player that handles virtually every format under the sun) and *HomeBank* (personal finance). If you're looking for a specific type of program but can't find something suitable, ask fellow Linux Voice readers on our forums at **http://forums.linuxvoice.com!** LV



The *Ubuntu Software Centre* provides access to thousands of downloadable programs with just a few clicks.

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

FAQ Arduino

Discover one of the world's simplest single-board computers.

BEN EVERARD

Q Ard-what? how do you even pronounce that?

A It's Ard-WEE-no. The project got its name from the bar where the founders used to meet, and the bar got its name from Arduino, Margrave of Ivrea and King of Italy (from 1002 to 1014 AD).

Q Right, what is this Ard-WEE-no thing then?

A Arduino is a project to make microcontroller boards, to make physical computing more accessible.

Q That answer just confused me more! What's a microcontroller, and what's physical computing?

A A microcontroller is a chip that contains a processor along with some flash memory and RAM. Typically, the processors in microcontrollers are far simpler than the sort of CPUs that you find in regular computers, so don't have all the features necessary to run a fully fledged OS like Linux. The normal way of using them is to compile code on a separate computer and copy it directly into the flash memory. When the Microcontroller is powered on, it just executes the single program stored in this memory.

Physical computing is computing that interacts with the real world in some way. For example, a physical computing system might use sensors to measure the environment, and then respond by lighting up different things depending on what readings it received.

The Arduino project has created a series of small boards that contain a microcontroller, and a large number of connectors to which you can add sensors. Programs can be written in the Arduino IDE on another computer and uploaded to the Arduinos via a USB cable. Typically, Arduino programs run without prompting by the user and without a keyboard or mouse (though sometimes buttons or joysticks are added to allow some user input).

"The Arduino's headers enable you to add extra hardware with no electrical knowledge at all."

Q That sounds cool. What are these Arduino boards like?

A The most popular Arduino board is the Arduino Uno Revision 3. This is small enough to fit in the palm of your hand. It's got 14 programmable digital input/output pins and 6 analogue input pins. It's all based on an ATMega 328 8-bit microcontroller which has 32K flash memory, 2KB of RAM and runs at 16MHz. These specs may sound appalling by the standards of modern computers, but remember that its main function is to monitor sensors and control simple hardware. It's more than powerful enough for most projects.

As well as the Uno, there are 20 other official boards that all work in the same way, but are different sizes and have different configurations of pins, and different power processors. For example, there's the Arduino Nano, which is much smaller than the Uno but has fewer inputs and outputs; the Arduino Due, which is larger and has more input and outputs a more powerful ARM processor; and the Arduino Lilypad which is designed to be sewn into clothing.

Q I like the idea of physical computing, but I don't know the first thing about electronics. Is there an easy way to get started?

A The headers on the Arduino allow specially designed add-on boards called shields to be plugged in. These enable you to add extra hardware to your Arduino with no electrical knowledge at all. There are loads of these shields from a wide range of manufacturers. They can be as simple as a few buttons to add input, or as complex as modems that interact with a mobile phone network. Perhaps the most common use of shields is to add environmental sensors, or output displays to your Arduinos.

Shields can be stacked, so it's possible to connect more than one to an Arduino at a time, but not all are compatible with each other when stacking as they may use the same pins to communicate.

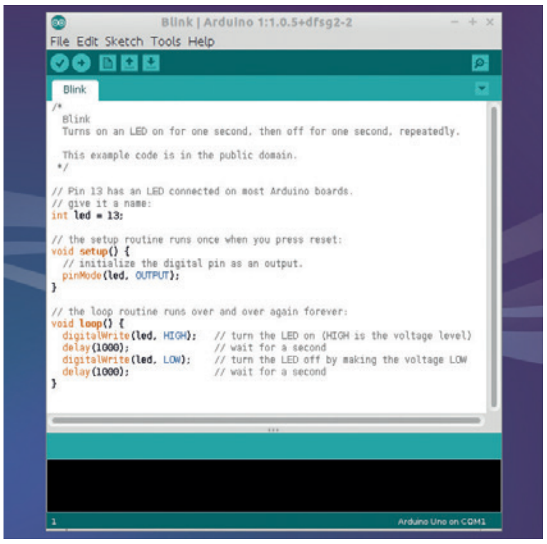
Q I already have a Pi, which has input and output pins. What are the advantages of an Arduino?

A There are quite a few advantages, depending on what projects you're doing. The Raspberry Pi runs Linux, which is a general-purpose OS, and has a whole heap of software running at once, and a kernel that allots processor time to different pieces of software. This means that it can be hard, or even impossible to perform precisely timed actions. On the other hand, and Arduino doesn't run an OS, so every CPU cycle is devoted to just one piece of software. If you need to switch a pin on or off with millisecond accuracy, the Arduino's microcontroller approach will do the job far better.

On the other hand, if you're doing any processing that requires high-level functions, particularly anything that needs significant processor power, or interaction using a monitor or keyboard, the Raspberry Pi will be a better option.

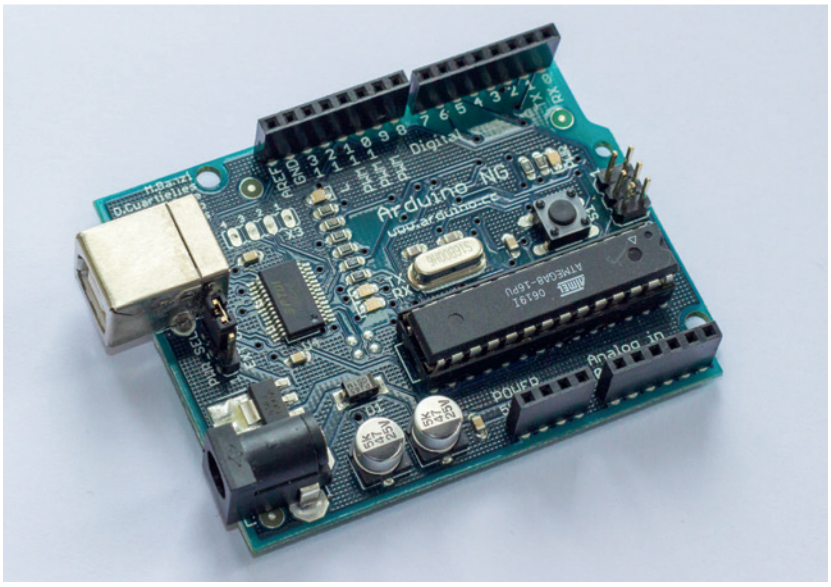
Q How do you program these Arduino boards?

A All the boards can be programmed using the Arduino IDE, which uses a dialect of C++ and runs on Linux, Mac OS X and Windows.



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

The Arduino IDE includes a wide range of programs to help you access the features of the hardware.



Most Arduino boards have rows of connectors to allow shields to push straight in.

All the boards are programmed in the same way. The IDE has a menu where you select the target hardware, and it compiles it appropriately. Most code will run on any of the boards unless it requires a hardware feature that's not present on some.

Q Is the Arduino IDE open source?

A All the Arduino software and hardware is open source. This means the IDE that's used to program the boards, the bootloader that enables uploading the software via USB, and the designs of the boards themselves.

Q The designs of the boards are open source? I thought open source was just about software.

A Although open source software is the most common embodiment of the open source principles, there's also a growing movement for open source hardware. To be open source, hardware has to release the schematics and design files under a licence that allows modification and redistribution. That way, just as people can modify FOSS, they can create new pieces of hardware based on the originals.

Q If it's open source, does that mean there are other boards based on the Arduino hardware?

A Yes, loads! There are lots of clones of the Arduino products that are often far cheaper than the originals (though they're usually lower quality as well). There are also many boards that modify the basic Arduino to include additional features, or fit into other form-factors.

Several of what are now official Arduino boards were designed by other organisations to fulfill niches that weren't well satisfied by existing microcontroller boards. Others have taken the Arduino hardware in entirely new directions. For example, there's the Udoo, which is a single-board computer that combines an ARM CPU running Linux with a microcontroller setup that's based on the Arduino Due. Together this gives you a computer that runs a Linux desktop, but also has a high level of control over the input and output pins.

Sometimes it feels like there's a new Arduino-based board coming out every day, and there are regular crowdfunding campaigns by people trying to launch new boards. They mostly have a project name ending in -duino.

Q The Arduino project sounds awesome. Is there a website I can go to and find out more?

A Ah! This is a slightly thorny question at the moment. There's a bit of a dispute between the founders of the project. The project's website has always been **www.arduino.cc**, but one founder has set up **www.arduino.org**, and is claiming to be the rightful heir to the project. For tradition's sake, we'll stick with **arduino.cc**, as this has the most active community. LV

DEBIAN PROJECT LEAD: NEIL MCGOVERN

Ben Everard and **Graham Morrison** head to the city of Cambridge in a big week for the world's biggest Linux distribution.

Debian is one of the oldest Linux distributions. It's also one of the most widely used and adapted, being at the core of more derivatives than our immediate memory cache can count. It's also one of the most democratic, and the project has just elected a new Project Leader to take over from previous incumbent Lucas Nussbaum, all within days of releasing Debian 8. It's the Project Leader's job to ensure Debian's community and engineers are listened to, and to provide a public face for the project, which is a huge challenge for a something as important as Debian. Which is why, when we found the new project leader was based in Cambridge – a mere three-hour drive away – we had to make the trip.

LV Can you tell us about what you do for a day job and how that relates to your work with Debian?

Neil McGovern: Yeah, sure absolutely. When I first thought about running, I first checked with the CEO and CTO of my employer, Collabora, who both started as free software developers themselves. At Collabora, one of the key things we try to do is accelerate the use of open source in the industry. So, not only using free software, but the methodologies behind it and trying to bridge the gap between this huge collection of free software and it being used in real life. So I'm in charge of all our engineers here.

LV That's a huge job in itself!

NMcG: It is, yeah. It has been useful and been described as herding cats a lot before. We try to hire the best open source people out there, so the bosses are more than happy with me doing work with Debian. It's quite important for Collabora as well, because lots of our customers end up using Debian or Debian-based distributions. So they're certainly very happy with it. We've got a really good policy of giving back to the community and trying to stay involved with the upstream policies rather than disappearing off somewhere.

LV Are you allowed to work on Debian as part of your daily 9 to 5 job?

NMcG: Yes, we've got a few schemes called community days where I take time to go and do it. Basically, our CEO said as long as it doesn't cause a problem, it's fine. If it does start to cause a problem, we'll just have a chat and work something out. At Collabora, we make sure it actually happens and carries on. It's really good.

LV Maybe it's early days, but it feels like, on the surface anyway, that there's quite a lot of overlap between what you do at Collabora, such as managing engineers, and what your job is and will be at Debian?

NMcG: The difference with Debian is that it's entirely voluntary driven, so you can't just tell people that they need to do something by a date or something similar. But a lot of what I do is trying to encourage our developers to grow and see what they want to do in the future, rather than just say 'that's your project, go on and do it'. My job is more to ensure that people get career development and things like that, so it works well.

As we're locally distributed both in Debian and Collabora, being able to talk to people online is one of the key things I have to do every day.

"Debian Project Leader is more of a figurehead role. You don't get to direct the way the project goes, but you can set your ideas out."

LV How do you do that? Is that through IRC?

NMcG: Yeah, lots of IRC. In Debian, there are lots of email lists. And just simple things like after someone's done quite a bit of work, just send them a quick thank you email can work wonders really.

At Debian, it's an interesting role, being project leader. It's more a figurehead role. You don't necessarily get to direct the way the project goes, but you can set your views out, where you think the project should go and just try to sort of speak to lots of other organisations and groups around Debian, and act as that liaison.

LV Would these other organisation be people that already work

“We’re over 45,000 in the latest release – bringing that all together is a lot of work.”

with and know Debian?
NMcG: So my handover involved a lot of speaking to the Free Software Foundation, the SFLC...

To say hello?
NMcG: To say hello and to work with them, because there are various things that we try to work together to try and increase the use of free software, and make sure that works really well.

Did Lucas Nussbaum have any pearls of wisdom for you when you took up his old job as DPL?
NMcG: I did get quite a good handover from Lucas, which is fairly unusual. And he’s still around, so I can still talk to him as well. Lots of the previous DPLs (Debian Project Leads) do talk to one other and share ideas. It’s quite a broad job; there are lots of demands on your time and it’s hard to fit it all in.

It sounds like one of those jobs that suck up any time that you have if you want it to.
NMcG: Yep, absolutely. Previously, when I was involved in quite a lot of politics, the one things that’s said when you ask “Oh, what about doing this”, and they’ll say “take as much time as you want or as little time as you want”, and it’s always a lie. It’s always a lot of time that you need to spend doing it, but it can certainly grow to a large amount of time if you need it, but at the moment it’s fairly manageable.

Can you tell us why you thought at this point you wanted to go for it?
NMcG: Yeah, so I’ve been involved with Debian for about 12–13 years or so, and I’ve done various bits and pieces, and for the last five years or so I was release manager. So I had – not Jessie, that’s just gone out – but the previous three releases I’ve been involved with. I finally stepped down from that as that’s also a considerable amount of work. It’s not the nicest job in the world to have to say no to people when they want their new package in and you’re in a freeze and you’re trying to stabilise the product.

Debian’s very clear on this, isn’t it?
NMcG: Yes, it is. But there is tension there, and just trying to bring that together is, well, I think we’re over 45,000 binary packages now in the latest release, so trying to bring it all together is a lot of work and it takes some time. Just trying to make sure you have that stable, reliable system that carries on working is a trade-off for being able to bring in the new things.

So what is it that you feel you can bring to Debian from maintaining those releases?
NMcG: I think that mostly it’s my relationships with all the people in Debian. So not only the release managing, I’ve done lots of press work, secretary work, before that I was one of the founding people of the secure testing teams, and I’ve written policy for web apps and various things. So I’ve interacted with so many people over the years that it’s a thing I think I can help with, which is basically getting people to talk to each other and work with each other well, because I know everyone so I can go and talk to FTP masters who I’ve had beers with in Mexico or Edinburgh or all over the world at DebConf. And just being able to use that personal relationship to try and help things along and get progress.

Is it something that you’ve always wanted to do, or is it just something that felt like a natural fit considering what you’ve done before?

Neil is the first Debian Project Leader since the anti-Systemd faction left to fork the Devuan distribution.

NMcG: I certainly didn't enter Debian thinking that. It's something that, certainly at the moment, is the right thing for the project and for me to be able to do. We'll see in 12 months if it was a horrible mistake or something, but I've always been aware of the amount of time it takes and the amount of dedication that's needed to actually do the job well, and making sure I was able to do that. The time was right to do that, and it's always a case of seeing who else is running and working out if you think you could do a better job or if you believe that, you can lead the project then.

LV Was there a point when you thought "Oh no, I've gone and done it now"?
NMcG: Probably about five minutes after the election results came out! There were mixed emotions. The election results came out late on Tuesday evening and I took my new post on Friday morning, so there were a few days to try to work out what was going on. It's slightly daunting I guess, the amount of work, but it's also exciting that I'm going to actually go and do this.

LV Have you started it with a duration in mind, or are you just going to see how it goes?
NMcG: For the moment, a year. Next time the elections come up, I'm going to pretty much do the same thing – try and work out whether I still think I'm doing a great job, would the project still benefit from me doing this role or is there anyone else that would be better, and if there's someone else that I think will be better then I'll back them and help them do it as well. Yeah, I certainly don't want to bite off more than I can chew, and commit to doing something too long really.

LV One of the things you said you want to do is spend some of the money that Debian has. Where do you see that money going – is that employing developers, developing the infrastructure?
NMcG: A lot of our infrastructure comes from machines that have been donated to us, and a lot of our hosting comes very nicely from ISPs that use Debian and like it so we get a lot of that. But hard drives need replacing and upgrades need to happen and machines get old so that they eventually need replacing, and we do quite a lot of that.

I'm not particularly keen on employing developers as a whole. That's been a thing we've looked at as a project in the past and it's caused a lot of disagreements within the project on how we do that, because there's a real risk that you create two classes of developers: the normal everyday volunteers, and those who are blessed to be paid by the project. In a volunteer project, that can create a lot of tension.

But there are things we can try and do to improve Debian and the way it works. We work online entirely, through email, sometimes IRC, but our primary mechanism is mailing lists. And the best way of getting progress is to get people together, actually sitting around the table and working together at it. So a lot of our funding goes for helping

"I'm not keen on employing developers… In a volunteer project, that can create tension."

things like DebConf, which is our annual conference. This year it's in Heidelberg, Germany, and next year it's in Cape Town in South Africa. Some of the new contributors can't afford to go there, so there are some programs to help pay for their plane tickets. It's really important that people get together.

The other thing I'm keen on is average. I'm sure we know that female and minority representation in software is pretty bad, and in open source software it's even worse.

And so there are programs like the Outreach, which started out as the Outreach Program for Women. Debian this year will be funding a minimum of four slots and is about to try and improve it. Hopefully we can use some of the funds that have kindly been donated to us for that.

LV Do you know where the funds for Debian come from? Is it basically individuals, is it companies?

NMcG: The vast majority of our everyday unallocated funding comes from individuals. We do have some corporate sponsorships, but that's mostly for the conference itself because we don't charge for attendance. We have to pay for the venue and various things, and also a lot of the accommodation is paid for by the organisation, so that people that want to come can do that and it doesn't cost them anything.

Debian has never really solicited donations, apart from DebConf, and we haven't really ever asked for any.

LV I know for such a hugely vital distribution...

NMcG: Yeah, I think the last time I checked we get about 40 or 50 thousand pounds a year of unsolicited donations, so making sure we're able to spend that effectively and actually use that for good projects is important.

LV Do you think that reflects the nature of Debian itself? That it's such an important distribution, you'd expect it to have a formal structure and be much more worried about donations, but it does everything at its own pace.

NMcG: Yeah, it kind of hit me a couple of days after I was elected – I went to the pub and five or six people who I know all came up to me and offered to buy me a pint, because they'd all heard about Debian and they'd all heard the news. And so sometimes I think we do forget about how well known Debian is now. It doesn't go out and trump its own horn, and try and do lots of marketing and drive it. But it is the basis of a lot of these systems that are out there, and lots of other distributions as well. It's quite an important distribution, we've found.

LV It would be a good contender for the most important distribution in the whole world!

NMcG: Yes, it's certainly one of the earliest and the basis for a lot of things.

LV What made you chose Debian? So, 12 years ago, what was there... Mandrake... there was no Ubuntu...

NMcG: My first install I think was Mandrake. I was at the University of Sheffield as part of the IT committee there, which basically ran the web servers for people to put up their websites because there wasn't any student web offering in those days. And then my housemate who I moved in with during the second year happened to also be a Debian developer. I'd used Debian a few times before but they encouraged me to get into that and I think I packaged an application called Drivel, which was a LiveJournal client at the time. So that was my earliest contribution.

LV That must have been quite difficult to put together?

NMcG: Yeah, it's become a lot easier now. The installer hasn't changed much since then. It's all been fairly well put together, and fairly stable, but it just requires a lot of work to make sure that the system as a whole can be installed easily. Certainly over the years, it's got a lot easier to install and put together. I mean you can just insert a CD, say 'go', accept the defaults and you know that in the end you'll have a really good working system.

LV So is there something about Debian that you particularly like?

NMcG: I think the primary thing about Debian is that it makes the decisions that are most technically correct beyond all levels.

Owing to its stability and number of packages, Debian is a popular base system for other distros – such as Ubuntu.

Even when it comes to something like Systemd?
NMcG: Yeah. And, as a community, we make decisions and we do it on the basis of getting it right and making sure things are aligned.

We're not a distribution that hides its problems. We do everything in the open, and that way of working together works really well. The Debian mailing lists have been less intense recently than they were about five or six years ago, and the project's calmed down a lot and is working.

Oh really?
NMcG: Apart from the rather large Systemd [disagreement]; that used to be a fairly common atmosphere on the lists but it's become a lot more calm and people just work together to do it. But even despite those huge flamewars that you used to get, we still produce the right solution and try to make sure things are stable and we carry on going together. We make it a distribution that people can rely on.

And do you have to remain impartial on decisions like that, now that you're the leader, or do you have to ensure there's debate and everyone gets heard?
NMcG: I think my role is to help lead discussion. I'm not a leader who says "We're doing this – follow me!". Obviously, on some issues, I have my own views on where we go…

And do you still state those?
NMcG: Yes, absolutely. But part of my role is to represent the views of the project as a whole, and to make sure the project has the discussions and that we come to a good consensus.

One of the things Debian is trying to do is PPAs (Personal Package Archives). Why is that important?
NMcG: It's a way of easing the development that we do, and making it easier to stage things and how we do it. We have a slightly different plan from the way Ubuntu does it. I don't plan to pushing something where anyone can have their own PPA with different software from anywhere.

For example, you could have a PPA to stage a new version of a library or a new version of, say, Gnome or KDE, and make sure it works altogether before we move it into our own stable distribution. And that should help us all ensure that things remain working a lot better, it's all about quality and making it easier for people to test things and create stuff.

Despite RPM being officially adopted by the Linux Standards Base as the standard Linux package format, Debian's Deb format is still extremely popular.

Are you planning to do the same as Ubuntu has done, and have all the software hosted in the same place?
NMcG: Yeah, it's already fairly easy for people to create their own repositories anyway, but being able to create a new version of, say, *Chromium*, or I think there was a new *WordPress* release out recently, and just being able to see that that works stable and testing and unstable, all at the same time, being able to use that to make sure nothing's going to break.

Would Debian's PPAs require anything from Launchpad to be able function properly?
NMcG: No, but there are a few systems out there that we're looking at and so we could write our own which goes with our own software. There are other build softwares that we could pull in. I think we looked at Launchpad, which is fairly tightly integrated into Ubuntu. Obviously, that's a bit of a deal breaker for us. It's been something we've been looking at for a long time. And it's

As a defender of the free software flame, Debian takes a firm stance on copyright issues – which is why *Firefox* in Debian is rebranded *Iceweasel*, for example.

something that the project and members of the project want. It's just trying to make sure we can accelerate the process.

Do you think people will be installing stable versions of Debian alongside PPAs to get the latest version of, say, *Chromium*, or are you going to want people experimenting?
NMcG: That can happen already, because we have the backports system, where we take things and do it that way. It's a way of making it easier, so for example, I think the Mozilla packaging team have **mozilla.debian.net** and they put their latest ones and the backport things to stable, so it's making it easier for that.

And for some people, it is important to get hold of the new versions and try and run them alongside the stable versions. But I think it's something that will complement stable, rather than change it, because all the larger organisations don't want to upgrade their software every six months and have a new distribution, so it's trying to keep that adequate.

"I don't want to go back to the days of finding a random application on a website."

Do you think package management in general can change? For instance, there are things like Snappy Ubuntu Core and the kind of containerisation and sandboxing of some packages so that different libraries can be installed alongside one another.
NMcG: Yeah, there are advantages and disadvantages of having that. A few years ago, there was a big security vulnerability in *Zlib*, which affected almost every package out there. The ability to update that and manage that, if you have to do it in essentially every application, is a huge hassle. Being able to maintain that sort of level of core system is important.

And that's how it should stay.
NMcG: Yeah, and there's also an issue with trust when it comes to random packages from the internet. I don't want to go back to the Windows days of finding random application on a website and you download it. In Debian, people trust Debian to do it.

We have security updates that we backport and we don't break the software, and that doesn't really come from using containers in that way. I think it reflects Debian's focus on producing those stable systems and making sure people can rely on and trust their systems.

And Debian doesn't feel any pressure just to go along with the current trends...
NMcG: Yeah, we created the **.deb** format and there have been lots of changes, like RPMs and various updates, but I still think the **.deb** format is one of the best package formats out there for managing packages. The complex way you can portray dependencies between them, and make sure things don't break, they stay there.

What do you think your biggest challenge will be over the next 12 months?
NMcG: Well there are a few things that have come out recently. Just before Lucas left, he was talking with various people about things like including ZFS on Linux, which is a tricky one because it's CDDL licenced [Common Development and Distribution Licence] rather than GPL, so there could be some licensing incompatibilities there. And it's the start of a new release, and so this is about the right time for a lot of churn.

Trying to make sure that we get those good foundations in and trying to work out how we can deal with, for example, the Free Software Foundation not agreeing that ZFS is compatible even as module. Although, on the other hand, people do it all the time. And so there is that pressure from both sides to try and marry up those two, and make sure Debian remains a free software distribution.

Is it possibly the most important thing, that Debian remains true to its origins as a defender of software freedoms, rather than taking the easy road?
NMcG: I think so. From a philosophical point of view, it is right that we do produce a good software distribution, so you don't end up with vender lock-in and people can have that trust in what you do. And additionally, what we're starting to see now is that since the Snowden days, there has been a lot of concern over privacy and what happens with your data. By making sure we have that in free software, you know what's happening with your data and that there's nothing untoward going on.

But also I think that being a free software distribution has been one of the reason that Debian has been so successful in corporate environments, because people know they're not going to be stuck when something like a company CEO is going to change and they're going to be left high and dry. Or also creating downstream distributions as well because, when Debian says it is free software, they know they can trust that. People can rely on Debian.

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
Reminder for the lazy: Hadrian's Wall does not mark the border between Scotland and England.

Regardless of the result of the latest UK general election, we should all be able to agree that the BBC's graphics department is doing a fantastic job. The weirdly blinking CGI mannequins of our politicians were uncannily lifelike, and somehow looked more human than the real-life leaders of the party. It's also extremely likely that without Linux and Free Software, these graphics wouldn't look anywhere near as good.

Sure, the BBC has money to burn and can afford to buy in whatever expensive solution it wants, but in the world of graphics, Linux is king. The BBC pays for its own developers to produce whatever its election coverage team requires, and Free Software gives it a fantastic base on which to work – without it, it would be forced to develop from scratch (costly) or buy off the shelf (limiting).

Will it blend?

This freedom gives benefits all the way down to smaller organisations and individuals. Anyone can download Synfig Studio, for example, watch a few video tutorials and start using an incredibly powerful professional-quality animation tool. Free doesn't mean cut-down – it's what gives software the potential to be excellent.

andrew@linuxvoice.com

On test this issue...

BOOKS AND GROUP TEST

Sharing documents, calendars and photos via Dropbox is convenient, but as it's a closed-source service, you have no idea who's rummaging through your files. Condoleeza Rice could be reading all your shopping lists, for example. The way to alleviate this fear is to choose some other software that will do the same job, so we're looking at some of the best solutions for Linux to preserve your peace of mind and get all the features you need. And in the world of books, we find out who Anonymous are, and learn *Qt* coding and teach our addled minds how to think.

Ardour 4

If you're wondering why this issue is late, blame **Graham Morrison** and his obsession with audio software.

DATA

Web
http://ardour.org
Developer
Ardour community
Licence
GPLv2

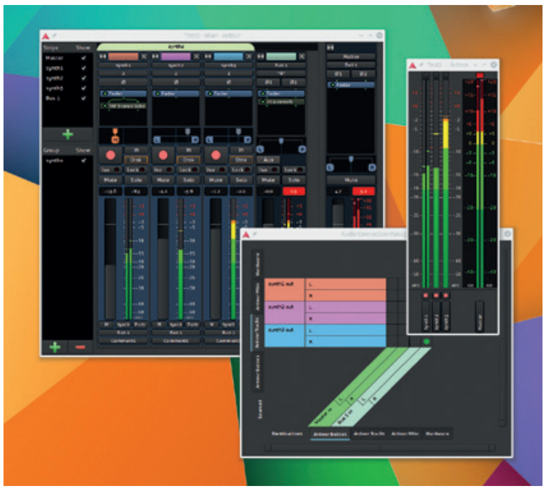
Music production is no longer niche. Thanks to a proliferation of apps, synths and effects on tablets and smartphones, and a new fervour for both old gear and new bits of dinky hardware, lots of people are making lots of music. But before you can share your music and become famous, you need to pull everything from these various sources into one place, mixing your tracks into a single file that can then be shared on SoundCloud or listened to on your phone.

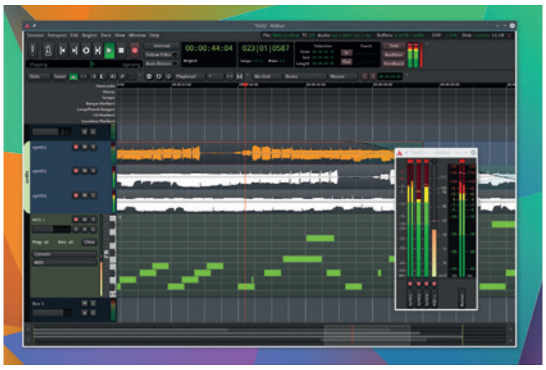
The applications that do this are called 'Digital Audio Workstations' (or, DAWs), a term that pre-dates phones having more processing power than a Commodore Amiga. These are powerful, all conquering tools that skilled engineers use to coax harmony from discord. But they're essential for us mortals too, because mixing and mastering your sounds is beyond the scope of *Audacity*. And in another throwback to a bygone era, these applications can still be expensive. The latest version of one of the market leaders, *Cubase*, for example, is £370.

"Ardour is a digital audio workstation for Linux, OS X, and for for the first time, Windows."

Ardour is one of those applications for Linux, OS X, and for the first time, Windows (thanks to a Google Summer of Code project). It's open source and it's free. But it's also the full time occupation for its lead developer, Paul Davis. While *Ardour* is free in all senses of the word it's definitely worth giving a little back when you do make your fortune.

Ardour is already widely used, both in commercial studios and as the framework for other commercial software, and version 4 is a major new release with many contributions from lots of different developers,


The monitoring sections are clearer than ever, and give you the perfect overview of your track levels, input and output.


Ardour's GUI makes few concessions to the modern world, except the inclusion of exporting audio directly to the SoundCloud.com service.

and comes two years after version 3.0. The Windows version is important, but the major feature for us is that a previous reliance on Jack has been removed. Jack is a powerful piping system for audio, letting you take an audio stream from your microphone, push it through some echo effect (for example) and pipe it back to *Ardour*. But Jack is tricky to get running properly and doesn't work well alongside other audio layers, such as *PulseAudio*. You'd typically have to kill off any other audio before launching *Ardour*, which always diminished its quick-fix effectiveness.

Sound has been simplified

Ardour will now use your ALSA drivers, just like most other audio applications. The low-level nature of its access means desktop audio may stop, which happened in our case, but service is resumed after quitting *Ardour* again. With our hardware, both on a laptop and a semi-pro Focusrite Saffire Pro 40, we had better stability with ALSA than we did running through Jack, especially with low latency settings. Latency is the delay between the sound going in and the sound going out, and we were able to get this down to a barely perceptible 10ms, even with the laptop's Intel HDA audio.

This direct access to ALSA is a huge advantage, and there's obviously been a lot of work in the background helping to make this happen. All of *Ardour*'s internal routing previously used Jack to avoid re-inventing wheels, but you can now interface with your audio hardware directly and still use *Ardour*'s incredible routing and connection windows to pipe audio wherever you need it within the application, all without Jack running. Running *Ardour* without Jack also means there's less distraction because you're not piping audio into external effects, keeping you within a single application for longer.

There has also been a graphical overhaul in version 4, with subtle improvements made to contrast, colour vibrancy and highlighting. Behind the scenes, rendering has moved to APIs like Cairo for great output and better performance on high-resolution displays. It looks more professional and much more like its commercial competitors, but *Ardour* is still an application that makes too few concessions for newcomers. Each track has the letters P, A, G, M and S, for instance, alongside the fader and track name. These represent Playlist (what source this track is playing), Automation, Grouping, Mute and Solo, but you'd never know. The learning curve is very steep when you're trying to figure out the meaning of letters while at the same time mastering a new environment. Nearly every other application we can think of uses more illustrative icons for these functions.

This austere approach can be seen in many aspects of *Ardour*'s GUI, and while we appreciate that mixing consoles can be incredibly complex, we'd love to see *Ardour* become more accessible. It's almost impossible to divine functionality from the GUI alone, and that's a huge shame because it hides a remarkable amount . The online manual is excellent, but we're as bad as most people at reading the things before diving into a major project.

Living colour

You'll spend most of your time editing – the process of trimming blocks of audio (known as regions in *Ardour*), moving those regions slightly to change their timing and cutting mistakes out. (This is what we do when we produce the Linux Voice podcast.)

The already ultra-powerful editing facilities have had a few upgrades. Automation lines are easier to edit, which is important, because they enable you to fine tune your mix without moving the faders during playback, and the rather unintuitive use of the S key to split blocks beneath the cursor has been made more accesible with a real 'cut' mouse tool, complete with a 'scissors' vector-based icon in the toolbar. However, we couldn't find a way of using this tool and selecting multiple regions at the same time, which is something we rely on, and we still would love to see region selection locked so we can simply split and move new regions without reselecting the same audio with every single operation. We also wish tracks could be moved into a folder, not a group or a bus, so they can be hidden unless unfolded, much like in an IDE.

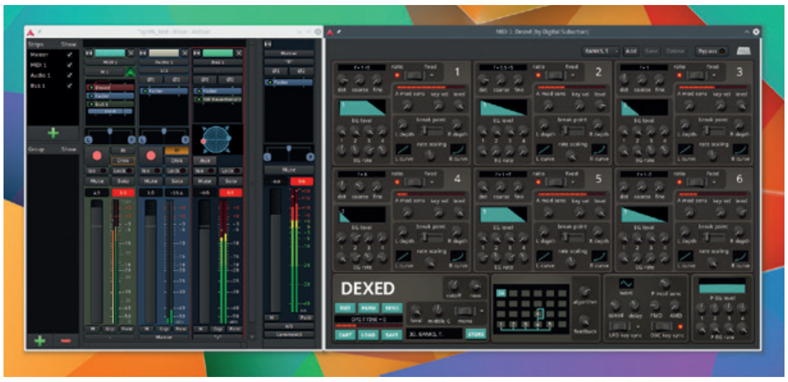
If you prefer putting your hands on real sliders and keys, *Ardour* still has good support for physical external remotes, and there's new 14-bit support for controllers like our Behringer BCR2000. MIDI regions, the biggest new feature for *Ardour 3*, now co-exist with audio groups and tracks, and *Ardour* can now use Linux VST synthesisers when Steinberg's VST development libraries are installed, alongside native LV2 audio generators and effects. MIDI editing itself is still secondary, and we'd rather use a MIDI sequencer for composing rather than the strained track view in *Ardour*. This release also takes 80% less memory, at least in the beginning of a project, and after you've spent some time memorising the keyboard shortcuts, the application feels brilliant to use. In short, *Ardour* is complicated, but there's nothing else like it.

LINUX VOICE VERDICT

You need the training, precision and skills of a surgeon. But Ardour is capable of awesome results.

★★★★☆

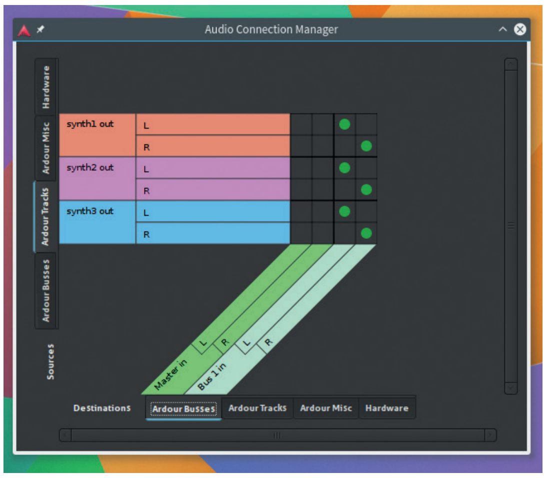
Dialling in the effects



Dexed, seen running here as a VST within *Ardour*, is a complete recreation of a Yamaha DX7 synth and sounds wonderful.

One area where Linux can't yet compete with other operating systems is in the quality and range of the audio effects you often need to use to make your audio sound good. You need decent compression for most tracks, for example, as well as some mastering compression and limiting for when the audio goes through the master bus before a final file. On both OS X and Windows, there are hundreds of effects to choose from, but often, the majority of these will be closed source, commercial and expensive.

Fortunately, there's still a small selection of open source alternatives that are worth it. In particular, VST library compatibility means we get the GUI bundled for the first time. *Dexed* is the best example of this and is capable of fantastic output. And of course, there are commercial options too. Our favourite come from Loomer (**www.loomer.co.uk**) and includes a gorgeous string machine. For some professional mastering effects, take a look at those from OverToneDSP (**www.overtonedsp.co.uk**).



Even without Jack connection system, you can still connect any source to any other input.

Debian 8 Jessie

After fierce debates about the switch to Systemd, Debian 8 is finally here. **Mike Saunders** checks out one of the most significant distro releases of the year.

DATA

Web
www.debian.org
Developer
Debian Project
Licence
FOSS licences

Debian 8, codenamed Jessie, is the most controversial release in Linux distribution history. That sounds rather tabloid-esque, but it's true: in the last two years we've seen giant flamewars on the web, developers quitting the project due to hate mail, and a fork of the distro in the name of Devuan. The reason for all this? *Systemd*, the init system replacing the "bag of bits" that was *SysVinit*.

This release adds support for 64-bit ARM machines, but drops the Sparc and IA-64 (Itanium) architectures due to insufficient developer support. Desktop-wise, Gnome 3.14, KDE 4.11 and Xfce 4.10 are available, and these run on a base of Linux kernel 3.16 and *Glibc 2.19*, compiled by *GCC 4.9.2*. So as expected, it's fairly up to date, but as with previous Debian releases the focus is on maximum stability rather than providing the latest bleeding-edge software.

Debian 8 will be supported with security and critical bugfixes until April 2020 – so it's playing in the same league as RHEL, CentOS and Ubuntu LTS as a reliable distro you can install and forget about, running the occasional **apt-get update && apt-get upgrade** command. Indeed, the distro has proven to be robust in our testing, as expected from Debian, despite the fact that it was released with 78 "critical" bugs still open. If you're running a bunch of servers that simply can't afford any glitches, you're better off waiting for Debian 8.1 to really polish the edges.

"Debian 8 is playing in the same league as RHEL, CentOS and Ubuntu LTS as a reliable distro."

Now, let's talk about *Systemd*. The Debian project had a choice: switch to this new suite of base tools, like almost every other major distro; try an alternative such as *Upstart* from Ubuntu; or just remain with the tried-and-tested *SysVinit*. After countless arguments and developer spats, the Debian Technical Committee voted to switch to *Systemd* – a vote that was accepted by most of the community, but led to the creation of Devuan, a Debian-based distro that features "init system freedom".

Out with the old

But why is *Systemd* so controversial? Until now, Linux has been very much a traditional Unix system: a bunch of shell scripts start up the machine, various disparate tools handle things like resource allocation, logging and timed events (*cron*), and log files are scattered around several directories. *Systemd* replaces all of this with an integrated set of tools built from the ground up. The upsides: administrators and app developers don't need to worry so much about the niggling differences between distros. They can write *Systemd* unit files to start and manage their software without worrying if they're running on Fedora, SUSE or Debian. It's also much easier to sandbox processes.

Systemd keeps track of the state of the system, so it can restart programs that stop or crash, and handle hotplugging events very quickly. The journal brings log files together and means you don't need to juggle regular expressions to say: "show me all warning messages from PID X in the last five boots."

The downsides: it's not really Unixy in many respects. Logs are stored in binary format, the interpreted shell scripts have been replaced by compiled C code, and *Systemd* is very much tied to the Linux kernel. This is why an online war broke out, why developers left and forks appeared, but time will tell if switching to *Systemd* was the right decision. Still, *Systemd* has been used by distros like Arch for several years now, so it's fairly mature, and most admins are learning to live with it – even if they don't love it. If you're a hardcore *SysVinit* fan and Debian lover, however, you'll have to do some distro-hopping now.

On the whole, it's a solid release that handles the transition to a new init system well, and while some of the bugs are cause for concern, nothing dodgy manifested itself in our testing. Kudos to the Debian team for (mostly) surviving a very difficult two years.

Gnome is Debian 8's default desktop, and apart from a dab of distro theming it's largely a vanilla setup.

LINUX VOICE VERDICT

The last big-name distribution to switch to *Systemd*, and the transition has gone well.

★★★★½

Visual Studio Code

After scanning the horizon for airborne pork, **Ben Everard** installs Microsoft's coding environment in Linux.

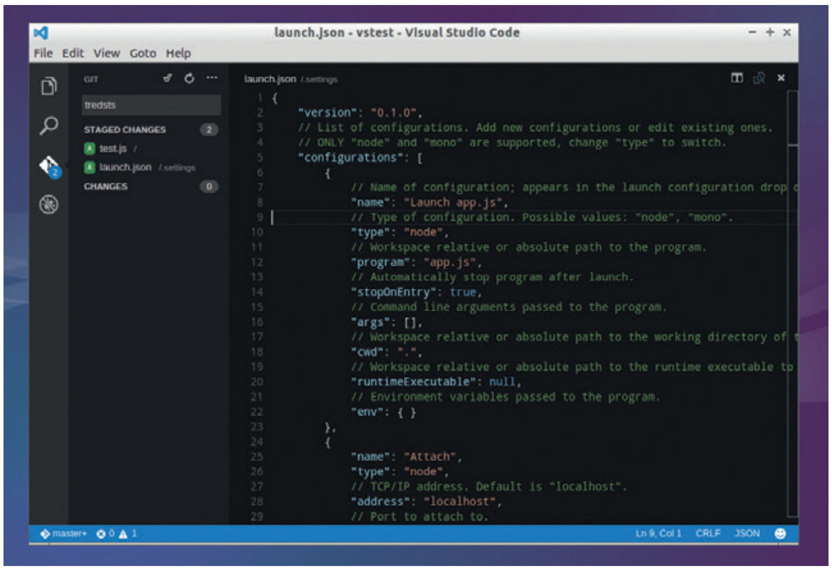
No, your eyes do not deceive you. We really are reviewing a Microsoft programming environment on Linux. In a move that surprised many people, including us at Linux Voice, Microsoft has announced a cross-platform development environment. *Visual Studio Code* (*VSCode*) runs on Linux, Mac OS X and Windows. At the time of writing, there was only a preview release available, which ran for us without any problems.

The install process isn't a great introduction to the software. *VSCode* comes as a Zip file that extracts into the current directory without creating a subdirectory for its files (as most Linux programs do). Once you've decompressed it, there's no instructions file to let you know what to do. Fortunately, it's quite simple: you just execute the binary called **Code**, and it starts.

Aside from the name, *Visual Studio Code* has very little in common with the *Visual Studio* IDE. It's a programmers' text editor with integrated debugger. *VSCode* is based on *Electron*, the engine behind GitHub's *Atom* text editor. At a technical level, this means it's built on Node.js using web technologies.

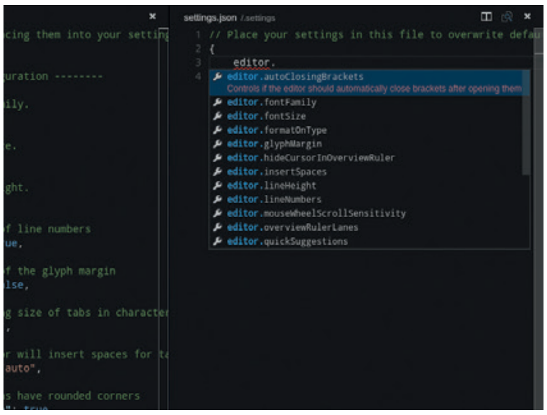
The user interface defaults to light-grey code on a dark grey background. This is easy on the eyes, but if you don't like it, you're out of luck: you can configure quite a bit about the editor in the **settings.json** file, but at the moment, there isn't the option to change the colour scheme.

The best feature of *VSCode* is the debugging support. At the moment, this is limited to ASP.net, C# and Node.js JavaScript, but Microsoft promises support for more languages in the future. You can set breakpoints, and examine variables. It's easy to control and doesn't contain any surprises. If – and this is a big if – Microsoft can extend this debugger support to more languages, then this will be an important feature for future releases. Microsoft has said that its debugger will work with other text editors such as *Vim* and *Atom*, but at the time of writing there weren't any details on this.



```
{
    "version": "0.1.0",
    // List of configurations. Add new configurations or edit existing ones.
    // ONLY "node" and "mono" are supported, change "type" to switch.
    "configurations": [
        {
            // Name of configuration; appears in the launch configuration drop d
            "name": "Launch app.js",
            // Type of configuration. Possible values: "node", "mono".
            "type": "node",
            // Workspace relative or absolute path to the program.
            "program": "app.js",
            // Automatically stop program after launch.
            "stopOnEntry": true,
            // Command line arguments passed to the program.
            "args": [],
            // Workspace relative or absolute path to the working directory of t
            "cwd": ".",
            // Workspace relative or absolute path to the runtime executable to
            "runtimeExecutable": null,
            // Environment variables passed to the program.
            "env": { }
        },
        {
            "name": "Attach",
            "type": "node",
            // TCP/IP address. Default is "localhost".
            "address": "localhost",
            // Port to attach to.
```

It feels surreal running something called *Visual Studio* on Linux, even if it is a stripped-down code editor rather than the full IDE.



```
cing them into your settin
uration --------
ily.
e.
ght.
f line numbers
ue,
f the glyph margin
lse,
g size of tabs in characte
,
r will insert spaces for t
auto",
s have rounded corners
```

```
// Place your settings in this file to overwrite defau
{
    editor.
    editor.autoClosingBrackets
    Controls if the editor should automatically close brackets after opening them
    editor.fontFamily
    editor.fontSize
    editor.formatOnType
    editor.glyphMargin
    editor.hideCursorInOverviewRuler
    editor.insertSpaces
    editor.lineHeight
    editor.lineNumbers
    editor.mouseWheelScrollSensitivity
    editor.overviewRulerLanes
    editor.quickSuggestions
```

Visual Studio Code's code completion ran well, though many programmers editors have a similar feature.

Grudging praise

Code completion is also restricted to the same languages as the debugger. The completion run well even on low-powered machines, and worked intuitively to speed up our coding. The editor can connect to a *Git* repository to enable you to version control your code. Once connected, staging and pushing changes just take a few mouse-clicks.

Some people may find it a little galling that *Visual Studio Code* is closed-source despite the fact that it's largely based on open source components. However, it doesn't violate any of the licences (for example, Node.JS and Electron are MIT licensed), so they Microsoft has the right to do this.

All told, *Visual Studio Code* is a good programmer's text editor. But it's entered an enormously competitive arena: the existing text editors for Linux are mature and highly tuned to developers' needs. For a new editor to stand out, it needs to really have something special. For *Visual Studio Code*, this could be the integrated debugger. However the current limitation to just a few languages and frameworks mean it's not yet useful to most programmers. LV

LINUX VOICE VERDICT

A good editor, but not good enough to tempt us from our regular development environment.

★★★☆☆

DATA

Web
https://code.visualstudio.com
Developer
Microsoft
Price
Free (closed source)

KDE Plasma 5.3

Christmas has come early for **Graham Morrison**.

DATA

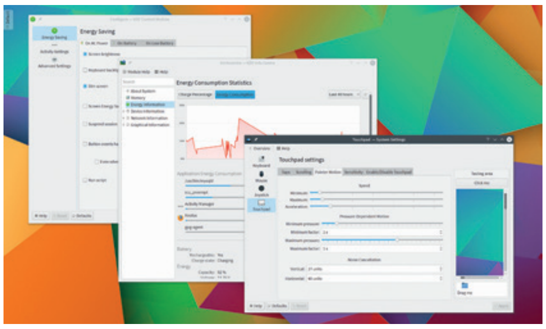
Web
http://kde.org
Developer
Team KDE
Licence
LGPL

The components that come together to make KDE are going through a prolonged phase of rapid development. These components are KDE Frameworks 5, KDE Plasma 5 and the suite of KDE applications ported to the new environment. KDE Frameworks 5.9 was released on 10 April, with the 15.04 applications suite of 72 ported KDE programs released almost a week later.

Many old applications, such as *Telepathy* and *Kdenlive*, have now been ported to Frameworks 5.9, but the ports of both the *Dolphin* file manager and *KMail* have yet to be completed, which is what we miss most. KDE Plasma 5 is the closest to what used to be just the KDE desktop, and when we looked at KDE Plasma 5.2 in issue 13, we concluded that KDE was now good enough to replace those aging KDE 4.x desktops you might be still using, and this is exactly what's happened with the release of Kubuntu 15.04, which now defaults to its version of Plasma 5.2 for the first time.

"There's a new info module, so you can quickly see what's killing your battery."

But development has continued apace, and Plasma 5.3 sports some decent upgrades that might make it worth Kubuntu users hunting out the Kubuntu Backports PPA. We were complaining about the lack of power management in KDE only six months ago, for instance. Our specific complaint was that you couldn't even get an estimate of how much battery life you might have left. And Plasma 5.3 goes well beyond this requirement by not only providing 'time until charged' and 'time until empty' statistics from the panel widget, but also letting you control screen and keyboard brightness, button events and when sessions get suspended on AC power, on battery power and on low battery. You can even create different settings for different activities, perhaps adding more aggressive power management in an activity you can use while travelling, for example. The icing on the power management cake is that there's a new info module that lists processes by energy consumption, so you can quickly see what's killing your battery.

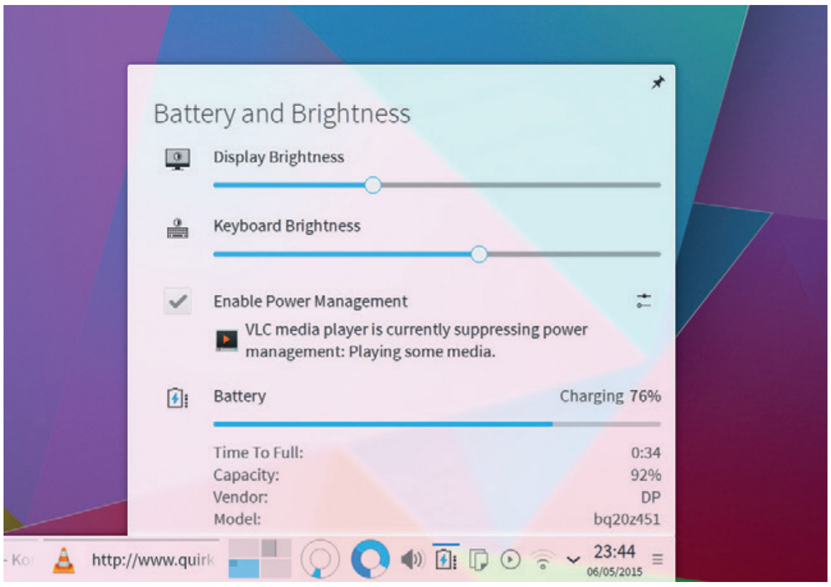

Background widgets in the new KDE can be moved without unlocking the panel. Just press, hold and drag.

Touch and go

Sticking with modules, we can finally dump the *synclient* script we were using to configure our Synaptics touchpad: there's now a touchpad module for the settings panel. Alongside the regular controls, such as tap to left click and pointer acceleration, there's two-finger and reverse scrolling, as well as noise cancellation and palm detection.

Finally, there are two experimental features added to this release. The first is a new media centre application called *Plasma Media Centre*. This runs full screen and is designed to be used as a new session from your login manager when your KDE box is connected to a TV or large screen. It's currently difficult to see why you'd want a KDE solution over something like *Kodi*, which does an almost perfect job. The KDE version rifles your storage for audio and visual content and presents thumbnails of its finds like a robot pet with a pack of cards. The second experimental feature is that the window manager and compositor can now use a nested Wayland session. We weren't able to test this, but it's great to see KDE one step closer to the X.org replacement we've all got such high hopes for.

The battery icon will show you if any applications are blocking the sleep process from activating (as *VLC* is doing here).


Battery and Brightness

Display Brightness

Keyboard Brightness

✓ Enable Power Management

VLC media player is currently suppressing power management: Playing some media.

Battery — Charging 76%

```
Time To Full:      0:34
Capacity:           92%
Vendor:              DP
Model:         bq20z451
```

http://www.quirk — 23:44 06/05/2015

LINUX VOICE VERDICT

This release is made for laptop users – it's difficult to resist the upgrade

★★★★☆

Synfig Studio 1.0

Mike Saunders finally puts his crayons and tracing paper away.

Synfig Studio has quite a lot in common with the 3D studio and rendering powerhouse, *Blender*. They're both graphical applications that generate animations, they're both notoriously difficult for beginners to get started with and they both started life as proprietary software within a commercial setting before being open sourced after their custodians experienced financial difficulty.

In *Synfig*'s case, it's taken almost 10 years to get from proprietary beginnings to a 1.0 release, and while development has sometimes been slow, its release is a monumental achievement. But *Blender* and *Synfig* are also very different. *Synfig Studio* is specifically developed to generate 2D animations, as you'd find in cartoons, rather than the 3D you now find in everything from Jar Jar Binks to *Jurassic World IV*.

In use, *Synfig* is much more like *Inkscape* than any other application, and *Inkscape* even supports *Synfig*'s file format. This is because it deals mostly with vector images. These are pictures that can be infinitely scaled because they're constructed by joining points, lines and curves together, just like fonts.

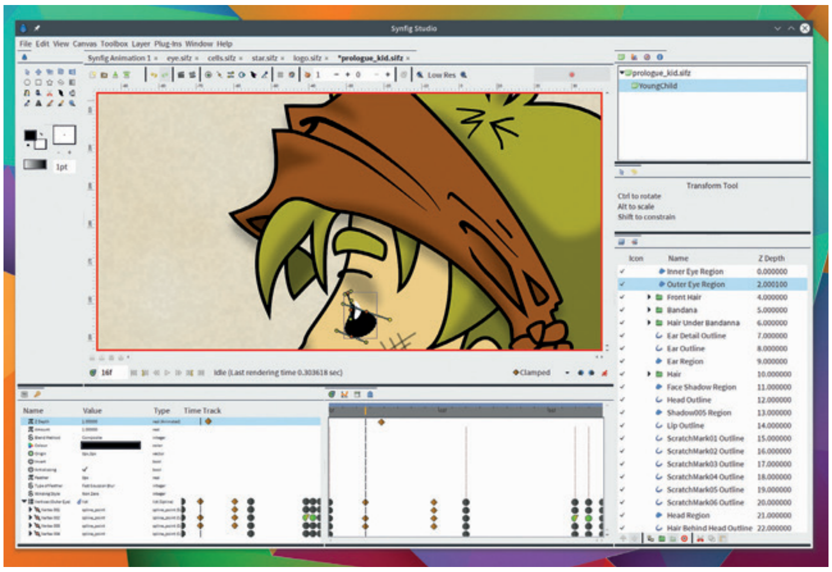
Alongside the essential drawing tools for creating various shapes, lines, curves and gradients, *Synfig* uses a timeline for plotting changes in these points over time. Just like the timeline in *Blender*, *Ardour* or *Kdenlive*, the timeline is used to map changes which are played back as a cursor travels from left to right.

Cross keys

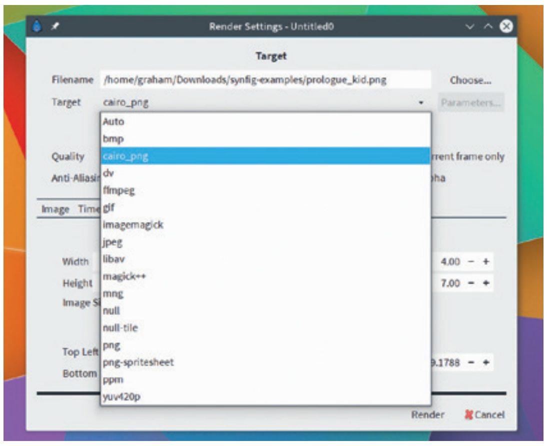
As with any modern animation studio, these changes occur at points you create known as keyframes. Anything that changes between one keyframe and the next will be interpolated, or tweened, to create smooth transitions from one keyframe to the next. This helps the animator avoid having to create every single frame, and can also aid in the generation of the animation by adding effects such as friction to the speed of the animation. As a crude example, if you put a circle at x=10 in the first keyframe and x=20 in the second, *Synfig* would move the circle across points x=11, 12… 18, 19 to complete the animation. *Synfig* enables you to define how you want this transition to happen. Choosing 'Ease in/out', for instance, would accelerate and brake the animation rather than moving at a constant rate. And you can do this with many other elements within an image too, such as an object's fill colour or the blend method.

Getting to grips with all this potential is difficult. We watched several excellent tutorial videos on YouTube to get ourselves familiar with the application, and these helped immensely. So too did the online help, which includes some great tutorials to help you get started. If you've used *Inkscape* or even *Gimp*, the GUI itself is fairly easy to understand. There are layers and sets, where you group objects together such as with an eye or an ear. We liked the sketch tool for mapping out ideas without having to include them or remove them for the final image and the timeline itself is relatively straightforward.

Unless you know what you're doing, things can get out of hand. Images are made up of dozens of spline points and vertices, all of which can be animated, and it's difficult to know where to start. However, this shouldn't put off potential animators, as it's wonderful having this kind of power in an open source tool. LV



Cairo is used as the rendering engine, and complex views and animations can take their toll on your system.



Filename /home/graham/Downloads/synfig-examples/prologue_kid.png

Target cairo_png

Auto
bmp
cairo_png
dv
ffmpeg
gif
imagemagick
jpeg
libav
magick++
mng
null
null-tile
png
png-spritesheet
ppm
yuv420p

There are plenty of export options, but no animated SVG.

DATA

Web
www.synfig.org
Developer
Robert B Quattlebaum Jr., Adrian Bentley & team.
Licence
GPL

LINUX VOICE VERDICT

More than a simple point-and-click animator, this is a serious tool that needs some skill to use.

★★★½☆

We Are Anonymous

Ben Everard is anonymous… wait, no he's not.

For about a year, the hacking collective known as Anonymous made headline after headline in newspapers around the world for their cyber insurgency. In *We Are Anonymous*, Parmy Olson uncovers what went on behind the scenes. Much of the action took place in IRC chatrooms as people planned, coordinated and executed attacks on targets around the world, bringing online services to their knees.

Olson follows some of the inner core of Anonymous through from the early days to the splinter groups such as LulzSec that terrorised parts of the internet. She follows them through message boards, public IRC, and private IRC channels to discover what really happened for the brief period when it seemed that they had mastered the internet.

It's a bittersweet tale of righteous anger, good-old-fashioned anarchism and petty squabbles. Most of the protagonists are in jail now, so the truth can come out without fear of repercussions. The details in the book (which come from interviews with the people involved, chat logs, and court records) portray a very different story than the one given by the media at the time.

Not everything was as it appeared. Some of this is due to certain members of anonymous deliberately trolling the news media at the time and feeding them false stories, while other parts of this are just due to the full picture not being available to anyone at that time.

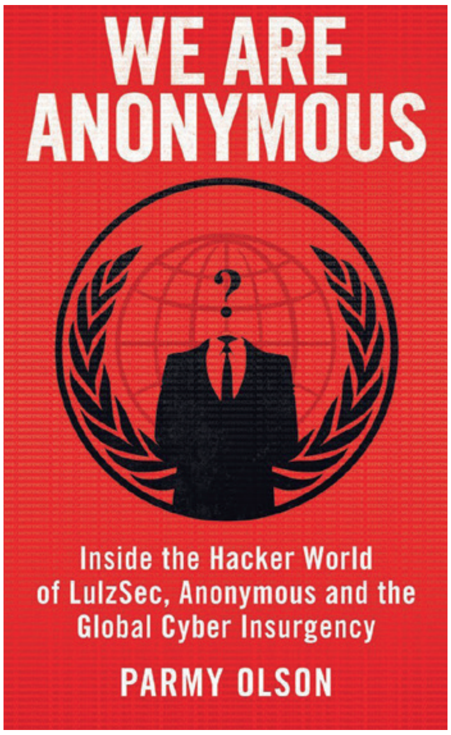
Anonymous claimed they don't forgive and they don't forget. Neither did the FBI.

LINUX VOICE VERDICT

Author Parmy Olson
Publisher William Heinemann
Price £12.99
ISBN 978-0434022083

A revealing peek deep inside the secretive world of the hacking collective.

★★★★☆


WE ARE ANONYMOUS
Inside the Hacker World of LulzSec, Anonymous and the Global Cyber Insurgency
PARMY OLSON

Qt 5 Blueprints

GTK or Qt? **Ben Everard** ponders the eternal question

Qt is one of the most important widget toolkits for Linux and cross platform development in general. It runs on most desktop operating systems and some mobile ones, and it's easy to use once you've learned your way around the labyrinthine tree of classes available to developers. In other words, it's a really important graphical toolkit for any programmer to know.

Qt 5 Blueprints is an easy introduction to *Qt* for C++ programmers. You can use *Qt 5* with other languages, but you'll be better served by a different book, as this one is closely tied to the language. The examples it gives are straightforward, so you don't need to be a skilled C++ programmer to follow it, but you'll probably struggle to get the most out of the toolkit unless you've got a reasonable grasp of that language.

Huang does a good job of going through his examples slowly, and everything is well explained. Perhaps the biggest problem with any book on *Qt* is the sheer size of the toolkit: it would be impossible to cover everything. *Qt 5 Blueprints* sticks to using just a few of the basic widgets, and trusts that once the user fully understands those, they won't have too much difficulty picking up the others.

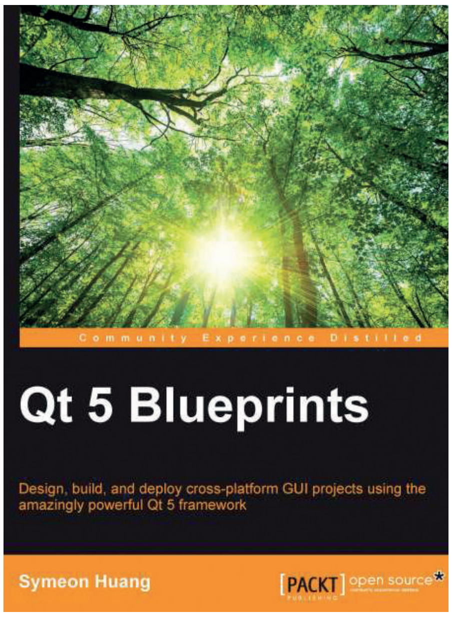
This book is probably best suited to people with a little C++ experience, but not much knowledge of GUI programming. After all, *Qt 5* is well documented, and experienced programmers will want something with more depth than this book.

LINUX VOICE VERDICT

Author Symeon Huang
Publisher Packt Publishing
Price £32.99
ISBN 978-1784394615

A gentle introduction to *Qt* for C++ programmers.

★★★★☆


Community Experience Distilled
Qt 5 Blueprints
Design, build, and deploy cross-platform GUI projects using the amazingly powerful Qt 5 framework
Symeon Huang
PACKT open source

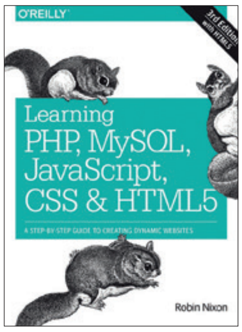
There are *Qt* bindings for many languages, but *Qt 5 Blueprints* focuses on C++.

Learning PHP, MySQL, JavaScript...

Graham Morrison finally gets beyond using <tables> for everything.

If you're just discovering how the World Wide Web works, you could be forgiven for thinking that web development is no longer something a beginner could get into. You could argue that you're better off learning *WordPress* or a shiny and modern web framework. However, we'd argue that it's still worth learning the basics.

This is a book that encapsulates what we'd describe as the basic technologies of the web, and it does it for the complete beginner, starting with programming concepts and moving through PHP, *MySQL* and JavaScript before ending up with HTML 5. We like the way it builds up your skills, and while the book is undoubtedly large, it tackles so many core technologies that each section is manageable. If you get a taste for web development, you'll certainly want to augment what you learn here with more specific titles. We'd also like to see something on sysadmin, and there's next to nothing about Linux. But what we liked most is that it makes building your own dynamic website feasible, which is both liberating and an important idea in itself.


If we'd typed the title out in full, we'd have written half the review already.

LINUX VOICE VERDICT

Author Robin Nixon
Publisher O'Reilly
ISBN 978-1-491-94946-7
Price £33.50

A brilliant overview that gives you everything you need to build an online empire.

★★★★★

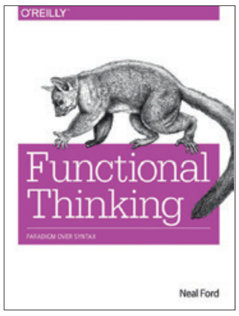
Functional Thinking

Never again will **Graham Morrison** mistake functional for functions.

Programming with a functional language is now more popular than ever, but if you've been raised in a diet of procedural programming, as most of us have, it's a difficult concept to grasp. Which is why we keep reading books about it (see *Becoming Functional* in issue 9).

Functional Thinking is a good attempt. Each chapter in this small book, is full of practical examples, mostly pointing out how terrible the procedural or object oriented approaches are by comparison.

There's no single instance or page where we had a Eureka moment. But as you work through the chapters, the ideas slowly percolate through your brain matter until they do start to have an effect. This is evident by our imagination working on problems we'd previously solved procedurally. But it is hard work, and for us at least, not much fun – we enjoy hacking around with procedures too much. What we imagine is missing is a more formal mathematical background, putting functional languages in context. But most of all, for us, it's also missing the spirit of adventure.


At only 164 pages, it's a book that's quick to read.

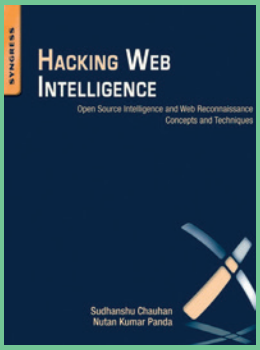
LINUX VOICE VERDICT

Author Neal Ford
Publisher O'Reilly
ISBN 978-1-4493-6551-6
Price £26.60

A useful and persuasive resource, but too dry for us, and not great value.

★★★☆☆

ALSO RELEASED...


Even without snooping, there's a lot of information about all of us online.

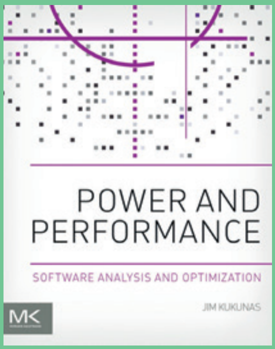
Hacking Web Intelligence

We love the idea behind this – delving into the web to uncover data most of us don't know exists – something called open source intelligence and web reconnaissance. We're off to find a black beret and a suitable location with internet in the Vercors Massif.

Python's quick style is perfect for test-driven development.

Test-Driven Python Development

Python has been a serious development tool for some time. This book will show you how to build a project around the test-driven development process, which Python is perfect for as it revolves around short, testable programming cycles.


Linux runs on all the fastest computers. But it can still go faster.

Power and Performance

If you take a look at our sysadmin section this month (p68), you'll see it tackles the complex issues surrounding asynchronous networks and the many ways you can make mistakes or improve things. Here's a book to outline many of the other possible pitfalls and upgrades.

LINUX VOICE

GROUP TEST

FILE SYNC MANAGERS

Mayank Sharma likes his data to be nimble and omnipresent, and is on the lookout for the ultimate data sync solution.

On test

Dropbox

URL www.dropbox.com
VERSION 3.4.3
LICENCE Proprietary
The closed source segment leader faces stiff open source competition.

Seafile

URL www.seafile.com
VERSION 4.0.6
LICENCE GPL v3
It doesn't offer much free storage, but could it impress with its features?

SparkleShare

URL www.sparkleshare.org
VERSION 1.4.0
LICENCE GPL v3
Will it be able to cash in on the popularity of its Git underpinnings?

BitTorrent Sync

URL www.getsync.com
VERSION 2.0.82
LICENCE Proprietary
Does this serverless tool have an edge over the others?

OwnCloud

URL https://owncloud.org
VERSION 8.0.2
LICENCE AGPL v3
It's going from strength to strength, but how does it match up to its peers?

SpiderOak

URL https://spideroak.com
VERSION 5.2.0
LICENCE Proprietary
Proprietary service known for its privacy-respecting features.

File sync managers

Back before there was Dropbox, the only practical solution to sharing files was to roll up your sleeves and configure a file server. It wasn't for the faint of heart, especially if you wanted to share and sync beyond the confines of your local network. Not only was the process an involved one, it demanded a certain level of expertise that you couldn't expect from an average desktop user. Then Dropbox came along and made the process idiot-proof. Suddenly sharing files and syncing them between computers and devices became a no-brainer.

Dropbox ushered in an era of cloud storage and sync services. It makes even more sense in these days of universal access when everyone owns multiple computing devices of all shapes and sizes and they're all hooked up to the internet.

Shuttling around data on USB drives as you move from the computer to the tablet makes about as much sense as snail mailing an SD card with your vacation pics to friends and family: it's preposterous. In the same way, these services automatically sync the documents you're working on, along with the ebooks you're reading and the music you're listening, to the online storage spaces and make them available on across your devices.

Privacy vs convenience

In this group test we examine six top rated storage and sync solutions that make data omnipresent. Broadly speaking, you can divide these services into two categories. On the one hand we have proprietary services that keep the data on remote storage servers outside your and control. For the privacy conscious, we also have open source services that offer all the convenience of an omnipresent storage service, while keeping the data within the confines of your firewall. Can the self-hosted solutions match the convenience of an online storage service?

"These services automatically sync your documents, ebooks, music and more."

Cloud host vs self host

We're testing two types of services in this group test, both of which have their advantages and disadvantages that you should be aware of before homing in on one. The biggest advantage of cloud storage services like Dropbox is the convenience of a quick rollout. However, on the downside you have to agree to their terms for hosting your data, which may expose your potentially sensitive data to risk. Also, most of the proprietary services have price-bound features and storage space restrictions. The extra effort required to set up your own storage server eliminates both these factors. Furthermore, it ensures full privacy and mitigates the risk of unexpected downtime.

Outside the network

Access your personal cloud from the internet.

By default self-hosted solutions like OwnCloud and Seafile will only be accessible from computers and devices within the local network; you don't get the benefit of connecting from anywhere, as you do with Dropbox.

But that's not to say that you can't access your private cloud from the internet. One solution is to get a static IP address from your ISP and then poke holes in your router's firewall. Or, you can set up Dynamic DNS in your router or local machine.

The smarter way though is to use a tunnelling service such as PageKite. The service uses a pay-what-you-want model. As a non-commercial user, you can use the service for free by filling out a form once a month, telling PageKit how you use the service, but it's definitely worth more than the $3/month minimum they request from individuals. Fire up a terminal and install PageKite with

```
curl -s https://pagekite.net/pk/ | sudo bash
```

Assuming your cloud server is running on port 8000, make it public with

```
pagekite.py 8000 mycloudserver.pagekite.me
```

That's it. Your private server is now publicly accessible on **https://mycloudserver.pagekite.me**. Remember to replace 'mycloudserver' with any name.

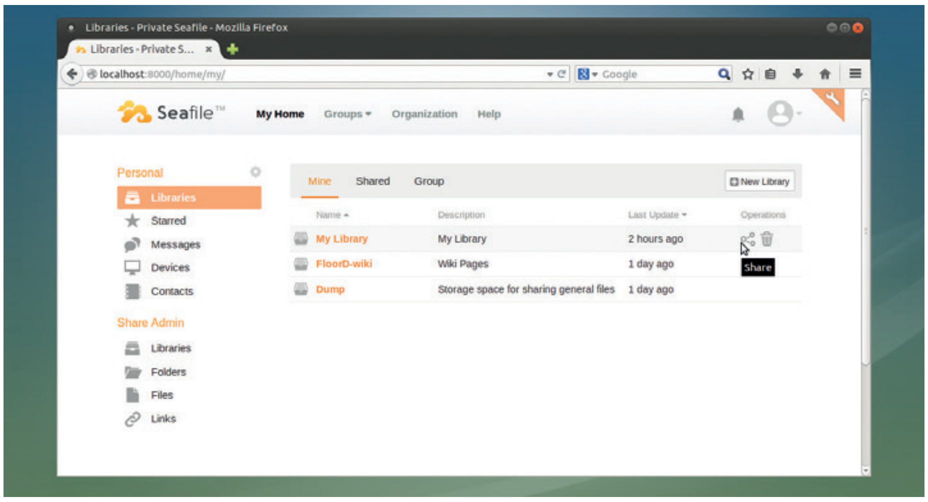
Seafile

Splash and dash.

Seafile is an open source solution that works pretty much like Dropbox and offers two rollout options. If you want the convenient option, you can download a client and set it up to sync across either of the two online storage servers – one in Germany and one in the US – that offer 1GB of free storage space. The other option is to set up a Seafile server within your firewall and extend the same sync facilities as any other cloud sync service.

Setting up a Seafile server doesn't take much effort. While the process is well documented it obviously requires a familiarity with configuring network-based software. For small installations, you can deploy Seafile with the *SQLite* database, while larger setups can use full-fledged databases like *MariaDB*, *MySQL* or *PostgreSQL*. The detailed manual also has instructions on hooking it up with existing web servers, including *Nginx* and *Apache*.

Seafile also lets users collaborate on documents, which is a very handy feature. New versions of documents are automatically generated after each modification, and you can easily switch to an older version without much effort.



Name	Description	Last Update	Operations
My Library	My Library	2 hours ago	
FloorD-wiki	Wiki Pages	1 day ago	
Dump	Storage space for sharing general files	1 day ago	

Regularly check the integrity of your file using the included *fsck* tool.

Easy to manage

You can use Seafile to create libraries that can optionally be encrypted with AES-256, and then add files to them. You can add users and organise them into groups and share libraries with groups, and enable the optional wiki module to enable users to collaborate on a wiki. Additionally, users can also message each other and members of a group can have a public discussion with each other.

Seafile offers flexible options to share these libraries with other users, who can hook to the server via the Seafile clients. Seafile has clients for all the popular desktop and mobile platforms, though the Seafile client for Linux is only available as precompiled Debs, so you'll have to rely on external repositories for RPMs. Every Seafile desktop client has a unique private key. When a client and a server connect, they exchange the public key and negotiate a session key. This session key is then used to encrypt the data transfer.

The service offers flexible options to share libraries or individual files with users or groups of users. You can also transfer ownership to other registered users and even allow them to create libraries. The service gives you the option to share a library with specific contacts or groups of users and enable read-write or read-only access to different libraries. You can even share single documents with another user. Members can upload, download and edit files online or even download the whole libraries from the cloud.

The Seafile client doesn't integrate with the file manager, but you can use it to share a folder on the desktop by uploading its contents into a shared library. You can also use it to create new libraries and download existing ones, and keep tabs on the file transfers.

Also, the service has a flexible version control system that lets you browse the history of a file and restore the file content to an old version. By default, Seafile will preserve the entire history of a file. However, you do have the option to specify a period of time for preserving old files per library.

VERDICT

A well rounded solution with features to impress all kinds of users.

★★★★★

SparkleShare

Not much glitter.

Just like with Dropbox and Seafile, SparkleShare offers the option to sync via protected storage spaces or set up your own SparkleShare server.

When you create a project, SparkleShare displays the SSH address of the host and the location to the shared directory. The service houses all shared directories inside the **/home/storage user** directory, which it creates during installation.

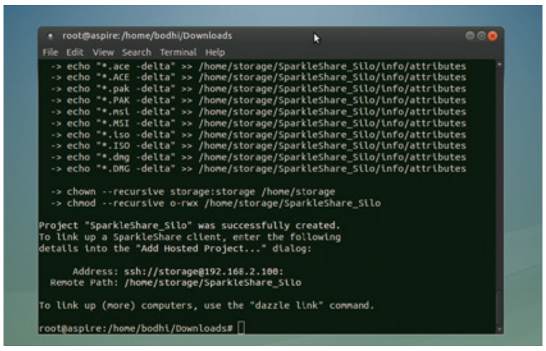
You can install SparkleShare clients from your distro's repository or compile it yourself, following instructions on the website. To hook up these clients to the SparkleShare host, you'll need to know the host's public SSH key and location, which was displayed earlier when you created the share.

Unlike some of the other services on test, SparkleShare is a pure file sharing service that doesn't offer any other features and facilities even as optional add-ons. Also, unlike other services, SparkleShare doesn't have a web-based interface and all the administration tasks must be done from the CLI.

The service has a desktop client that can be used to connect to a SparkleShare and displays notifications about the sync tasks as well as lists changes across all shared directories.

Since it's based on *Git*, SparkleShare has version control built-in. Using the tool, you can share some of your projects with other users, while keeping others private. However, unlike other options you can't share files with links. Also while the file transfers and sync worked as advertised in our tests, the developers themselves admit that SparkleShare isn't great for storing photo or music collections and large binary files that change often, such as video editing projects, which would rule it out for most desktop users.

"Since it's based on Git, SparkleShare has version control built-in."


root@aspire: /home/bodhi/Downloads
File Edit View Search Terminal Help

```
  -> echo "*.ace -delta" >> /home/storage/SparkleShare_Silo/info/attributes
  -> echo "*.ACE -delta" >> /home/storage/SparkleShare_Silo/info/attributes
  -> echo "*.pak -delta" >> /home/storage/SparkleShare_Silo/info/attributes
  -> echo "*.PAK -delta" >> /home/storage/SparkleShare_Silo/info/attributes
  -> echo "*.msi -delta" >> /home/storage/SparkleShare_Silo/info/attributes
  -> echo "*.MSI -delta" >> /home/storage/SparkleShare_Silo/info/attributes
  -> echo "*.iso -delta" >> /home/storage/SparkleShare_Silo/info/attributes
  -> echo "*.ISO -delta" >> /home/storage/SparkleShare_Silo/info/attributes
  -> echo "*.dmg -delta" >> /home/storage/SparkleShare_Silo/info/attributes
  -> echo "*.DMG -delta" >> /home/storage/SparkleShare_Silo/info/attributes

  -> chown --recursive storage:storage /home/storage
  -> chmod --recursive o-rwx /home/storage/SparkleShare_Silo

Project "SparkleShare_Silo" was successfully created.
To link up a SparkleShare client, enter the following
details into the "Add Hosted Project..." dialog:

      Address: ssh://storage@192.168.2.100:
  Remote Path: /home/storage/SparkleShare_Silo

To link up (more) computers, use the "dazzle link" command.

root@aspire:/home/bodhi/Downloads# 
```

The SparkleShare client can connect to the host as well as to other hosting sites, including GitHub and Bitbucket.

VERDICT
It'll appeal to developers but isn't as complete as some other solutions.
★★★☆☆

BitTorrent Sync

Peer-to-peer version 2.

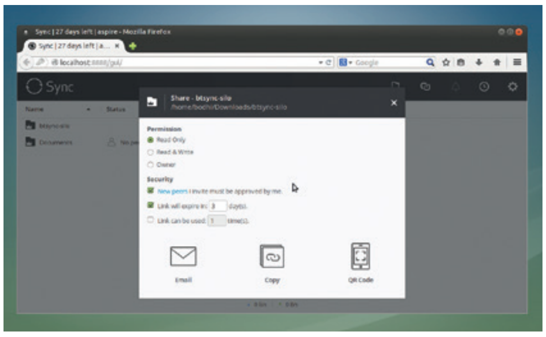
BitTorrent Sync is the proprietary decentralised file-sharing service from the developers of the popular peer-to-peer file sharing technology. The USP of the service is that it employs no server and the files are securely transferred between users. This makes BitTorrent Sync one of the easiest services to get started with.

The service has clients for all the popular platforms, however, the Linux flavour is web-based. There is no installation involved and you just download, extract and run the server. Then head to its web interface and create an admin user. You can then add folders that you want to sync. Each folder has a unique key that must be passed on to any other device that you wish to sync with. You can email the key as text or scan a QR code if you're setting up sync on the mobile version of the service. While creating a key to share a folder, the service gives you the option to limit access permissions. The service encrypts all traffic using a private key derived from the shared secret key.

Keep it simple

The service offers no add-ons and is a pure file sharing and sync service. Also it isn't quite as easy as some of the other services on test for some tasks such as backups. Plus, if you want to share files with other people, they'll need to install the BitTorrent Sync client and equip it with the code for the folder you wish to share.

To simplify the process, the service lets you generate links for sharing. You can set expiration times for the links you create so that the link is only active for a specific length of time. You can also optionally configure the service to notify you when a link is clicked and you can approve and deny access accordingly. The service has two versions. The Free version lets you share up to 10 folders, while the paid Pro version offers advanced features such as on-demand access and folder-level access control.



Share - btsync-silo
/home/bodhi/Downloads/btsync-silo

Permission
[✓ Read Only]
Read & Write
Owner

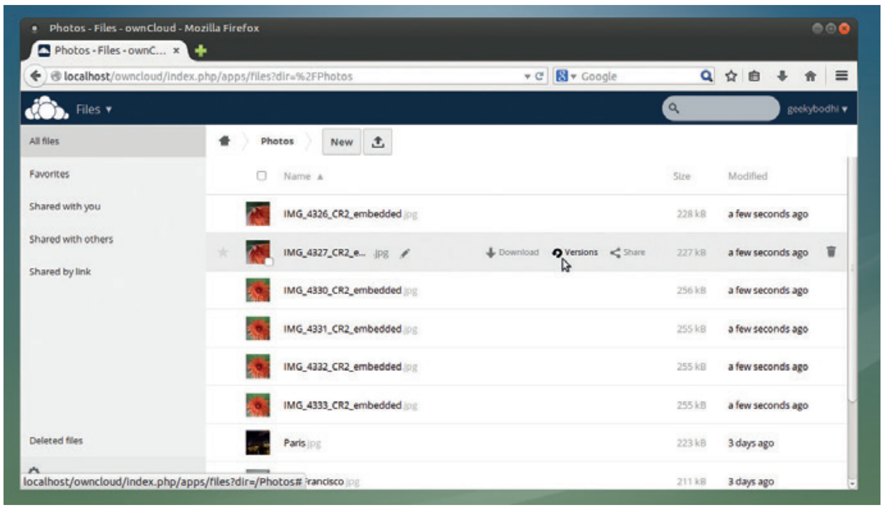
Security
[✓] New peers I invite must be approved by me.
[✓] Link will expire in: 3 day(s).
[ ] Link can be used: 1 time(s).

You can freely test the £26/year Pro service which offers more granularity to folder syncing.

VERDICT
Interesting solution but requires everyone to use the client.
★★☆☆☆

Owncloud

Overcast forecast.



Name	Size	Modified
IMG_4326_CR2_embedded.jpg	228 kB	a few seconds ago
IMG_4327_CR2_e... .jpg	227 kB	a few seconds ago
IMG_4330_CR2_embedded.jpg	256 kB	a few seconds ago
IMG_4331_CR2_embedded.jpg	255 kB	a few seconds ago
IMG_4332_CR2_embedded.jpg	255 kB	a few seconds ago
IMG_4333_CR2_embedded.jpg	255 kB	a few seconds ago
Paris.jpg	223 kB	3 days ago
rancisco.jpg	211 kB	3 days ago

OwnCloud also has an impressive browser-based interface and a file manager that can handle files in most popular formats.

OwnCloud is perhaps the most popular DIY storage and sync solution on test here. You can manually install the OwnCloud server on your machine or from the official repositories for popular distributions. Installing the server is straightforward and well documented. Just like Seafile, by default, OwnCloud uses the *SQLite* database and the *Apache* web server, but it can plug in to an existing *MySQL* database and will also work with other web servers, including *Nginx* and *Lighttpd*.

Owncloud's biggest feature is its extensibility. It's much more than a file storage and sync server. It ships with a photo gallery app, uses *Ampache* for music streaming and has a web-based PDF viewer and ODF viewer.

There's also a task scheduler, an online text editor, and an app to store bookmarks. Plus, you can download other apps from its app store. It hosts productivity tools such as a URL shortener and the *Roundcube* mail server, multimedia tools such as the *JW Player*-based video player, and tools to visualise storage space and scan files with the *ClamAV* virus scanner.

All encompassing

You can use OwnCloud to create encrypted storage spaces, and the server also keeps tracks of all versions of files. Its versioning mechanism ensures you never run out of space (by automatically deleting old versions). Another advantage that we like is that OwnCloud lets you mount external cloud storage drives, such as Google Drive, Amazon S3, Dropbox, and OpenStack Swift, and can seamlessly manage data on these along with that in your private cloud.

OwnCloud's impressive desktop client displays notifications, shows you a summary of sync activity, and offers the ability to throttle upload and download bandwidth, and pause and resume transfers. The client also lets you add local folders and specify patterns for files or directories that shouldn't be synced.

If you have a large group of users, OwnCloud will impress you with its user management options. While adding users in OwnCloud, you can restrict their storage space and organise them into different groups. You can share your admin responsibilities with other users and make certain users group admins, and can select which users or groups you want to share files with and whether you want to give them permission to modify the files. You can share an item with someone who isn't on your OwnCloud server by sharing a URL to that file on your server. You can also password-protect the link and set an expiration date.

VERDICT
The most comprehensive storage and sync server on test here.
★★★★☆

Other options

Sync clients of all shapes and sizes.

If you don't have any issues keeping the data you want to sync in a drive on the other side of the planet, you can use any of the cloud storage services that suit your budget. Almost every cloud storage service now supports Linux, so you have plenty of options, including CloudMe and Mega.

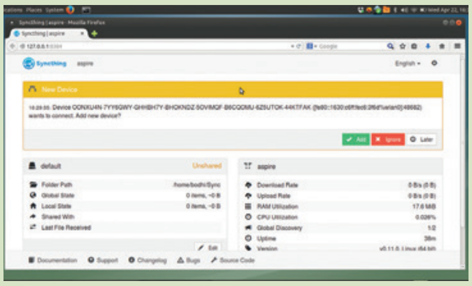
The geekier ones can use the venerable command-line *rsync* utility or its graphical cousin LuckyBackup (as seen in LV015's Masterclass). Both are wonderful tools for backing up data and keep folders in sync across machines but lack features such as the ability to share files with links.

There's also Syncthing, which is often described as the open source alternative to BitTorrent Sync. Syncthing is well designed with security and privacy in mind and works across all major platforms. Setting it up is similar to the process used with BitTorrent Sync, and the services uses a global discovery server to connect clients anywhere on the internet. However, it didn't always work during our tests and we also had trouble connecting a PC to the alpha Android client.

The cloud at home

Another option for the more hands-on among us is to set up a NAS (Network Attached Storage) that offers syncing facilities. One of the easiest options if you go down this path is the proprietary freeware *Tonido* software. Unlike other synchronisation tools, *Tonido* doesn't store the data on its public cloud and only relays it through *Tonido*'s servers.

Then there's the open source OpenMediaVault distro, which has plugins to sync data to USB drives. You can even configure something like BitTorrent Sync to sync data from your NAS.



Syncthing is shaping up to be a very useful and apt replacement for BitTorrent Sync.

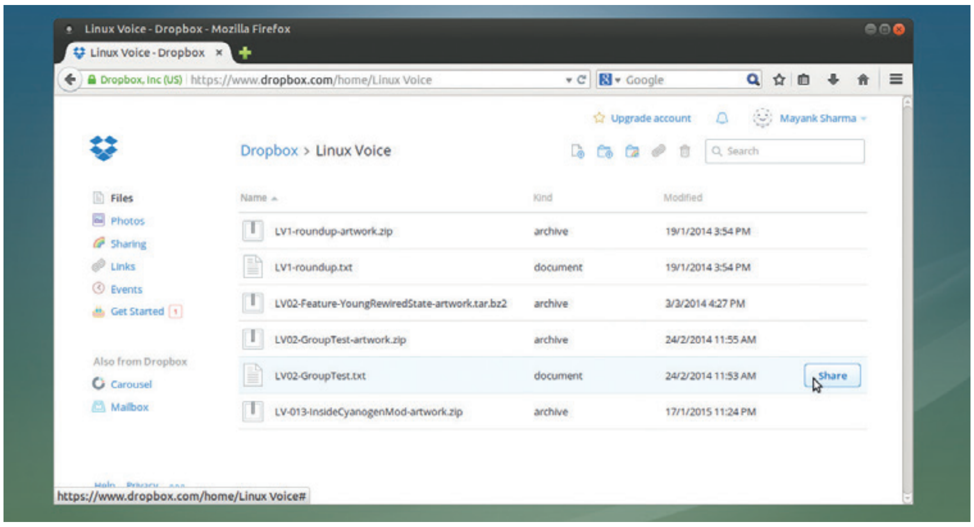
Dropbox vs SpiderOak

The proprietary behemoths.

Dropbox is one of the most widely used storage and sync services. It has clients for all the major operating systems and offers 2GB of free storage space with paid plans starting at $10/month for 1TB. The service creates a **~/Dropbox** folder under your home directory, and any files you drop here and automatically synced to the online Dropbox account and from there to all the devices connected to your Dropbox account.

The service installs a system tray icon with which you can keep track of the syncing status and view a list of recently changed files. The icon also lets you change your Dropbox preferences, including the location of the synced Dropbox folder, and network settings. You can also use it to configure Dropbox's selective sync feature to only sync a selection of folders to your computer.

Unlike other similar services, Dropbox has only a minimal client for Linux, and even that client officially only supports Gnome's *Nautilus* file manager. Using the right-click context menu you can move any folder on your computer to the Dropbox folder. Once inside the Dropbox folder, you can use the context menu to share them with links and view older versions of the file.


Linux Voice - Dropbox - Mozilla Firefox
https://www.dropbox.com/home/Linux Voice

Dropbox > Linux Voice

Name	Kind	Modified
LV1-roundup-artwork.zip	archive	19/1/2014 3:54 PM
LV1-roundup.txt	document	19/1/2014 3:54 PM
LV02-Feature-YoungRewiredState-artwork.tar.bz2	archive	3/3/2014 4:27 PM
LV02-GroupTest-artwork.zip	archive	24/2/2014 11:55 AM
LV02-GroupTest.txt	document	24/2/2014 11:53 AM
LV-013-InsideCyanogenMod-artwork.zip	archive	17/1/2015 11:24 PM

The Dropbox client only supports a limited number of Linux desktops.

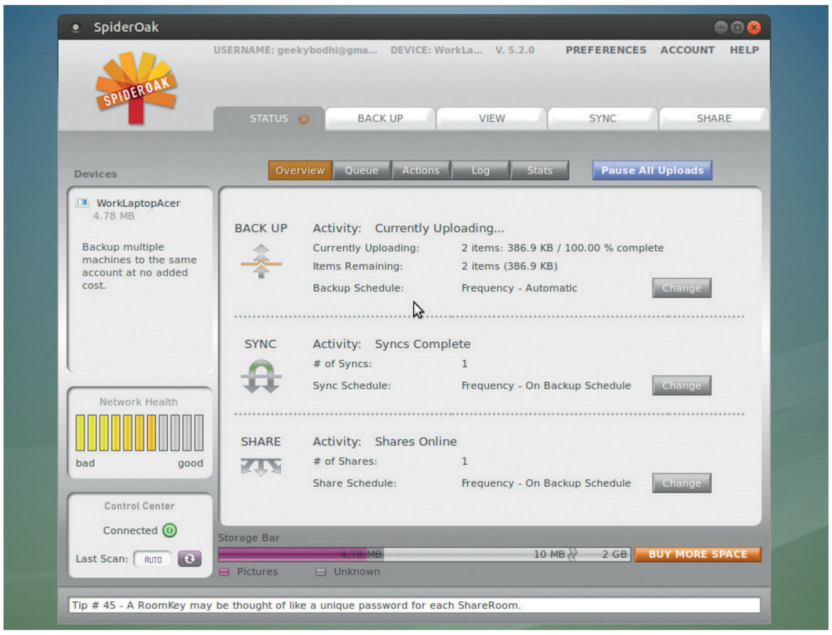
Dropbox has partnered with Microsoft to enable you to edit any *Microsoft Office* documents in your Dropbox from within your web browser. Dropbox keeps multiple versions of all files and from its web interface you can view several versions of deleted files and easily restore the one you want.

Rich oaky flavour

The other proprietary service, SpiderOak, has crafted a name for itself with its impressive list of features and its focus on ensuring its users' privacy. It encrypts data before uploading it, and does so without missing out on advanced features such as versioning. The service also implements data deduplication to minimise the drain on bandwidth and also save storage space. The service offers 2GB of free storage space and has plans starting at $7/month for 30GB.

Unlike Dropbox, SpiderOak can be used to back up the contents of any folder on your computer and even external and network drives. The service has an advanced filtering mechanism, and you can use wildcards to exclude particular files and folders from the backup. Furthermore, instead of sharing single files, SpiderOak lets you arrange the files in your SpiderOak account that you wish to share inside virtual silos known as ShareRooms. Every ShareRoom has a unique URL that enables recipients to view and download its contents. You can optionally password-protect these virtual storage areas as well.

To top it all, SpiderOak's interface lets you keep an eye on all ongoing tasks as well as the current downloading and uploading queue, and much more.


SpiderOak
USERNAME: geekybodhi@gma... DEVICE: WorkLa... V. 5.2.0

WorkLaptopAcer
4.78 MB

Backup multiple machines to the same account at no added cost.

BACK UP
Activity: Currently Uploading...
Currently Uploading: 2 items: 386.9 KB / 100.00 % complete
Items Remaining: 2 items (386.9 KB)
Backup Schedule: Frequency - Automatic

SYNC
Activity: Syncs Complete
# of Syncs: 1
Sync Schedule: Frequency - On Backup Schedule

SHARE
Activity: Shares Online
# of Shares: 1
Share Schedule: Frequency - On Backup Schedule

Control Center
Connected
Last Scan: AUTO

Storage Bar: 4.78 MB 10 MB 2 GB

Tip # 45 - A RoomKey may be thought of like a unique password for each ShareRoom.

SpiderOak's client offers complete control over all aspects of the service.

VERDICT

DROPBOX Doesn't offer any feature that you can't get elsewhere.
★☆☆☆☆

SPIDEROAK Simple to set up with impressive privacy protection measures.
★★★★☆

OUR VERDICT

File sync managers

If you want a solution that's full of features and can do a lot more than just sync files, there's no beating OwnCloud – we use it at Linux Voice to share documents, and it works brilliantly for our needs. But the purpose of the group test is to find a tool that primarily syncs files and offers related conveniences. With that criterion, OwnCloud is overkill, and despite being fans we can't get ourselves to recommend it to users who are looking for a simple solution to keep data between their devices in sync.

The problem with SparkleShare is its involved process, which might not be very intuitive to new users. Also, it involves finding a way to securely transfer the public SSH keys between the host and the client. Then there's the fact that the developers have themselves admitted that SparkleShare isn't really suitable for storing images and audio nor for large binary files that change often. On top of that the developers have put the development of the Android client on hold, which undermines the benefit of being able to sync all your files across all of your devices.

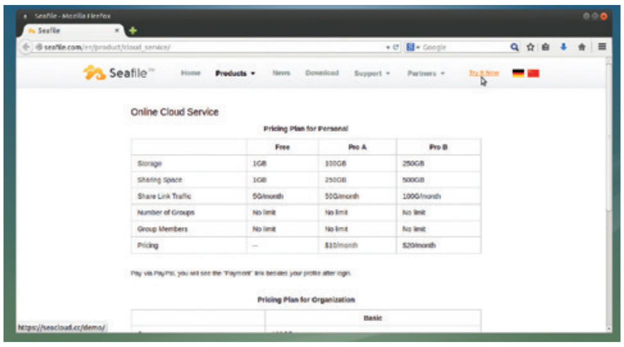
BitTorrent Sync loses out for being entirely focussed on one task – to sync files. This makes it unsuitable for other tasks such as backups. Other factors that tick us off are the fact that it's a proprietary client that you should have on all your devices to sync and share – we'll always choose the best tool for the job, but the idea of becoming reliant on a proprietary solution makes us a little uncomfortable.

Of the two proprietary clients on test, we were surprised by SpiderOak. It's well designed and takes steps to ensure our privacy. If you don't mind offloading your data to the cloud for a price, take your business to SpiderOak.

This leaves us with Seafile, which aces this test. For the extra configuration steps it requires you to undertake, Seafile offers more than enough features to compensate, such as client-side encryption. It offers you all the advantages you get with other services, such as an extensive file-revision history and the ability to selectively share files with links. You can deploy Seafile for personal use or easily scale it up as a collaboration server for a whole lot of users.

All factors considered, Seafile is a robust file synchronisation tool that'll meet the expectations of the greatest number of users. LV

"Seafile offers advanced features such as client-side encryption."


Seafile - Mozilla Firefox
Seafile
seafile.com/en/product/cloud_service/
Google
Seafile
Home
Products
News
Download
Support
Partners
Try it Now
https://seacloud.cc/demo/

Online Cloud Service

Pricing Plan for Personal

	Free	Pro A	Pro B
Storage	1GB	100GB	250GB
Sharing Space	1GB	250GB	500GB
Share Link Traffic	5G/month	50G/month	100G/month
Number of Groups	No limit	No limit	No limit
Group Members	No limit	No limit	No limit
Pricing	–	$10/month	$20/month

Pay via PayPal, you will see the "Payment" link besides your profile after login.

Pricing Plan for Organization

	Basic

You can demo Seafile on its website before deploying it.

1st Seafile

Licence GPL v3 **Version** 4.0.6

www.seafile.com
The right mix of features and usability.

2nd SpiderOak

Licence Proprietary **Version** 5.2.0

https://spideroak.com
For convenience seekers who aren't averse to proprietary software.

3rd BitTorrent Sync

Licence Proprietary **Version** 2.0.82

www.getsync.com
The ideal solution for easily syncing files, but closed source.

4th Owncloud

Licence AGPL v3 **Version** 8.0.2

https://owncloud.org
Feature-rich software that's probably overkill for simply syncing and sharing.

5th SparkleShare

Licence GPL v3 **Version** 1.4.0

www.sparkleshare.org
Loses out because of its geeky bent and lack of mobile clients.

6th Dropbox

Licence Proprietary **Version** 3.4.3

www.dropbox.com
This is 2015 and we know better.

	Self-hosted	Cloud server	Sync files	Share files	Back up data	Add-ons	Mobile client
OwnCloud	Y	N	Y	Y	Y	Y	Y
Seafile	Y	Y	Y	Y	Y	Y	Y
SparkleShare	Y	Y	Y	Y	N	N	N
BitTorrent Sync	Y	N	Y	N	N	N	Y
Dropbox	N	Y	Y	Y	N	N	Y
SpiderOak	N	Y	Y	Y	Y	N	Y

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.

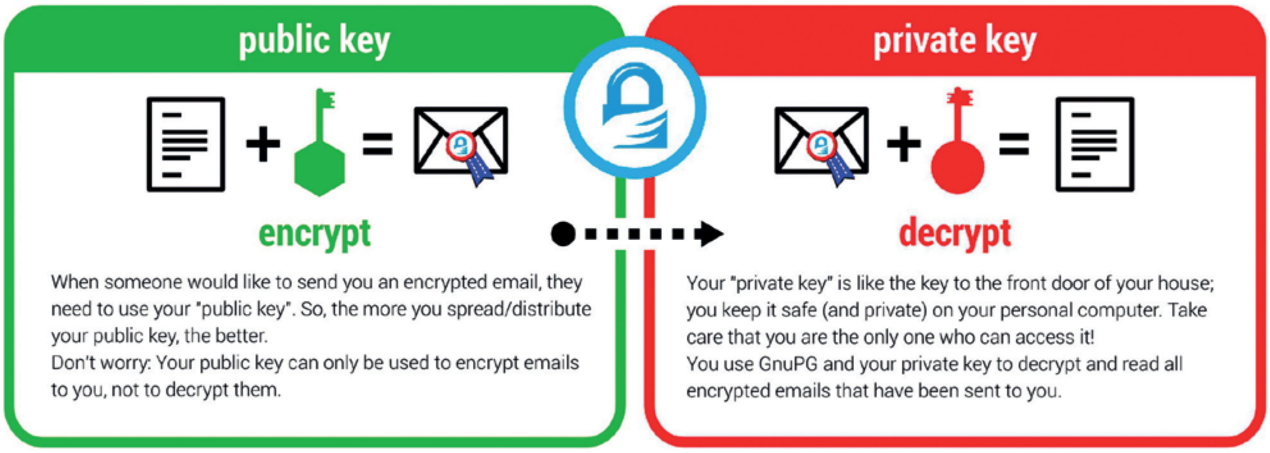
GnuPG is platform independent. That means it works with every email address and runs on pretty much any computer or recent mobile phone. GnuPG is free and available at no charge.

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?

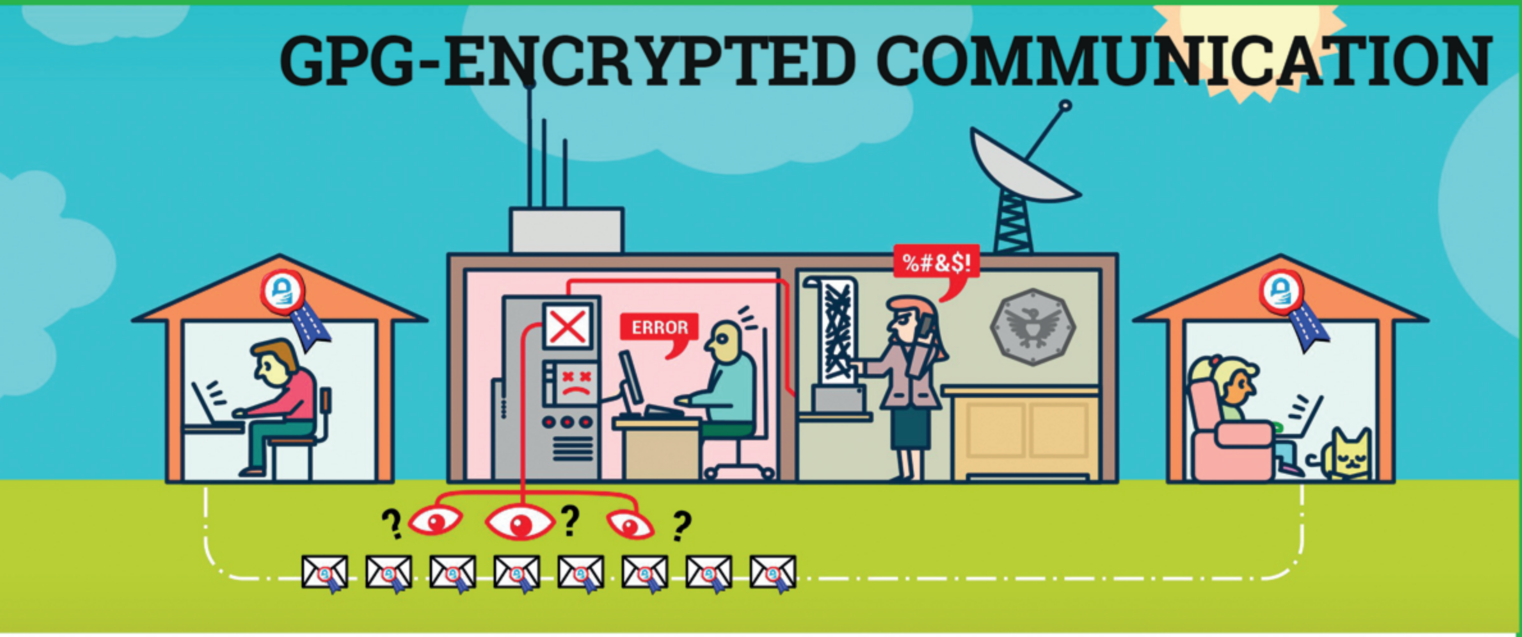
GnuPG is Free Software and uses Open Standards. That is essential to be sure that software can really protect us from surveillance. Because in proprietary software and formats, things might happen beyond your control.
If no one is allowed to see the source code of a program, nobody can be sure that it 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

SUBSCRIBE

shop.linuxvoice.com

Get your regular dose of Linux Voice, the magazine that:

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

UK subs prices

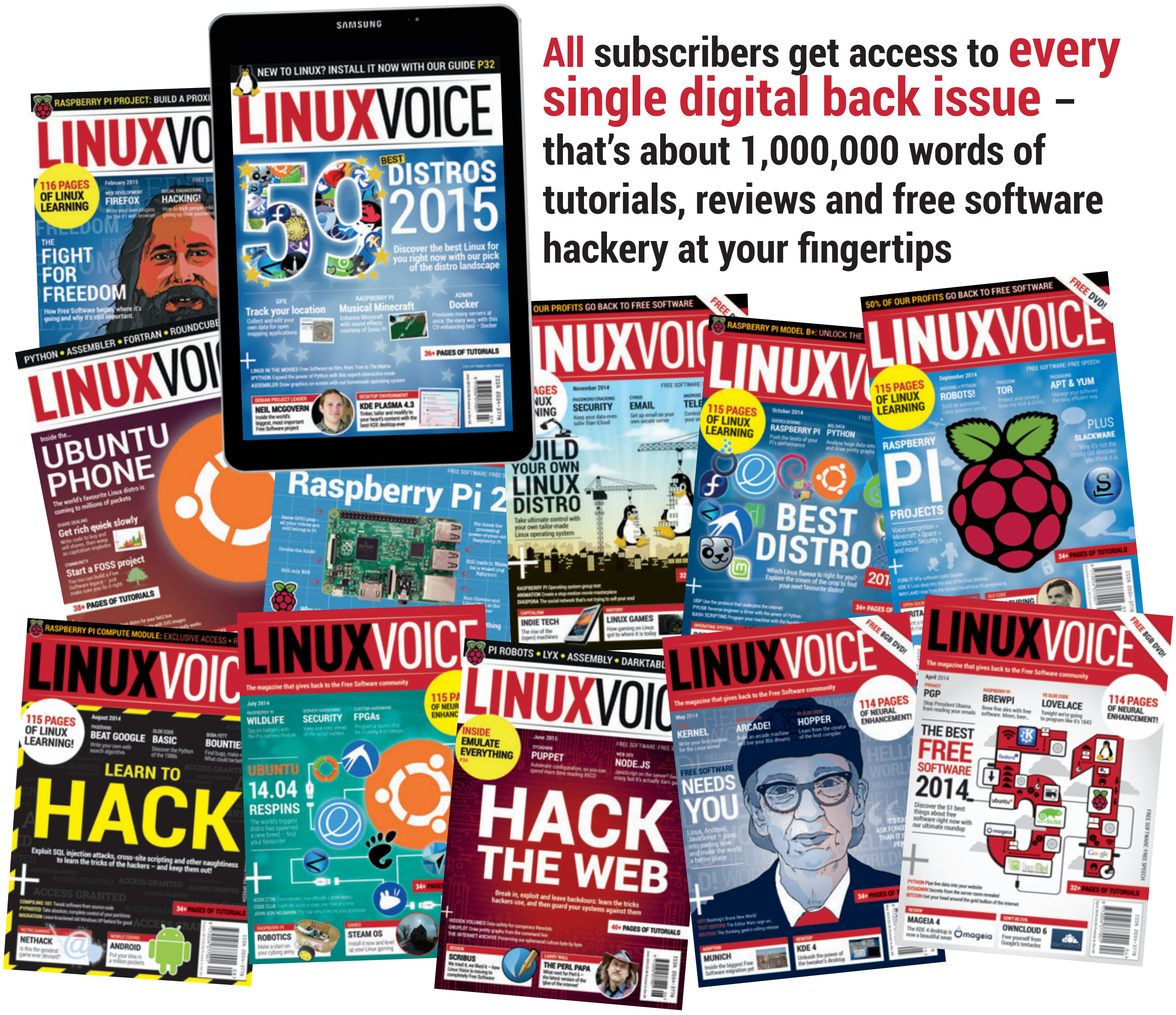
12-month print & digital: **£55**
12-month digital only: **£38**

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

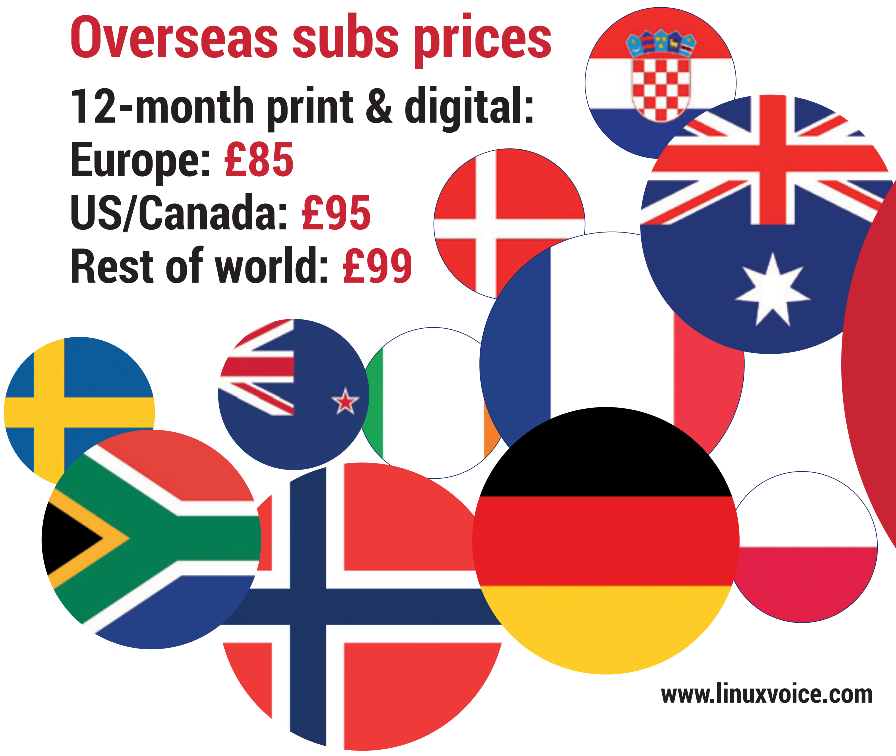
Payment is in Pounds Sterling. 12-month subscribers will receive 12 issues of Linux Voice a year. 7-month subscribers will receive 7 issue of Linux Voice. If you are dissatisfied in any way you can write to us to cancel your subscription at subscriptions@linuxvoice.com and we will refund you for all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



All subscribers get access to every single digital back issue – that's about 1,000,000 words of tutorials, reviews and free software hackery at your fingertips

Overseas subs prices

12-month print & digital:
Europe: £85
US/Canada: £95
Rest of world: £99



DIGITAL SUBSCRIPTION*

ONLY £38

*WHEREVER IN THE WORLD YOU ARE – IT'S DIGITAL, SO THERE ARE NO POSTAGE COSTS

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.

Asynchronous network programming

You know what a network socket is. Now, let's learn how to juggle with thousands of them.

Computers are meant to compute. But quite often programs just pump loads of data from here to there and back again without incurring much CPU load. This is known as an I/O-bound task, and there are several ways to do it in Linux. The simplest way is to block I/O. Traditional **read(2)** and **write(2)** system calls are blocking, and the socket samples from LV006's Core Tech (**www.linuxvoice.com/core-technology-6**) and LV007 were blocking as well.

In blocking I/O, the operating system multiplexes data coming from various sources for you. It's easy but not very scalable. To handle N simultaneous client connections, you'll need roughly N OS-level processes or threads (internally, both are just "tasks" in Linux). A typical server can handle up to several hundred threads. This may be enough already, and some services (say, *Asterisk*) happily follow this route. However, you won't build the next Twitter this way. And this is where asynchronous I/O comes into play.

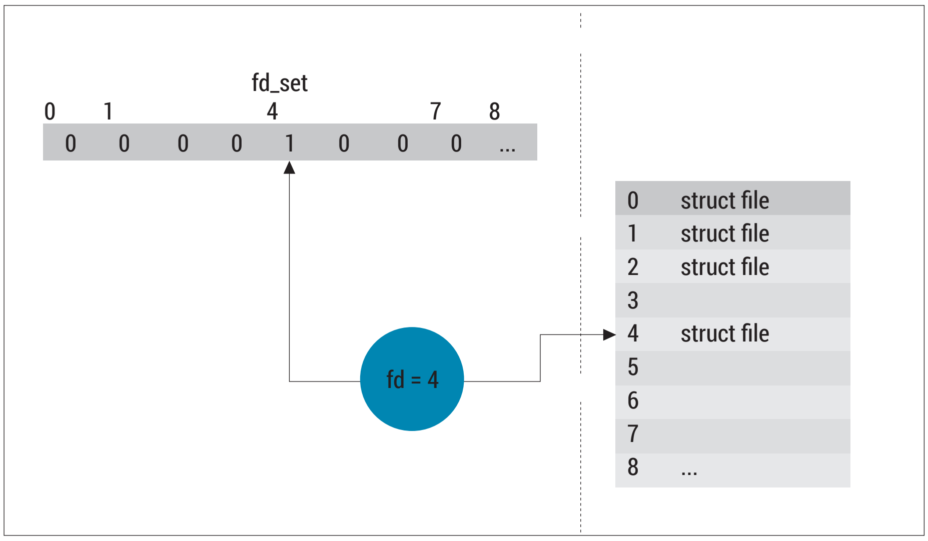
The idea is to do multiplexing yourself: while your code is waiting for A to reply, it can serve B and C. Asynchronous code is usually built around an event loop concept: first, it checks if there are any requests ready to read, and processes them. Then it checks if it can write any responses. Finally, if there are no more items to handle, the process is put to sleep until some event (or timeout) occurs. Unix systems provide various ways to do asynchronous I/O. They have different scalability characteristics, and here we'll cover three that are most important for Linux.

You are selected

The most traditional multiplexing mechanism is probably the **select(2)** system call. It's been present in Unix since 4.2BSD (circa 1983). **select(2)** receives three sets of file descriptors: the first one is monitored for read availability, the second for write availability, and the third is watched for errors. Each set is represented as a bitmask. When you call **select(2)**, it blocks until either event you are waiting for occurs or timeout fires, or the process is interrupted with a signal (as we looked at in LV011). With **pselect(2)** you have greater control over signals, but we won't cover this variation here.

A typical **select(2)**-based event loop may look like this:

```
#include <sys/select.h>
int main() {
 fd_set rfds;
 int fds[NUM_FD];
 struct timeval tv;
 int i, retval, max_fd = -1, done = 0;

 /* Fill fds[] with file descriptors */
 while (!done) {
  FD_ZERO(&rfds);
  max_fd = -1;
  for (i = 0; i < NUM_FD; i++) {
   FD_SET(fds[i], &rfds);
   if (fds[i] > max_fd)
    max_fd = fds[i];
  }
  tv.tv_sec = 3;
  tv.tv_usec = 0;

  retval = select(max_fd + 1, &rfds, NULL, NULL,
&tv);
  if (retval > 0) {
   for (i = 0; i < max_fd + 1; i++)
    if (FD_ISSET(i, &rfds)) {
     /* Data has come, handle it */
```



This is how **fd_set**, the file descriptor (**fd**) and **struct file** relate to one another. Bit 4 is set, so **select(2)** uses the struct file's **poll()** method to check if **fd 4** is ready

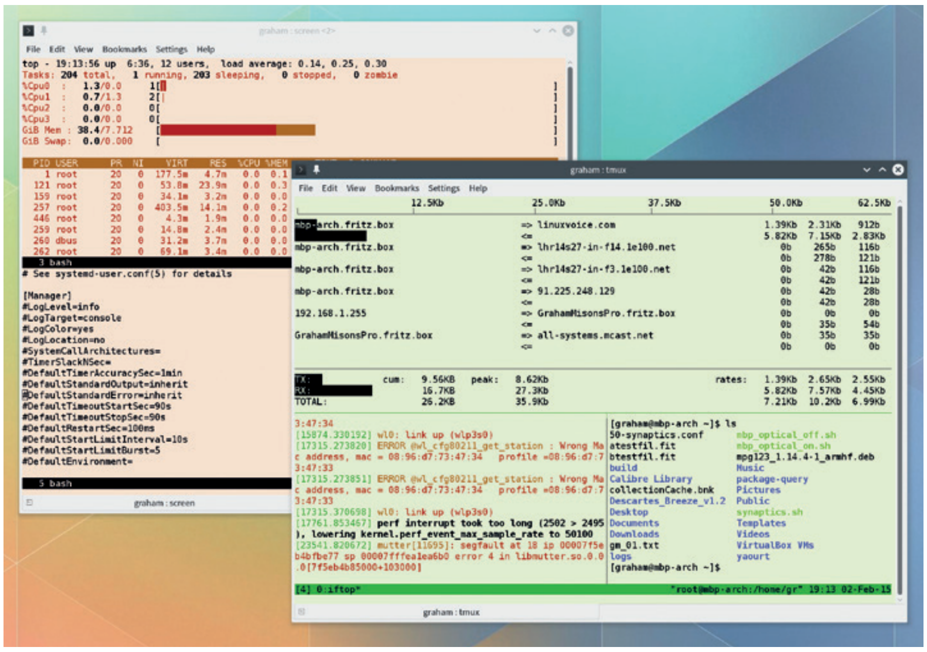
```
      }
  } else if (retval == 0) {
    /* Timeout happened */
  } else {
    /* Error occurred */
  }
 }
 return 0;
}
```

The lines beginning **FD_** are macros to deal with file descriptor sets. If a bit number N is raised (set to1) in **fd_set**, **select(2)** watches for the file descriptor number N.

We start with clearing the bitmask with **FD_ZERO()** and setting bits corresponding to **fds[i]** with **FD_SET()**. Then we call **select()**, which returns -1 on failure or number of ready file descriptors on success. In the latter case, we check for descriptors having events with **FD_ISSET()**, and handle data. Here, we focus solely on readability, but it is readily extensible to a real-world case. Returning a value of zero means timeout (**tv**, which is three seconds here). We also pass **select(2)**, the highest file descriptor number in the set. Note that we have to recreate arguments at every loop iteration as **select(2)** overwrites them, including **tv**.

select(2) is "level-triggered". This term is attributed to BSDCON 2000 presenter Jonathan Lemon, and comes from the design of hardware interrupts. It means that the system call will signal readiness for a file descriptor as long as it will be readable, not when it starts to be readable for the first time. The alternative design to this is known as "edge-triggered".

It's important to make file descriptors non-blocking with **fcntl(fds, F_SETFL, O_NONBLOCK)** or similar before starting the event loop. The reason is that readiness signalled from **select(2)** is only a hint from the kernel, not a strong guarantee. Another process can get in and read the data before we have the chance to do it, for instance. If you're interested in how the file descriptor can make its way into a separate process, see Core Technology in LV015. In this case, a call to **read(2)** will block a whole event loop. It's fatal, as **select(2)** won't be called again, and we'll miss events on all file descriptors. In other words, one client will ban all others, effectively making a DoS attack on our server.



```
File  Edit  View  Bookmarks  Settings  Help
top - 19:13:56 up  6:36, 12 users,  load average: 0.14, 0.25, 0.30
Tasks: 204 total,   1 running, 203 sleeping,   0 stopped,   0 zombie
%Cpu0  :   1.3/0.0     1[|                                        ]
%Cpu1  :   0.7/1.3     2[|                                        ]
%Cpu2  :   0.0/0.0     0[                                         ]
%Cpu3  :   0.0/0.0     0[                                         ]
GiB Mem : 38.4/7.712    [                                         ]
GiB Swap:  0.0/0.000    [                                         ]

  PID USER      PR  NI    VIRT    RES  %CPU %MEM
    1 root      20   0  177.5m   4.7m   0.0  0.1
  121 root      20   0   53.8m  23.9m   0.0  0.3
  159 root      20   0   34.1m   3.2m   0.0  0.0
  257 root      20   0  403.5m  14.1m   0.0  0.2
  446 root      20   0    4.3m   1.9m   0.0  0.0
  259 root      20   0   14.8m   2.4m   0.0  0.0
  260 dbus      20   0   31.2m   3.7m   0.0  0.0
  262 root      20   0   69.1m   3.4m   0.0  0.0
   3 bash
# See systemd-user.conf(5) for details

[Manager]
#LogLevel=info
#LogTarget=console
#LogColor=yes
#LogLocation=no
#SystemCallArchitectures=
#TimerSlackNSec=
#DefaultTimerAccuracySec=1min
#DefaultStandardOutput=inherit
#DefaultStandardError=inherit
#DefaultTimeoutStartSec=90s
#DefaultTimeoutStopSec=90s
#DefaultRestartSec=100ms
#DefaultStartLimitInterval=10s
#DefaultStartLimitBurst=5
#DefaultEnvironment=

   5 bash
```

```
File  Edit  View  Bookmarks  Settings  Help
           12.5Kb         25.0Kb          37.5Kb          50.0Kb       62.5Kb
mbp-arch.fritz.box                => linuxvoice.com                1.39Kb  2.31Kb   912b
[redacted]                        <=                               5.82Kb  7.15Kb  2.83Kb
mbp-arch.fritz.box                => lhr14s27-in-f14.1e100.net         0b    265b    116b
                                  <=                                   0b    278b    121b
mbp-arch.fritz.box                => lhr14s27-in-f3.1e100.net          0b     42b    116b
                                  <=                                   0b     42b    121b
mbp-arch.fritz.box                => 91.225.248.129                    0b     42b     28b
                                  <=                                   0b     42b     28b
192.168.1.255                     => GrahamMisonsPro.fritz.box         0b      0b      0b
                                  <=                                   0b     35b     54b
GrahamMisonsPro.fritz.box         => all-systems.mcast.net             0b     35b     35b
                                  <=                                   0b      0b      0b

TX: [redacted]   cum:   9.56KB   peak:   8.62Kb           rates:   1.39Kb  2.65Kb  2.55Kb
RX: [redacted]          16.7KB           27.3Kb                    5.82Kb  7.57Kb  4.45Kb
TOTAL:                  26.2KB           35.9Kb                    7.21Kb  10.2Kb  6.99Kb
3:47:34                                         [graham@mbp-arch ~]$ ls
[15874.330192] wl0: link up (wlp3s0)            50-synaptics.conf      mbp_optical_off.sh
[17315.273820] ERROR @wl_cfg80211_get_station : Wrong Ma atestfil.fit  mbp_optical_on.sh
c address, mac = 08:96:d7:73:47:34   profile =08:96:d7:7 btestfil.fit  mpg123_1.14.4-1_armhf.deb
3:47:33                                         build                  Music
[17315.273851] ERROR @wl_cfg80211_get_station : Wrong Ma Calibre Library  package-query
c address, mac = 08:96:d7:73:47:34   profile =08:96:d7:7 collectionCache.bnk  Pictures
3:47:33                                         Descartes_Breeze_v1.2  Public
[17315.370698] wl0: link up (wlp3s0)            Desktop                synaptics.sh
[17761.853467] perf interrupt took too long (2502 > 2495 Documents     Templates
), lowering kernel.perf_event_max_sample_rate to 50100 Downloads       Videos
[23541.820672] mutter[11695]: segfault at 18 ip 00007f5e gm_01.txt     VirtualBox VMs
b4bfbe77 sp 00007fffea1ea6b0 error 4 in libmutter.so.0.0 logs          yaourt
.0[7f5eb4b85000+103000]                         [graham@mbp-arch ~]$
[4] 0:iftop*                                    "root@mbp-arch:/home/gr" 19:13 02-Feb-15
```

Both *tmux* and *screen* are similar, but the more recent *tmux* uses **poll(2)** via **libevent**, while the older *screen* relies on **select(2)**.

Flexible fishing

select(2) has one limitation you've probably already spotted. Bitmasks are not very flexible data structures, and they have a hard-coded maximum length (typically 1024 bits). This means file descriptors having numbers of 1024 and greater can't be watched, which may pose problems to long-running daemon processes. Moreover, **fd_set** can't have any gaps. So, if you're interested in a single file descriptor number 1023, you waste more than thousand bits. It's not only memory space that's wasted, however. **select(2)** will also need to loop through the whole mask just to see that all bits but the last one are not set. This is not good for performance either.

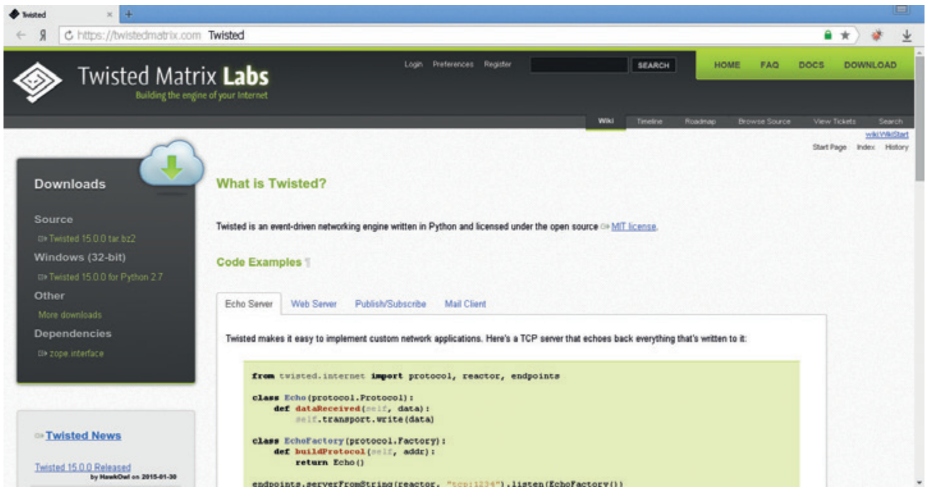
poll(2) is a more flexible alternative. It's younger than **select(2)** so you can come across veteran Unix system that lacks it. For Linux, it doesn't matter: both **poll(2)** and **select(2)** are supported, and both are in POSIX 1003.1g standard. Unlike **select(2)**, **poll(2)** accepts an array of **struct pollfd** items that can be as small or as large as you need. Without initialisation, the rest of the usage is similar:

```
#include <poll.h>
int main(int *fds) {
 struct pollfd pfds[NUM_FD];
 int i, retval, done = 0;
 int timeout_msec = 3000;
 int fds[NUM_FD];

 /* Fill fds[] with file descriptors */
 for (i = 0; i < NUM_FD; i++) {
   pfds[i].fd = fds[i];
   pfds[i].events = POLLIN;
 }

 while (!done) {
   retval = poll(pfds, NUM_FD, timeout_msec);
   if (ret_val > 0) {
     for (i = 0; i < NUM_FD; i++) {
       if (pfds[i].revents & POLLIN) {
         /* Data has come, handle it */
             }
   } else if (retval == 0) {
     /* Timeout happened */
   } else {
     /* Error occurred */
   }
 }
 return 0;
}
```

Flying higher

Being system calls, **select(2)**, **poll(2)** and **epoll(7)** are among the lowest-level mechanisms you can get in Linux. The last one is also OS-dependent: it won't work in FreeBSD or Solaris, not to mention Microsoft Windows.

If you think it would be good to have some way to abstract these differences and low-level complexity, you're not alone. There are several C/C++ libraries that do this job, but **libevent** and **libev** are arguably most popular. Both provide a platform-independent event loop API that is callback-based. You assign C functions that act as callbacks to events, you start the event loop, and the library does the rest. This way, you get all the benefits of asynchronous I/O without having to bother with platform details.



What is Twisted?

Twisted is an event-driven networking engine written in Python and licensed under the open source MIT license.

Code Examples

Twisted makes it easy to implement custom network applications. Here's a TCP server that echoes back everything that's written to it:

```
from twisted.internet import protocol, reactor, endpoints

class Echo(protocol.Protocol):
    def dataReceived(self, data):
        self.transport.write(data)

class EchoFactory(protocol.Factory):
    def buildProtocol(self, addr):
        return Echo()

endpoints.serverFromString(reactor, "tcp:1234").listen(EchoFactory())
```

Twisted is an old but very powerful way to do asynchronous network programming in Python.

Each **struct pollfd** contains the file descriptor (**fd**) to watch and the bitmask of events to wait for (**events**). Events are described with flags like **POLLIN** (data can be read), **POLLOUT** (the descriptor is writable) or **POLLERR** (error occurs). On output, **revents** will contain the events occurred. As with **select(2)**, on success the number of file descriptors ready is returned. Zero means timeout and -1 signifies an error (**errno** stores the details, as usual). The timeout value is passed to **poll(2)** as a plain number of milliseconds; -1 means infinity and 0 forces **poll(2)** to return immediately. This is not what you usually want, but it's useful for "peeking", event loop integration and other hardcore stuff.

Having a flexible array instead of a fixed-size bitmask is certainly a step forward. However, both **select(2)** and **poll(2)** suffer the same scalability issue. Namely, their time complexity is about 'O(N)' where N is the number of file descriptors watched. This means that the time needed to **poll()** a file descriptor set is proportional to its size. It's a challenge for high-loaded services, known as the "C10k problem", and you can find a good overview of it here: **www.kegel.com/c10k.html**.

Down to the kernel

To understand why **select(2)** and **poll(2)** behave this way, we'll need to learn how they are implemented at kernel level. In Linux, each open file has a corresponding **struct file** instance in kernel space, and a file descriptor is basically an index into an array of these objects. This structure provides file operations (**f_op**) whose actual implementation depend on whether the descriptor in question is for a socket, a pipe, an on-disk file or anything else. One of these operations is **poll()**: usually, it adds a calling process to the wait queue and returns the bitmask representing the current **struct file** state.

select(2) and **poll(2)** call **f_op->poll()** for each descriptor in the set, and return if there is anything ready. Otherwise, the process is put to sleep (**syscall** blocks) until another part of the kernel (like the network subsystem in the case of a socket) detects activity on the file or a timeout elapses. When the process wakes up, the **syscall** resumes and once again loops over the entire descriptor set checking for events, as it has no idea what caused the wakeup. If there were any events the caller wants to know, control is returned to userspace. Otherwise the process sleeps again. As each file descriptor takes about the same time to handle, the overall loop time grows linearly with the descriptor set's size.

In high-loaded services, 'O(N)' behaviour is not what you want, as it puts a limit on how many concurrent connections you can handle. Another mechanism is required, and it comes in many OS-specific forms. FreeBSD has **kqueue**, Solaris offers **/dev/poll**. In Linux, we have **epoll(7)**. Originally, it was also implemented as **/dev/epoll** in the Solaris spirit, but Linus banned the idea so it was rewritten as a set of system calls, similar to FreeBSD's. The best thing about **epoll(7)** is that it allows for 'O(1)'s performance. This means it runs roughly the same length of time regardless what the size of descriptor set is. We'll show you some examples shortly, but let's discuss how **epoll(7)** achieves this result first.

The key to **epoll(7)**'s scalability is clever use of callbacks and a bit of indirection. Wait queues in the kernel have an associated callback function, which is called when the process is resumed. For **poll(2)** and **select(2)** it's the same and quite simple, but **epoll(7)** registers its own sophisticated callback. The process fishing for events with **epoll(7)** is added to two wait queues. One is per-file descriptor watched, like in **poll(2)** or **select(2)**, and another is per epoll instance. When the kernel detects activity on a watched file descriptor, it calls **ep_poll_callback()**, which adds the descriptor to a so-called "ready-list" and wakes up the process sleeping on the epoll instance queue. Then the ready-list is scanned, and if there is anything of interest to userspace, the syscall returns. Otherwise, the process is put to sleep again.

Note that this never requires the kernel code to loop over the whole file descriptor set. It only scans the ready-list, which requires **O(NumReady)** steps, with **NumReady** being the number of descriptors ready. Under the assumption that it doesn't depend on how many descriptors you watch (which sounds reasonable), this yields 'O(1)' performance. **select(2)** and **poll(2)** are just multiplexers, whereas **epoll(7)** is event-driven. Moreover, in **epoll(7)**, the in-kernel representation of the descriptor set and related data structures are preserved across system calls, thus reducing setup and teardown costs per each call.

Event loop revisited

Now, when we understand how the whole thing works, let's learn how to use it. Compared to **poll(2)** or **select(2)**, **epoll(7)** is not a single system call but a set of three. The first one, **epoll_create(2)**, creates an **epoll** file descriptor (**epoll_fd**), which is basically a handle for the in-kernel **epoll** instance. When the last process closes **epoll_fd**, the **epoll** instance is also freed. Then, **epoll_ctl(2)** is the way to add and remove file descriptors or otherwise modify the file descriptor set. Finally, **epoll_wait(2)** is what you call to fish for events.

This is what a typical **epoll(7)**-based event loop (without most of the error checking) may look like:

```

#include <sys/epoll.h>
#include <unistd.h>
int do_epoll(int *fds) {
 struct epoll_event ev, events[NUM_FD];
 int i, retval, done = 0;
 int timeout_msec = 3000;
 int fds[NUM_FD];
 int epoll_fd;

 /* Fill fds[] with file descriptors */
```

```
epoll_fd = epoll_create(NUM_FD);

for (i = 0; i < NUM_FD; i++) {
  ev.events = EPOLLIN | EPOLLET;
  ev.data.fd = fds[i];
  epoll_ctl(epoll_fd, EPOLL_CTL_ADD, fds[i], &ev);
}

while (!done) {
  retval = epoll_wait(epoll_fd, events, NUM_FD,
timeout_msec);
  if (retval > 0) {
    for (i = 0; i < retval; i++) {
                if (events[i].events & EPOLLIN) {
                  /* Data has come, handle it */
                }
    } else if (retval == 0) {
      /* Timeout happened */
    } else {
                /* Error occurred */
    }
}
 close(epoll_fd);
 return 0;
}
```

The only argument **epoll_create(2)** receives used to be a hint for descriptor set size. Modern kernels ignore it (albeit it must be non-zero for backward compatibility), and there is newer **epoll_create1(2)** that doesn't take this argument. Events are represented with **struct epoll_event**, which again stores a file descriptor to watch and the events mask, as in **poll(2)**. You add these structures to **epoll_fd** with **epoll_ctl(2)**. Other possible arguments include **EPOLL_CTL_DEL** to remove descriptors, and **EPOLL_CTL_MOD** to change the events mask.

Most event mask flags are quite descriptive. For instance, **EPOLLIN** is for readability test, and **EPOLLOUT** is for writeability. Note the use of **EPOLLET**, however. As the example shows, it can be combined with other flags. **ET** stands for "edge-triggered", which is another distinctive feature of **epoll(7)**. **poll(2)** and **select(2)** are always level-triggered, meaning that they will signal the event as long as the condition holds – if, for example, there is data to read. With **EPOLLET**, only events that change the descriptor state (eg from empty to readable) are signalled. It may be tempting to use it for new data indication, and it will work. However, you should always keep in mind that the event is signalled on a state change. If 5k of data arrived in a socket, and you read only 3k of it before calling **epoll_wait(2)** again, you'll lose the event for the next 5k of data coming. That's yet another reason to set file descriptors to non-blocking state, so you can drain the descriptor safely.

epoll_fd is itself pollable. You can add it to the file descriptor set, and it will become readable if there are any events signalled. It's not a good idea to add an **epoll** file descriptor to its own set, however.

While **select(2)**, **poll(2)**, and **epoll(7)** work well on sockets or pipes, you shouldn't assume the same for regular files. Carefully use non-blocking disk access, yielding an event loop when necessary, or try POSIX Asynchronous I/O here.

Isn't everything a file already?

If all you have is a hammer, everything looks like a nail. If your program is built around an event loop, it's very convenient to have every input as a pollable file descriptor.

Luckily, almost everything in Linux is already a file. Signals are notable exceptions here, but Linux has several mechanisms that can come to the rescue. They appeared first in kernel 2.6.22, and are very similar in their operation. You are given a way to obtain a special file descriptor that becomes readable when some event occurs. This can be a signal delivered to your process, a timer expiration or some other event that needs your attention. You can think of these mechanisms as coming from the same family, as they share a common suffix: **fd**. Their names are **signalfd(2)**, **timerfd**, and **eventfd(2)**. The first one delivers Unix signals, the second is for timers and the third is suitable for posting application-level events or as an IPC mechanism (see LV015). It can also serve as an event channel between userspace and the kernel. For instance, it is the mechanism *QEMU* and *KVM* use for bi-directional notifications. It is also employed in high-speed kernel-space device emulation (the VHost and VFIO subsystems).

Consult the corresponding man pages for details and usage examples. For **timerfd**, try **man 2 timerfd_create**.

On the Python side

The code samples we've seen so far used C, but that doesn't mean you can't write asynchronous code in a high-level language such as Python. In this case it is unlikely you'll do system calls directly, however. Most probably, you'll rely on some asynchronous framework or networking library, like Gevent, Tornado or Twisted. Let's have a sneak peek on what happens behind the curtains here, taking the latter as an example.

Twisted is a callback-based framework. When asynchronous operation (say, socket read) starts, you get a 'Deferred' object that you can attach your callbacks to. Those are called when the data is ready.

Twisted can preserve an illusion of synchronous code execution. It relies on co-routines, the **yield** keyword and a custom decorator (**@defer.inlineCallbacks**). Instead of attaching callbacks yourself, you 'yield' the 'Deferred', and the decorator does this for you. When the value is ready, the decorator takes it and sends it back to the **yield** expression, or injects an exception in case of error. All this runs in a loop which terminates when the wrapped function returns a value or simply ends.

Command of the month: strace

This month, we've learned several ways to create event loops in a Linux application. How do you know which one the given program uses?

Try **strace(1)**, a small tool that does a great job tracing system calls live. It requires root permissions to run, because snooping on another user's processes is certainly a privileged operation. You can trace a process you start yourself with **strace program_name**, but more importantly, you can also connect to the process already running with **strace -p PID**. You can also add **-e desc** to filter out everything but file descriptor-related system calls.

Consider the following:

```
$ sudo strace -p 3492 -e desc -t
11:35:56 poll([{fd=5, events=POLLIN}, {fd=6,
events=POLLIN}, {fd=3, events=POLLIN}], 3, 1065) =
1 ([{fd=6, revents=POLLIN}])
11:35:57 poll([{fd=5, events=POLLIN}, {fd=6,
events=POLLIN}, {fd=3, events=POLLIN}], 3, 145) = 0
(Timeout)
```

Here I traced the **update-notifier** process on an Ubuntu 14.04 LTS system, and I also added **-t** for timestamps. We see **update-notifier** relies on **poll(2)**, and you can also note how **strace(1)** decodes the arguments and return values for you. We'll come back to **strace(1)** next issue.

FOSSpicks

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

Hunting snarks is for amateurs – **Ben Everard** spends his time in the long grass, stalking the hottest, free-est Linux software around.

Microkernel

Hurd

There are two different types of code running on a computer: kernel space code and userspace code. Code in the kernel space is held in memory and has full access to the hardware, while code in userspace is controlled by the kernel and can access the hardware by calling procedures in the kernel. However, there's also a large grey area where code could live either in kernel space or userspace – filesystem drivers, for example. Most operating systems have these in the kernel; however, there's no reason they can't live in userspace.

Linux, BSDs, and almost all modern operating systems have a lot of systems functionality in the kernel. Theoretically, there may be an advantage to shifting this out into userspace: doing so would keep the kernel smaller and could increase security. These minimalist kernels are known as microkernels, and Mach is one of these.

Hurd is a set of servers and related protocols that run on top of the Mach microkernel to enable userspace software to run everything that's needed.

The easiest way to try Hurd is to download a *Qemu* image file from **https://people.debian.org/~sthibault/hurd-i386**. Once unzipped, this can be run with:

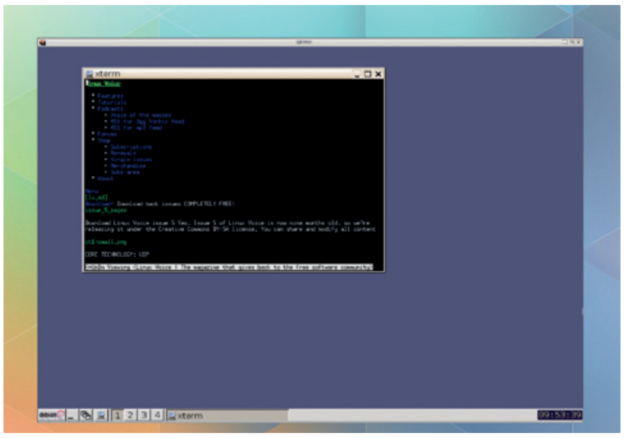
```
qemu-system-x86_64 -m 1024 debian-hurd-20150105.img
```

A work in progress...

At the time of writing, there isn't a build using the latest 0.6 kernel, but this may change by the time you read this. This is built on Debian, but with Hurd and Mach instead of the Linux kernel. There aren't as many packages as in the Linux version of Debian, because some software requires features that aren't yet in Hurd. If you're a glutton for punishment, you can build a system yourself from the instructions at **www.gnu.org/software/hurd/hurd/running/qemu.html**.

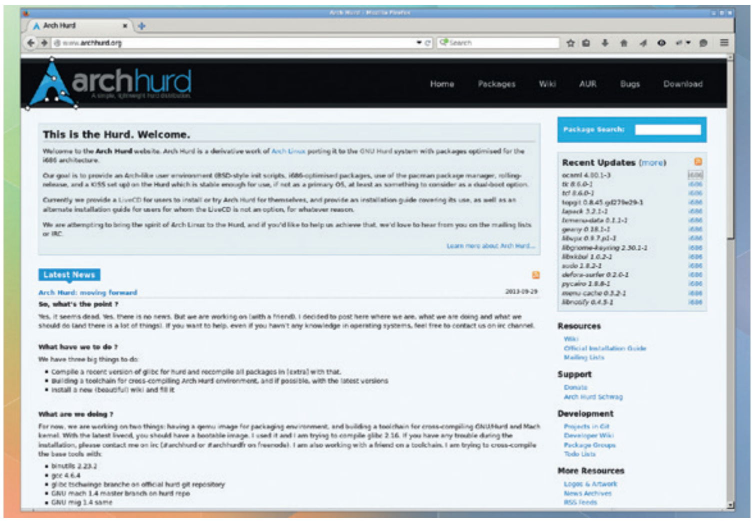
Mach and Hurd have been in development for 25 years, but it's not particularly stable, and there's limited hardware support. Hurd simply doesn't have enough developers to realise its vision in the short term, a problem that's exacerbated by the complexity of the microkernel model.

There are versions of Debian and Arch that use Hurd and Mach in place of the Linux kernel, though without a mature kernel, and a software stack designed to take advantages of the specific features that a microkernel would give, it's hard to realise the potential of Hurd. For the time being, Hurd is an interesting idea, but it's not yet a serious kernel for real work. Of course, there was a time when the same could be said of Linux...



```
Linux Voice

 * Features
 * Tutorials
 * Podcasts
    * Voice of the masses
    * RSS for Ogg Vorbis feed
    * RSS for mp3 feed
 * Forums
 * Shop
    * Subscriptions
    * Renewals
    * Single issues
    * Merchandise
    * Subs area
 * About

Menu
[lv_ad]
Download! Download back issues COMPLETELY FREE!
issue_5_pages

Download Linux Voice issue 5 Yes, Issue 5 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

ct1-small.png

CORE TECHNOLOGY: UDP

Viewing <Linux Voice | The magazine that gives back to the free software community>
```

The Debian Hurd image doesn't come with much by default, but it does have *IceWM* and the *w3m* text-based web browser.



This is the Hurd. Welcome.

Welcome to the **Arch Hurd** website. Arch Hurd is a derivative work of Arch Linux porting it to the GNU Hurd system with packages optimised for the i686 architecture.

Our goal is to provide an Arch-like user environment (BSD-style init scripts, i686-optimised packages, use of the pacman package manager, rolling-release, and a KISS set up) on the Hurd which is stable enough for use, if not as a primary OS, at least as something to consider as a dual-boot option.

Currently we provide a LiveCD for users to install or try Arch Hurd for themselves, and provide an installation guide covering its use, as well as an alternate installation guide for users for whom the LiveCD is not an option, for whatever reason.

We are attempting to bring the spirit of Arch Linux to the Hurd, and if you'd like to help us achieve that, we'd love to hear from you on the mailing lists or IRC.

Learn more about Arch Hurd...

Latest News

Arch Hurd: moving forward 2013-09-29

So, what's the point ?

Yes, it seems dead. Yes, there is no news. But we are working on (with a friend). I decided to post here where we are, what we are doing and what we should do (and there is a lot of things). If you want to help, even if you haven't any knowledge in operating systems, feel free to contact us on irc channel.

What have we to do ?

We have three big things to do:

- Compile a recent version of glibc for hurd and recompile all packages in [extra] with that.
- Building a toolchain for cross-compiling Arch Hurd environment, and if possible, with the latest versions
- Install a new (beautiful) wiki and fill it

What are we doing ?

For now, we are working on two things: having a qemu image for packaging environment, and building a toolchain for cross-compiling GNU/Hurd and Mach kernel. With the latest livecd, you should have a bootable image. I used it and I am trying to compile glibc 2.16. If you have any trouble during the installation, please contact me on irc (#archhurd or #archhurdfr on freenode). I am also working with a friend on a toolchain. I am trying to cross-compile the base tools with:

- binutils 2.23.2
- gcc 4.6.4
- glibc tschwinge branche on official hurd git repository
- GNU mach 1.4 master branch on hurd repo
- GNU mig 1.4 same

Recent Updates (more)

ocaml 4.00.1-3	i686
tk 8.6.0-1	i686
tcl 8.6.0-1	i686
topgit 0.8.45.gd279e29-1	i686
lapack 3.2.1-2	i686
lxmenu-data 0.1.1-1	i686
geany 0.18.1-1	i686
libvpx 0.9.7.p1-1	i686
libgnome-keyring 2.30.1-1	i686
libxklavier 1.0.2-1	i686
sudo 1.8.2-1	i686
defora-surfer 0.2.0-1	i686
pycairo 1.8.8-1	i686
menu cache 0.3.2-1	i686
libnotify 0.4.5-1	i686

Resources
Wiki
Official Installation Guide
Mailing Lists

Support
Donate
Arch Hurd Schwag

Development
Projects in Git
Developer Wiki
Package Groups
Todo Lists

More Resources
Logos & Artwork
News Archives
RSS Feeds

Arch Hurd doesn't get much development, but the most recent update on the website assures us that the project isn't dead.

PROJECT WEBSITE
www.gnu.org/software/hurd/index.html

Video editor

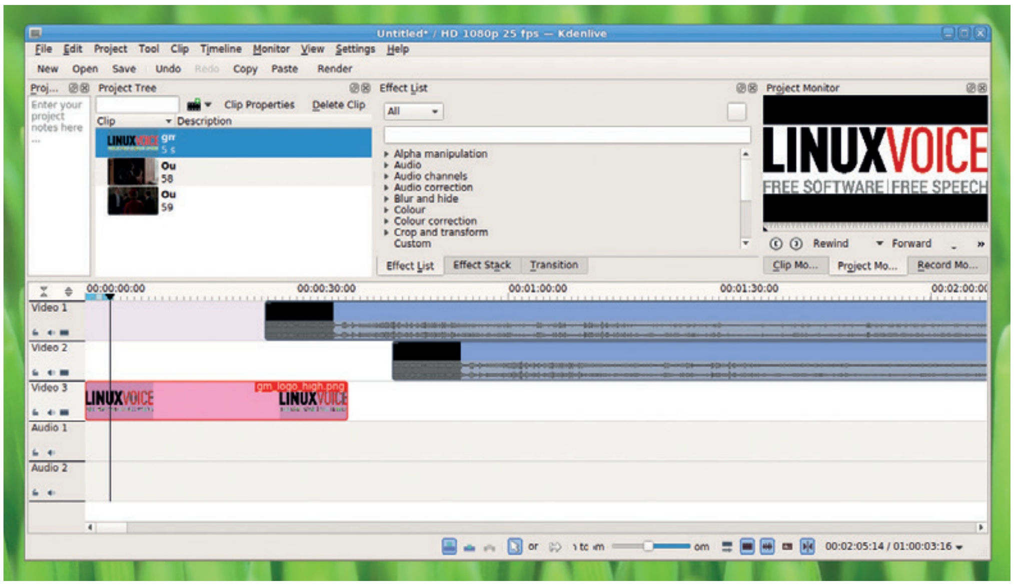
Kdenlive

Once upon a time, video production was a specialist activity which only a few highly skilled individuals ever dared to try, but now, with YouTube, Vine and other user-generated video sites, it's becoming more common for ordinary people to produce videos for themselves. *Kdenlive* is a video editor for the masses.

Kdenlive is a non-linear video editor for KDE. The non-linear part of the name means that it's not destructive of the source material (almost all digital video editors are non-linear).

Version 15.04 is the first release of *Kdenlive* since the project became an official part of KDE. This change should mean that we see more frequent updates than previously, as it will keep pace with the official KDE release schedule. It also means that the software's switched to using the *Qt 5* graphical toolkit, so it might be quite some time before the latest build makes it into the more conservative distros, but users of Arch and other cutting-edge distros should already have it.

The interface is nicely laid out, with the default layout giving you tracks at the bottom and the clips, effects and preview in a line across the top. This gives you access to what you need without over-complicating the window.


Untitled* / HD 1080p 25 fps — Kdenlive
File Edit Project Tool Clip Timeline Monitor View Settings Help
New Open Save Undo Redo Copy Paste Render
Project Tree
Clip Properties Delete Clip
Effect List
Alpha manipulation
Audio
Audio channels
Blur and hide
Colour
Colour correction
Crop and transform
Custom
Effect List Effect Stack Transition
Project Monitor
LINUXVOICE
FREE SOFTWARE | FREE SPEECH
Rewind Forward
Video 1
Video 2
Video 3
Audio 1
Audio 2
00:02:05:14 / 01:00:03:16

The *Kdenlive* interface is welcoming for beginners, but doesn't hide anything away from the user.

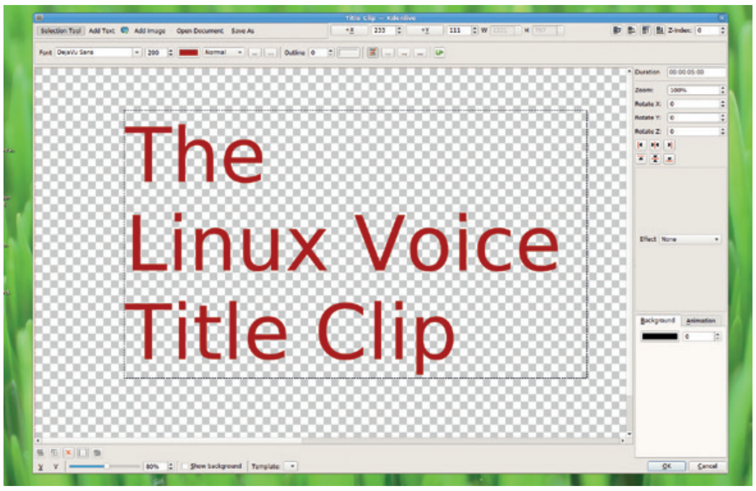
Kdenlive can use either *FFmpeg* or *Libav* to convert tracks as they're imported or exported, so it supports a huge range of video file formats. In its basic usage, *Kdenlive* enables you to take a series of video and audio tracks and combine them to make a whole video. Each clip can be as short as a single frame or as long as you like, and you can merge between them with a range of transitions. As well as merging tracks, you can add various effects, such as colour transformations or spatial distortions.

"Kdenlive enables you to combine a series of video and audio tracks to make a whole video."

For many years, we've been put off *Kdenlive* because it hasn't been stable. In fact, once we had to abandon a project because we just couldn't get it to work with the video files we needed. However, the last few releases have seen a noticeable rise in stability, and it's been a while since it's crashed on us.

The last couple of years have seen a considerable amount of interest in video editing on Linux, and people are putting money into creating better software. *Kdenlive* raised almost $7,500 in an Indiegogo crowdfunding campaign, while more recently *Pitivi* (a competing open source video editor) raised over €23,000.

We're starting to see the fruits of this cash injection, and the standard of all tools in the field is rising. Perhaps we'll get to a day when the quality of tools for video production on Linux is as good as the quality of the developer tools.



Kdenlive includes a title clip editor for creating simple text overlays for your videos.

PROJECT WEBSITE
https://kdenlive.org/

Private file sharer

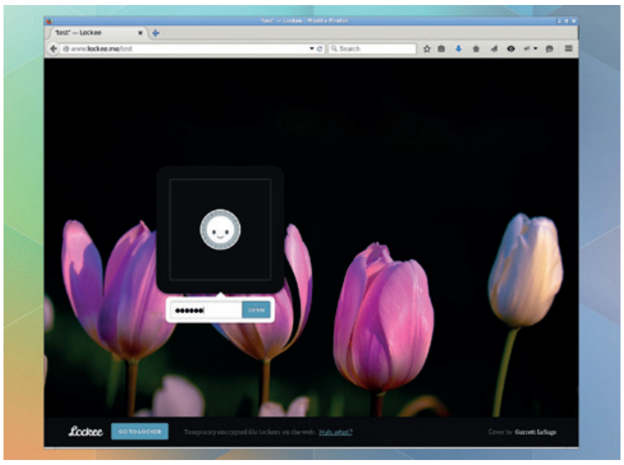
Lockee

Have you ever tried to share a large file with someone and found that none of the existing solutions really worked? Email just won't handle it. There are some cloud-based options (Google Drive and Dropbox to name but two), but these require you to hand your data over to organisations that are known to data-mine for profit, so may not be suitable for private data. There are other solutions that provide more security, but require the person you're sharing the data with to install software, and that's not always possible.

In other words, there are lots of ways you can share large files over the internet, but none of them work well. Enter *Lockee*. This solves the problem by providing temporary encrypted online storage. You can use the public instance at **www.lockee.me**: just think up a name for the shared file (known as a storage locker), and create a password. Your files will be encrypted on your device, and uploaded to a locker. You can share a link with anyone, give them the password, and they can download the contents of that locker, and after 24 hours, the files are deleted.

Encryption is done client-side using the Stanford JavaScript Crypto Library using 512-bit AES. This should be sufficient for most security needs, but we can't guarantee that there aren't any weaknesses in it.

The server-side software is all open source, so you can deploy it to your own server if you wish to tinker with any of the settings (such as the amount of time files are held for before they're deleted). It's based on node.js, and can be installed with the *npm* package manager, so you can your own private file sharing running in just a few minutes.



The web interface is simple and easy to use, because *Lockee* focuses on doing just one thing, and doing it well.

"Lockee encrypts your files and uploads them to a locker."

PROJECT WEBSITE
www.lockee.me

Backup tool

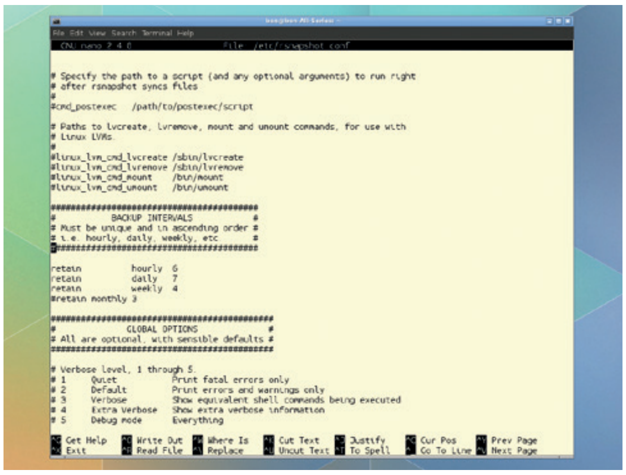
Rsnapshot

Rsync is arguably the best tool for making copies of filesystems. It's almost endlessly configurable, but as a consequence of this power, it can also be complex to fine-tune. *Rsnapshot* is a set of shell scripts that wrap around *Rsync* to provide easy access to some of its most useful backup features.

As well as providing the ability to create a snapshot, *Rsnapshot* lets you create incremental backups based on this snapshot. With clever use of hardlinks, this means you can store the state of the backed-up directory at any point of time with very little extra space. It maintains the permissions of the original files, so if you're running locally, the owners of the files can restore them without the root user having to get involved.

Since *Rsnapshot* uses *Rsync* and SSH to do the backups, you can save files from a server with just these two utilities installed, so there's no need to put extra software on the machine you want to back up. You just need to install *Rsnapshot* onto the machine that will save the backups, and you can set it running.

There are lots of wrappers around *Rsync*. *Rsnapshot*'s main selling point is its simplicity, and the fact that it's command-line based, so it can be run from *Cron* (or *Systemd* timers if you're cutting-edge). Once it's set up, it will just continue running, meaning you never need to worry about your backups. Just install it and forget about it. Then, should the unfortunate happen, just browse through the backups (which are just a clone of the backed-up directories), and restore anything you need.



```
  GNU nano 2.4.0                File: /etc/rsnapshot.conf


# Specify the path to a script (and any optional arguments) to run right
# after rsnapshot syncs files
#
#cmd_postexec   /path/to/postexec/script

# Paths to lvcreate, lvremove, mount and umount commands, for use with
# Linux LVMs.
#
#linux_lvm_cmd_lvcreate /sbin/lvcreate
#linux_lvm_cmd_lvremove /sbin/lvremove
#linux_lvm_cmd_mount    /bin/mount
#linux_lvm_cmd_umount   /bin/umount

#########################################
#           BACKUP INTERVALS            #
# Must be unique and in ascending order #
# i.e. hourly, daily, weekly, etc.      #
#########################################

retain          hourly  6
retain          daily   7
retain          weekly  4
#retain monthly 3

###########################################
#             GLOBAL OPTIONS              #
# All are optional, with sensible defaults #
###########################################

# Verbose level, 1 through 5.
# 1     Quiet           Print fatal errors only
# 2     Default         Print errors and warnings only
# 3     Verbose         Show equivalent shell commands being executed
# 4     Extra Verbose   Show extra verbose information
# 5     Debug mode      Everything

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text   ^J Justify   ^C Cur Pos     ^Y Prev Page
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line  ^V Next Page
```

Rsnapshot is configured by the file **/etc/rsnapshot.conf**. Using this, you can tune its behaviour however you want.

"Rsnapshot is a set of shell scripts that wrap around the Rsync utility."

PROJECT WEBSITE
http://rsnapshot.org

eBook editor

Sigil

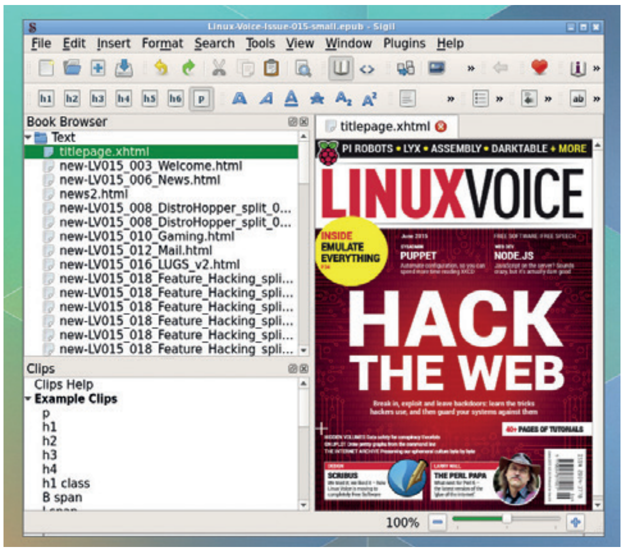
EPub, the most popular format of eBooks, is a format that uses Zip files to bring together HTML documents and associated content (CSS, images, etc) into a single file that can be saved to a device. This format means you can use standard web technologies, but still read the contents offline. Since ePubs use HTML, you can create them with any HTML authoring tool, however, a specialist ePub creator makes it easier to bring everything together into a single file.

Sigil has everything you need to create your own ePub files. These could be ones your want to share with your friends, or your groundbreaking first novel that you want to publish online (*Sigil* can create files for Google Play Books, and *Sigil*'s ePubs can be converted into the appropriate format for Amazon's Kindle Store using the *Calibre* eBook manager).

You can create your books using either WYSIWYG editing or HTML editing of the files, but *Sigil*'s best features are its ePub checking tools. The ePubs standard is stricter than standard HTML, and some viewers have very pedantic rendering engines, so it's important to thoroughly check them before sending them out into the world.

Sigil + Calibre

We use *Sigil* as well as *Calibre* for this when we produce the Linux Voice ePubs, which are available to subscribers. Each has different strengths, so we'd strongly recommend any prospective eBook editors try out both to see which works best for them. Since the ePub format is standard, it's trivial to work on a single book using both tools, and we haven't had a problem doing so.


Linux-Voice-Issue-015-small.epub - Sigil
File Edit Insert Format Search Tools View Window Plugins Help
Book Browser
Text
titlepage.xhtml
new-LV015_003_Welcome.html
new-LV015_006_News.html
news2.html
new-LV015_008_DistroHopper_split_0...
new-LV015_008_DistroHopper_split_0...
new-LV015_010_Gaming.html
new-LV015_012_Mail.html
new-LV015_016_LUGS_v2.html
new-LV015_018_Feature_Hacking_spli...
Clips
Clips Help
Example Clips
p
h1
h2
h3
h4
h1 class
B span
titlepage.xhtml
LINUXVOICE
HACK THE WEB
100%

We use *Sigil* in the process of creating the Linux Voice ePub.

> "Sigil has everything you need to produce your own ePub files."

PROJECT WEBSITE
http://sigil-ebook.com/

Wikipedia reader

Xowa

Wikipedia is probably the greatest single information repository mankind has ever created. It holds a frankly mind boggling amount of information, and despite the potential for vandalism, maintains a high level of accuracy. There's only one problem with it: you have to be online to access it.

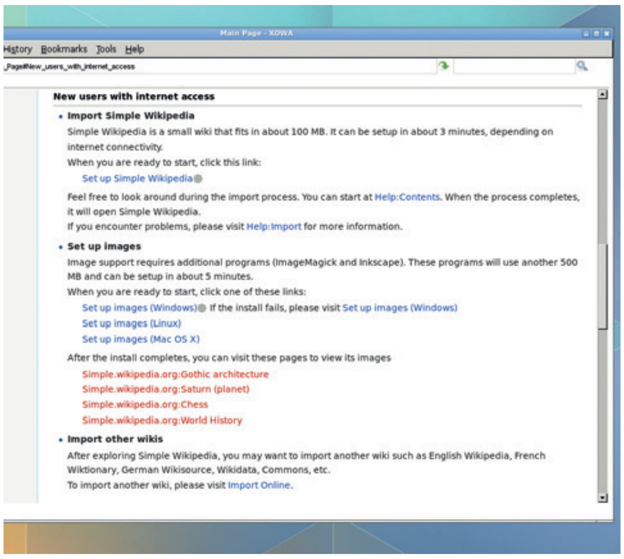
For most situations, this isn't a problem. In todays always-online, smartphone-connected world, many people are connected to the internet every second of every day. However, there's a significant proportion of the world where that simply isn't the case. This is true in rural parts of rich countries, and in huge swathes of the developing world.

Xowa allows you complete access to the whole of Wikipedia when you're offline. This is, obviously, a huge amount of data, so it does require a large download when you install the wiki, but once you've got the data, it's available whether or not you're connected.

Knowledge of the world

If you can get the database, you can store it for use offline. This means that as well as working with Wikipedia in any language, you can use it to grab wiki dictionaries (wiktionaries) or any of the other sources from WikiMedia.

There are links to the appropriate downloads in the Tools > Import Online menu. It lists all the options by size, so you can get anything from the full English Wikipedia (11GB) to the Arpitan (a language from Provence) version of the dictionary that's just 1MB. We would tell you what articles are in the latter one, but we couldn't find an Arpitan reader to translate for us.


Main Page - XOWA
History Bookmarks Tools Help
_Page#New_users_with_internet_access

New users with internet access

- **Import Simple Wikipedia**
  Simple Wikipedia is a small wiki that fits in about 100 MB. It can be setup in about 3 minutes, depending on internet connectivity.
  When you are ready to start, click this link:
  Set up Simple Wikipedia
  Feel free to look around during the import process. You can start at Help:Contents. When the process completes, it will open Simple Wikipedia.
  If you encounter problems, please visit Help:Import for more information.
- **Set up images**
  Image support requires additional programs (ImageMagick and Inkscape). These programs will use another 500 MB and can be setup in about 5 minutes.
  When you are ready to start, click one of these links:
  Set up images (Windows) If the install fails, please visit Set up images (Windows)
  Set up images (Linux)
  Set up images (Mac OS X)
  After the install completes, you can visit these pages to view its images
  Simple.wikipedia.org:Gothic architecture
  Simple.wikipedia.org:Saturn (planet)
  Simple.wikipedia.org:Chess
  Simple.wikipedia.org:World History
- **Import other wikis**
  After exploring Simple Wikipedia, you may want to import another wiki such as English Wikipedia, French Wiktionary, German Wikisource, Wikidata, Commons, etc.
  To import another wiki, please visit Import Online.

Xowa's help files are themselves wiki pages; they're included in the standard install, while you have to download other wikis.

PROJECT WEBSITE
http://xowa.sourceforge.net

Integrated development environment

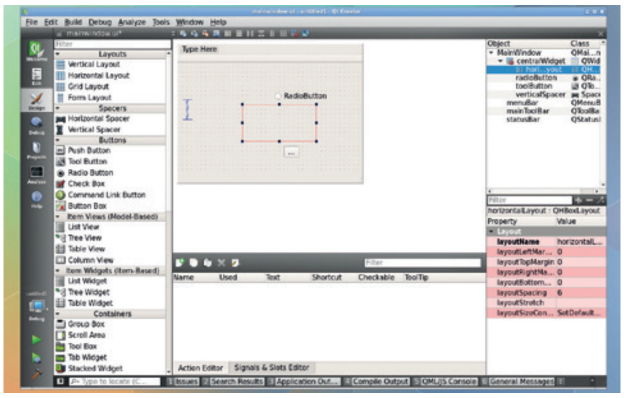
Qt Creator

Writing software requires a few different tools. Depending on your exact requirements, you may need a text editor, build system compiler (or interpreter), interface designer and source code manager. There are two options: you can find separate tools for each task, or you can find one powerful tool that does it all. If you find yourself going down the latter route, you'll need an Integrated Development Environment (IDE).

Qt Creator, as you may have guessed from the name, is an IDE for the *Qt* toolkit. This means it's ideal for building software for KDE and LXQt as well as any of the mobile environments built on *Qt*, such as Sailfish. In fact, *Qt* is an excellent option for cross-platform development, and runs on most modern OSes. It you need even more cross-platform capabilities, *Qt Creator* supports building HTML apps, which provide a great option for cross-platform development.

Although *Qt Creator* can work with other languages and platforms, it's designed to work with C++ and the *Qt* toolkit. Version 3.4 comes with improved support for refactoring, and smarter syntax highlighting. As well as the FOSS version, there are commercial versions of *Qt Creator* with better integration with testing tools, and support from the makers. The community version is perfectly usable for most people, and doesn't feel deliberately crippled to generate sales of the commercial version.

"Qt Creator is an integrated development environment for Qt."


Drag-and-drop your own GUIs, and let *Qt Creator* generate all the code for you.

The biggest advantage of using *Qt Creator* is probably the designer, which lets you drag-and-drop your user interface. This integrates well with the rest of the software, so you can easily build quite complex user interfaces. Fans of the *GTK* toolkit will have to wait for the soon-to-be-released *Gnome Builder* IDE to have the same level of functionality.

PROJECT WEBSITE
https://wiki.qt.io/Category:Tools::QtCreator

Network graphical configurer

NetCTL GUI

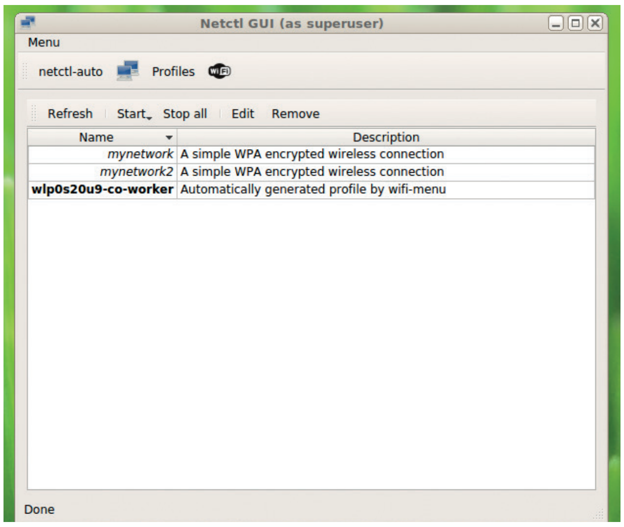
Arch Linux uses *netctl*, a tool to help you configure your network containers using profiles and *Systemd*'s networkd daemon. *NetCTL GUI* is a graphical tool that lets you do all this without having to dig down to the command line.

Of course, some people may argue that if you don't like digging down to the command line, then Arch Linux probably isn't the right distro for you. Maybe they have a point, but there are a few situations when we think *NetCTL GUI* could come in handy. For example, you arrive at a coffee shop and take out your laptop. You go to connect to the wireless network. Can you remember the syntax for the profiles or the arguments for *netctl*? Perhaps you can, but we find that we forget stuff like this all the time.

Usually, the solution is simply to look it up on the Arch Wiki, but you can't until you get your computer connected. In this situation, *NetCTL GUI* would save you fiddling around trying to read the wiki on your phone, or – even worse – turning to a Macbook using friend and uttering the most embarrassing words a Linux user ever has to say to a non-Linux user "Can I borrow your computer for a second? I've forgotten how to configure networking on Arch Linux". *NetCTL GUI* can save you this humiliation – install it now. You may never need it, you may never even open it, but in case you do, you'll thank us.

Safety first!

NetCTL GUI is written using *Qt*, but it's such a simple application that it shouldn't look out of place on a *GTK*-based desktop. We tested it on the Mate desktop, and it worked without problems.



Name	Description
mynetwork	A simple WPA encrypted wireless connection
mynetwork2	A simple WPA encrypted wireless connection
wlp0s20u9-co-worker	Automatically generated profile by wifi-menu

NetCTL GUI is possibly the most important piece of software to stop Arch Linux users losing face.

PROJECT WEBSITE
http://arcanis.name/projects/netctl-gui/

FOSSPICKS Brain Relaxers

Run and gun

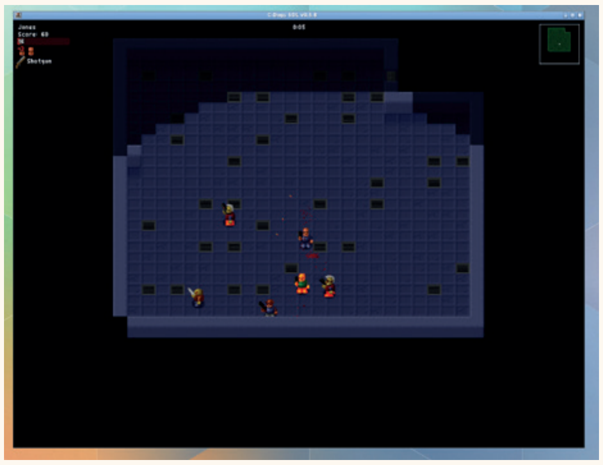
C-Dogs SDL

Originally a shareware game for DOS, *C-Dogs* is a fast-paced running and shooting game for one to four players. It's a straightforward game of keeping on the move and shooting anything else that moves. You can play as several different characters, and the multiplayer options enable you to play either with or against the other players.

The original game was available until 2001, and at the end of its commercial life, Ronny Wester, its programmer, open sourced the code. Naturally, this was picked up by a few developers, and soon the code was ported to the SDL game library. *C-Dogs SDL* was born, and it worked on far more platforms than the original game, including Linux. There's even an Android version for those of you who use Linux on the go.

The graphics are true to the game's 90s heritage, and won't tax your graphics hardware, or require a high-res screen. Simple and pixellated are the two adjectives that best capture the visual feel of the game. Of course, as every retro gamer knows, you can't judge a game by the graphics, and the simple visuals belie great gameplay. There's no real plot to follow, or complex manoeuvres to master: just keep moving, and keep shooting. It's the way games used to be.

C-Dogs SDL is easy to pick up, and quick to play, but also entertaining enough that you won't get bored too soon. It's probably not the game for epic weekend-long sessions, but a quick stress reliever after a long day... just what arcade games were intended for.



The name *C-Dogs* comes from Cyberdogs (the games doesn't feature any pirates).

PROJECT WEBSITE
http://cxong.github.io/cdogs-sdl

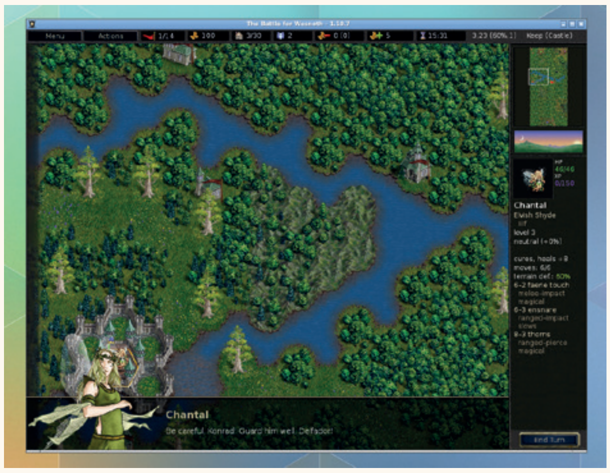
Turn-based strategy

The Battle for Wesnoth

The Battle for Wesnoth is a turn based strategy game set in a Tolkien-esque fantasy realm. You play as one of six factions, which are made up by various combinations of men, elves, goblins, orcs, trolls, dragons, skeletons, mermen and others. There isn't a central theme, but a series of 16 campaigns that you can play, each with its own storyline.

You take it in turns (against the computer) to make your moves and plan your attacks, much like a graphically-appealing game of chess played on a hexagonal grid. Like chess, different players have different abilities, and the key to success is to learn how to use each piece so as to maximise its effectiveness. The similarity ends there though, because in *The Battle for Wesnoth*, you can play with whichever pieces you like, since you amass your army by paying gold to recruit troops. Now I think about it, that could make chess a little better as well. I never could work out a good way of using knights.

Wesnoth walks the line of not being too hard to put off beginners and at the same time not being so easy as to get boring. Even newcomers to the turn-based strategy genre should find it a entertaining game.



Who needs Steam, or friends, or the outside world? Not us. We just need *The Battle For Wesnoth*.

"... a turn-based strategy game set in a Tolkien-esque world."

Once you've completed the 16 official campaigns, you can continue with user-generated ones – there are enough of these that even avid gamers should be playing for a long time.

PROJECT WEBSITE
www.wesnoth.org

SUBSCRIBE

shop.linuxvoice.com

Get your regular dose of Linux Voice, the magazine that:

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

US/Canada subs prices

1-year print & digital: **£95**
12-month digital 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.

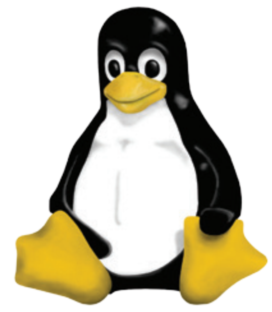
NEXT MONTH IN LINUXVOICE

ON SALE THURSDAY 25 JUNE

BE YOUR OWN SYSADMIN

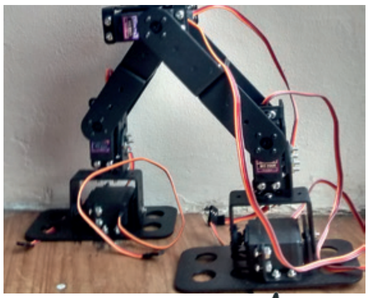
Run your own servers, manage your own data, and take control over all your digital happening – all with Free Software.

EVEN MORE AWESOME!



Inside the kernel

Delve inside the project that everyone relies on – the Linux kernel. Be advised that there may be swearing and intolerance for low quality code…



Robot takeover

Ben is still in his shed, working on an army of Linux-powered robot super soldiers. Who knows what he's planning in the fug of flux and solder?

Profits giveaway

At Linux Voice, we like to put our money where our mouth is, and we're giving half our (modest) first year's profits to help Free Software. Here's where it's going.

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

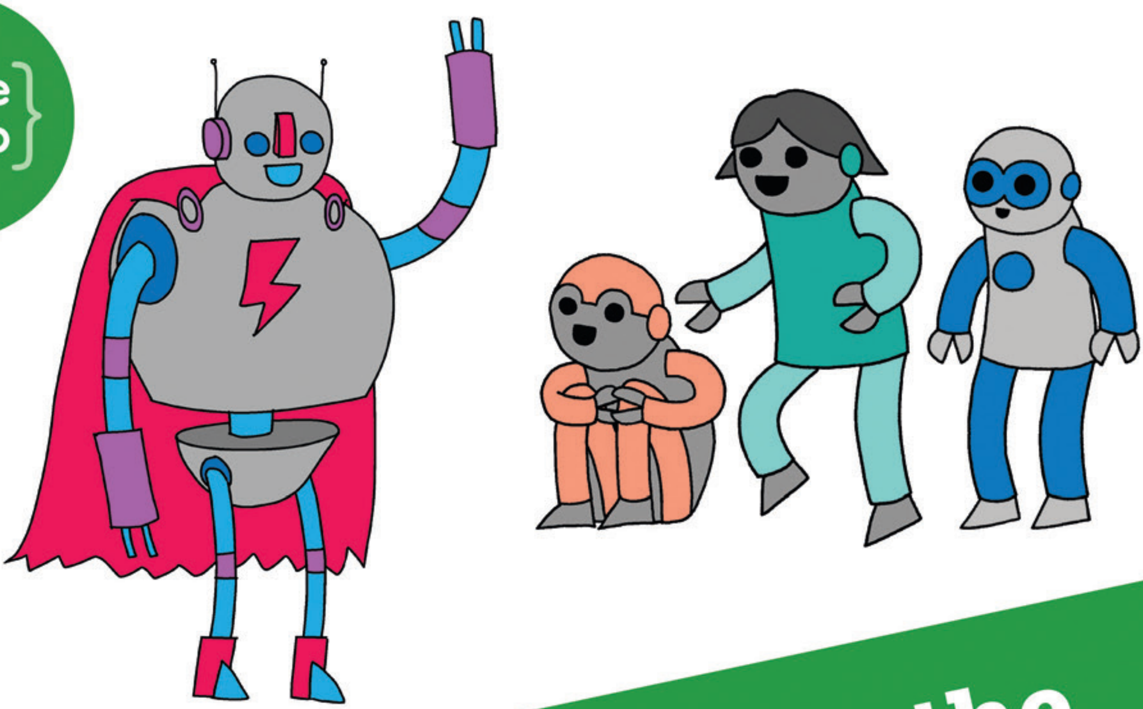
Printed in the UK by
Acorn Web Offset Ltd

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 February 2016 when all content (including our images) is re-licensed CC-BY-SA.
©Linux Voice Ltd 2014
ISSN 2054-3778

Subscribe: **shop.linuxvoice.com**



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
Does his computing on an abacus. It may be basic, but the hardware is open.

I've been writing about Linux for almost half a decade, and I honestly never thought I'd report on Microsoft's *Visual Studio* for Linux. This issue I've done exactly that (albeit in a stripped-down programmer's editor, not the full *Visual Studio* IDE). This raises so many questions. Is this a cynical marketing move or a genuine change of strategy? Will we see more Linux software from Microsoft? What does this mean for Linux as a desktop operating system? For now, we can only speculate wildly, so here are my bold predictions for the future.

Speculation

I think that this is a genuine change from Microsoft, and will likely be accompanied in the future by more software releases for Linux, and continued support. However, it will be several years before we see a major Microsoft product (such as the full *Visual Studio*, or *Office*) released on desktop Linux. We've seen Linux grow slowly but steadily in market share for the last ten years. This release is an acknowledgement of Linux's increasing importance, and I think the trend's likely to continue for some time (at least five years). Will we see a dramatic increase in the uptake of Linux in this time frame? Only time will tell.
ben@linuxvoice.com

LINUX VOICE TUTORIAL

MULTITRACK AUDIO WITH ARDOUR

GRAHAM MORRISON

A great new release gives the perfect excuse to explore some of the new features in this Free Software audio editor.

WHY DO THIS?
- Add effects to recordings
- Edit out dead air and other mistakes

Continuing our loose theme of looking at open source applications capable of competing with the very best commercial alternatives, this month we're going to examine the new version of *Ardour*. *Ardour* is an audio powerhouse, capable of recording, editing and mastering music that could be released professionally (see our review on p50).

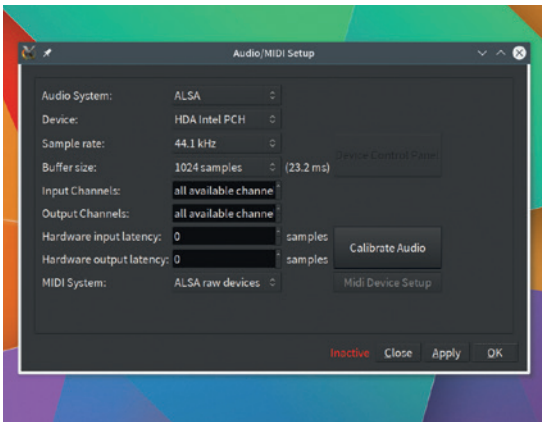
Like *Gimp*, it's difficult to work out how to use it simply by loading the application, but the new version has made things considerably easier by removing Jack as a pre-requisite. Jack is the audio connectivity layer that's powerful but complicated. Its requirement by *Ardour* added a considerable hurdle for users who just wanted to try the application. But you can now simply launch *Ardour* just as you would a similar application such as *Audacity*. We'll use our standard laptop hardware to record a few tracks of audio and use *Ardour* to mix these down to a single audio file. It's the audio equivalent of creating and merging layers in *Gimp*.

Step by step: Record and edit an audio track

1 Connect to your audio

When *Ardour* launches, you'll be asked where you want to create a new session. It's worth doing this in a separate folder, because there are usually lots of files associated with a single project. After giving your new session a name, the 'Audio/MIDI Setup' wizard appears. This is where you tell *Ardour* which audio hardware and driver to use, as well as the sample rate and buffer size. Now that you no longer need to run Jack in the background, you can select ALSA as the audio subsystem and you shouldn't need to make any other system changes. Sample rate is usually best at 44.1kHz (the same as a CD) and the buffer size is fine at 1024 samples.

A larger buffer will help with audio glitches, especially on slower machines, but a larger buffer also increases the time it takes for audio generated in *Ardour* to make a sound – known as latency. The buffer latency is shown to the right of the buffer size field. By default this is 23.3ms (1 second = 1000ms).

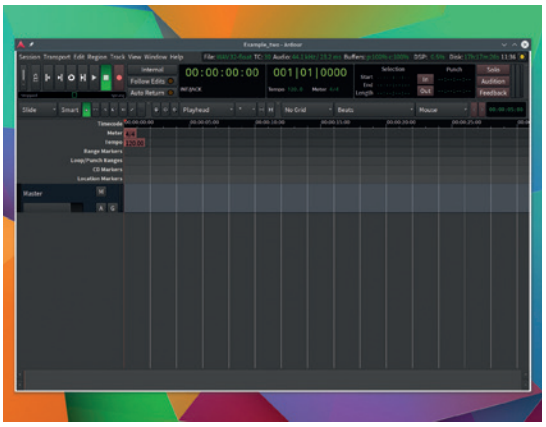

Audio/MIDI Setup

Audio System: ALSA
Device: HDA Intel PCH
Sample rate: 44.1 kHz
Buffer size: 1024 samples (23.2 ms)
Input Channels: all available channe
Output Channels: all available channe
Hardware input latency: 0 samples
Hardware output latency: 0 samples
MIDI System: ALSA raw devices

2 Explore the interface

Click on OK and you'll be presented with a warning about memory limits. We'd recommend disabling this and closing it. The main application window will then appear. First appearances can be intimidating because *Ardour* gives you no clues about what to do next, so we're going to use this as an excuse to provide a brief overview of the interface.

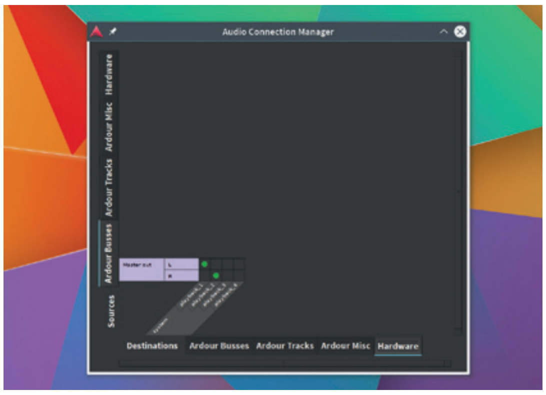
In the very top-right of the window, you'll find various stats about the recording format, CPU state (labelled as DSP) and how much recording time is available in your hard drive, labelled as 'Disk'. The multi-coloured LED is clickable and will be off when there are no messages, yellow when there's an update and red if there are problems. The two large number fields are for the cursor position within the recording, as a time and as bars/beats. Transport controls are to the left and beneath there's cursor, snapping and mouse options. Finally, the entire lower section of this window is for each of your recorded audio tracks, which we'll create now



3 Audio connections

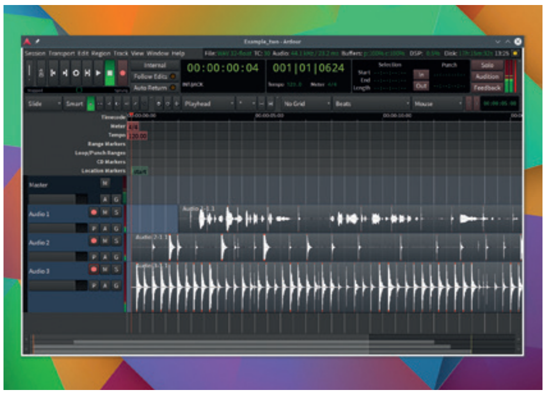
Before you can record anything you have to add a track to record something into. There's already a single entry labelled Master. All tracks and audio will be mixed into this Master bus and sent to your headphones/speakers. To get a clearer idea of this, add a new audio track from the Track menu.

A new red track will be added, and you should notice a volume meter bouncing up and down on both this track and the Master bus. This is the input from your microphone connected by default to the new track. All connections within *Ardour* are configurable from the Window > Audio Connections panel. Currently, the hardware 'Source' is connected to the track 'Destination', the Source > Track is connected to the Destination > Master bus and the Source > Master bus is connected to the Destination > Hardware Output.



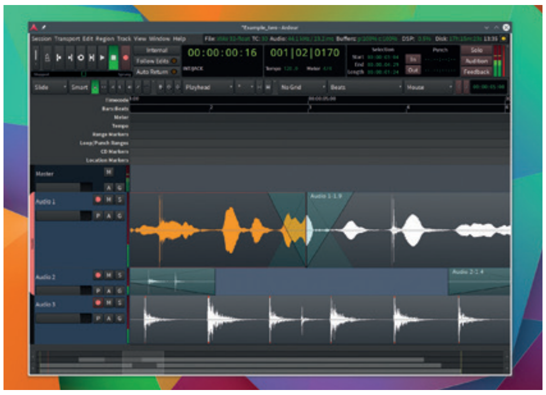
4 Recording audio

Ardour will always attempt to make sensible connections by default, but you can always go into the Connections window and change them, much as you would in a recording studio by unplugging one cable from a mixer channel and connecting it somewhere else. This is how you'd set up multiple inputs to be recorded at the same time on different tracks, for example. Hopefully, your microphone or other input is already connected to the input and the volume meter is bouncing around on the track. This means you're ready to record, and to make a recording you need to 'Record Enable' the new track by pressing the red button and then press the large red recording button in the transport section followed by 'Play'. As the recording progresses you should see the amplitude waveform drawn into your record enabled tracks.



5 Editing your masterpiece

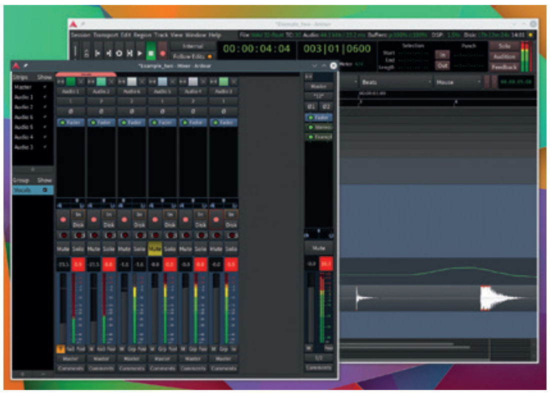
The next step is editing, and just like you'd edit out errors with *Gimp*, it's now time to edit your recordings. Use the cursor to trim the beginning and end of the audio, or move blocks against one another in time to improve timing. Drag corners to add fade-in and fade-out effects. Press C for cut mode, which will let you split blocks so you can remove or edit only one part. Press T to enter stretch mode. This will let you drag blocks to fit the time/space, but doesn't sound so great with extreme stretches. You can get a better view of your editing by changing the zoom level with your mouse wheel, or adjust the vertical height of each track using the right-click menu, and can group tracks together and edit across the multiple tracks when more than one is selected.



6 Mastering and output

The final step is mastering your edit into a single file. To start, open the mixer view from the Window menu. You can adjust the relative volume and left/right balance of your tracks here. If you've got any LADSPA effects installed, you can use them too. If you want to make changes happen over time, you'll need to use automation, enabled back in the track view by pressing the A button. Select Fader to adjust the volume and use the Write mode, which will record changes in the fader level as you adjust them during playback.

To export your creation, select Session > Export > Export to Audio. Change the format to WAV or Ogg Vorbis and make sure the time span covers your whole project. Now click Export. *Ardour* will play through your recording and write the output to a single audio file. LV



LINUX VOICE TUTORIAL

CODE YOUR MINECRAFT WORLD USING SONIC PI

LES POUNDER

Learn a new way to hack Minecraft, using Sonic Pi. Shape your world and make a lot of noise doing it!

WHY DO THIS?

- Merging *Minecraft* and *Sonic Pi* into one project is a genius way of introducing coding, music and game design.

TOOLS REQUIRED

- A Raspberry Pi 2.
- Keyboard,mouse and monitor with sound capabilities.
- If your monitor does not have sound, plug in headphones to your Raspberry Pi 2 and use Sonic Pi preferences to change output.
- You can download the completed code from **https://github.com/lesp/LinuxVoice_Issue16_Education/archive/master.zip**

Two applications on the Raspberry Pi have ignited imaginations more than any others: *Minecraft* and *Sonic Pi*. *Minecraft* is the open world sandbox game in which you can build anything; it even has its own Python API that enables anyone to shape their world using programming. *Sonic Pi* is a popular music creation tool that uses Ruby as its programming language. But what if we could merge them together and control *Minecraft* using *Sonic Pi*, while programming music? Well now we can!

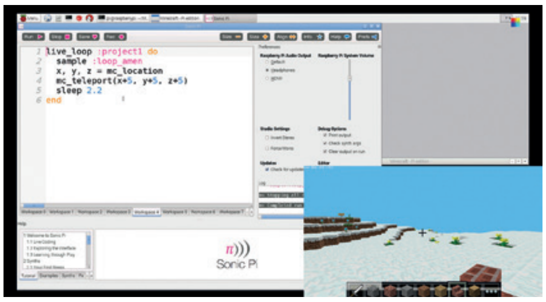
In this tutorial we will be using the development version of *Sonic Pi*, as at the time of writing it is the only version capable of talking to *Minecraft*.

Setting up *Sonic Pi Dev* requires a little command line magic. So on your Raspberry Pi 2, connected to the internet, open a terminal and issue the following command.

```
git clone git://github.com/samaaron/sonic-pi.git
cd sonic-pi/app/gui/qt
sudo ./rp-fetch-deps
./rp-build-app
cd ../../server/bin/
./compile-extensions.rb
cd ../../../
./bin/sonic-pi
```

So what have we just done? Well, we've used **git clone** to download the code from GitHub, then changed directory and run a script to download the dependencies for *Sonic Pi Dev* to work. We have then built the application from source, and finally run *Sonic Pi Dev* from its directory rather than call the version of *Sonic Pi* that's installed by default on Raspbian.

Sonic Pi



```
live_loop :project1 do
  sample :loop_amen
  x, y, z = mc_location
  mc_teleport(x+5, y+5, z+5)
  sleep 2.2
end
```

In our first project we create a teleporter that plays music upon a successful teleport. Beam us up, Mr Scott!

Sonic Pi contains eight workspaces, and each workspace has its own programming panel. Workspaces can be played simultaneously and will automatically save when *Sonic Pi* is closed.

Just above the programming panel are four buttons used to play and stop your composition, save your work to a file, and lastly to record your composition to a WAV file for use in another application.

With *Sonic Pi Dev* ready to go, let's fire up *Minecraft Pi*; you can find it in the menu under Games. Create a new world in *Minecraft Pi* and wait for it to load. When it's ready we need to switch our focus back to *Sonic Pi Dev*, and the easiest way to do this is by pressing the Tab key on your keyboard. This will release the mouse and enable us to select the *Sonic Pi Dev* window.

In *Sonic Pi Dev* we'll write a line of code in Workspace 0 to test that the connection between *Sonic Pi Dev* and *Minecraft Pi* is complete. The most basic test is to push a message to the *Minecraft* chat window and we do this as follows.

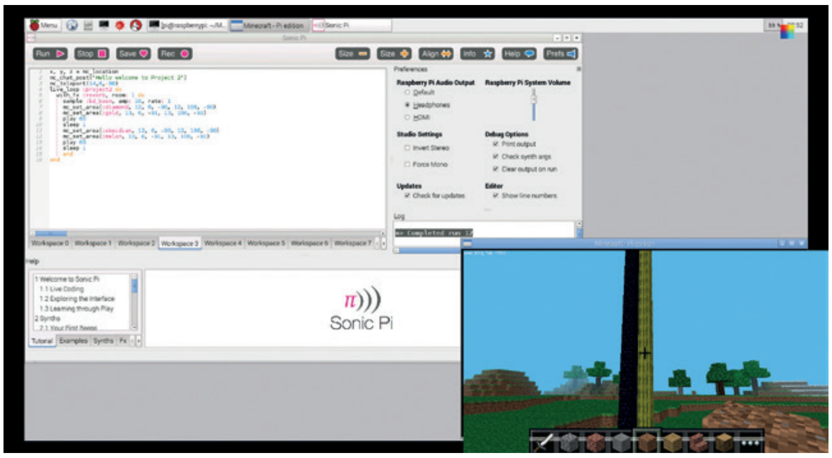
```
mc_message("Hello World")
```

Press Run and change your focus back to the *Minecraft Pi* window. You will see "Hello World" on the screen, proving that the connection has been made. Unlike the Python API, which requires us to import the *Minecraft* library, *Sonic Pi 2.5.0 Dev* is already configured to connect to *Minecraft* if we give it a command that will output in the *Minecraft* world.

Project 1 – Mixing music with worlds

Our goal for this project is to teleport the player from one place to another and to play a jingle at the same time. Let's begin with our first line of code, which is the start of an infinite loop – but not your typical "while True" loop. Rather, we use.

```
live_loop :NAME do
```



```
x, y, z = mc_location
mc_chat_post("Hello welcome to Project 2")
mc_teleport(14,6,-90)
live_loop :project2 do
  with_fx :reverb, room: 1 do
    sample :bd_boom, amp: 10, rate: 1
    mc_set_area(:diamond, 12, 6, -90, 12, 106, -90)
    mc_set_area(:gold, 13, 6, -91, 13, 106, -91)
    play 65
    sleep 1
    mc_set_area(:obsidian, 12, 6, -90, 12, 106, -90)
    mc_set_area(:melon, 13, 6, -91, 13, 106, -91)
    play 63
    sleep 1
  end
end
```

The two towers use different block types to give them a different colour. There are lots of block types for you to use, just be careful with the flowing lava and water.

```
#Code to loop indefinitely
end
```

This loop enables us to update its contents on the fly, an increasingly popular and challenging practice known as live coding, where changes to code are instantly reflected and shown to a live audience. A live loop needs a name, so we'll call this one "project1".

Once you make changes to the code you'll need to reload the code using the Play button; it will take effect the next time the code is loaded, with no break in the music. At the end of a loop we must make sure that we close the loop using **end**, otherwise we'll generate an error in *Sonic Pi*.

Inside the loops we'll add the code that we wish to repeat, which will be as follows.

```
Find the position of the player
Teleport them 5 blocks in every direction
Play a sample to indicate success.
```

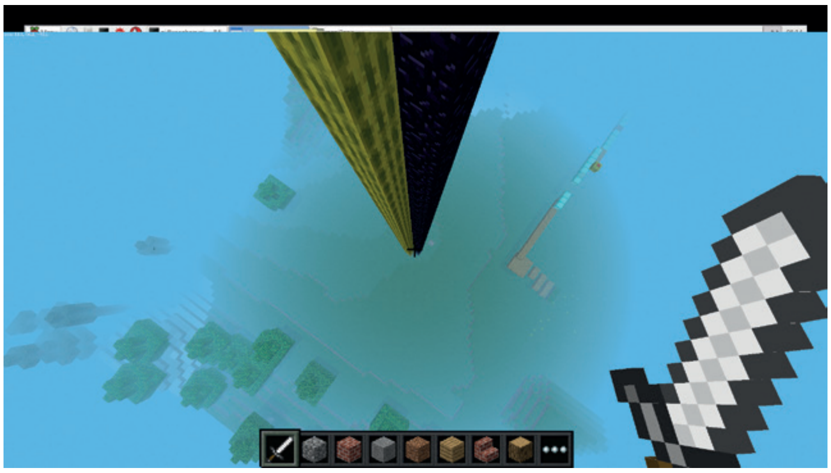
Teleportation is achieved by changing the player's x,y and z positions in the world, and we can do this in two ways: using a relative position to our current position, or with an absolute position. The benefit of using a relative position is that we can transport the player around the map no matter where they are, whereas an absolute position will return our player to the same place in the map.

In order to teleport a player to a relative position we'll need to know where they are in the *Minecraft* world. We can find this out using **mc_location**, which will return the player's coordinates in the world. In order to breakout the coordinates into each axis we will create three variables called **x**, **y** and **z**. Now our code will look like this.

```
live_loop :project1 do
  x, y, z = mc_location
end
```

Now each time the loop iterates we will update the variables with the player's current location. Let's use these positions to teleport our player using **mc_teleport** from the current position to 5 blocks in every direction, effectively moving us in a 3D movement to the top-right of a 5x5x5 cube – that is, until gravity takes hold and drops you to the ground. We'll include a two-second delay to enable the teleport to be successfully displayed on the screen.

```
live_loop :project1 do
  x, y, z = mc_location
  mc_teleport(x + 5, y + 5, z + 5)
  sleep 2.2
end
```



In Project 2 the towers are 100 blocks high, with each block being 1m high. This means that we have created 200m of blocks in less than a second!

With the code in place, let's test it out. Click on the Play button and switch your focus to *Minecraft*. Every two seconds your player will be teleported to a new location. With that test successfully under our belt press the Stop button to cease teleporting the player. Now we need to add the sound effect to accompany the teleport. There are many different samples that come with *Sonic Pi*, but one of the most recognisable is called Amen, used since 1969 as a drum solo and a sample added to hip hop music. In our code it looks like this.

```
live_loop :project1 do
  sample :loop_amen
  x, y, z = mc_location
  mc_teleport(x + 5, y + 5, z + 5)
  sleep 2.2
end
```

Now click on the Play button to restart the code, and we should see our player teleport along with a brief sample of the Amen loop.

Project 2 – Create to the beat

One of the most exciting areas for merging *Sonic Pi* and *Minecraft* is creating interactivity between the two. In this project we'll show how we can synchronise the beat of the music to activity in the world. Let's start with a blank workspace and a *Minecraft* world open and ready. The first line of code is a simple post to chat message welcoming the player to the project.

```
mc_chat_post("Hello welcome to Project 2")
```

Next we will teleport the player to a location:

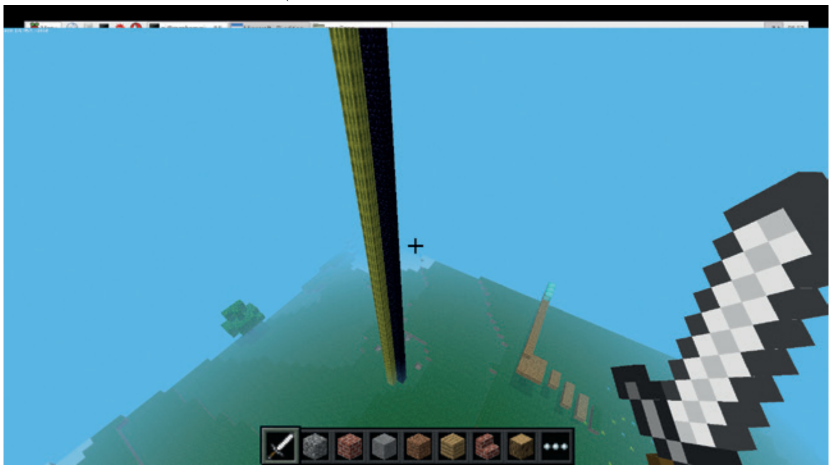
```
mc_teleport(14,6,-88)
```

With these two lines completed we now turn our attention to the loop. Again we will use the same live

Sonic Pi and Ruby

Developed by Dr Sam Aaron as a personal project to create music using programming logic, Sonic Pi's is now part of the default Raspbian operating system, bringing it into the homes of many thousands of eager Raspberry Pi users.

It's powered by Ruby, and its ease of use and clear syntax enables anyone to compose their own music. The Ruby language was created in the 1990s by Yukihiro "Matz" Matsumoto as a personal project to help create an object-oriented scripting language. Sonic Pi is a great gateway to learning more about Ruby, and there are two great resources we'd recommend if you're just starting out: an application called *Kids Ruby*, available from **http://kidsruby.com**, and the book *Ruby Wizardry*, written by Eric Weinstein and published by No Starch Press.



Our towers rise menacingly from the ground and dwarf the course for Project 3!

loop as per project ,1 but we'll call the loop **project2**.

```
live_loop :project2 do
```

In the loop we will create another loop to handle playing any samples of audio with effects, in this case reverb to give the audio a presence and gravity. The sample that we will play is a boom, and we will play it once per second.

```
 with_fx :reverb, room: 1 do
  sample :bd_boom, amp: 10, rate: 1
```

Now we change our focus to building the towers of blocks. The towers are built at absolute coordinates, so they will appear at the same place every time. We first create the two towers using the command **mc_set_area** and then specify the type of block to use. We then provide the starting x,y and z coordinates and then the finishing coordinates. So our first tower is made of diamond and starts at x12, y6 and z-90 on the map. We then specify that the x and z coordinates remain the same, but the tower will be 100 blocks tall, this is done by setting the final y value to 106.

```
  mc_set_area(:diamond, 12, 6, -90, 12, 106, -90)
  mc_set_area(:gold, 13, 6, -91, 13, 106, -91)
```

With the first tower complete, you can see that our second tower is made of gold and has the same height, but the start and finishing x and z coordinates are next to the main tower. Now that we have created the towers we'll play a note to create the beat of our project.

```
  play 60
  sleep 1
```

Sonic Pi uses MIDI numbers to identify notes that you can play in your compositions, and the note number 60 is equivalent to a middle C. After the note has been played we instruct the project to pause for a second. This helps us to control the beat.

We will now alter the blocks that make up our towers, by changing the block type, but keeping the x, y and z coordinates the same. In this case we swapped the diamond for obsidian and the gold for melon.

```
  mc_set_area(:obsidian, 12, 6, -90, 12, 106, -90)
  mc_set_area(:melon, 13, 6, -91, 13, 106, -91)
```

We now play a D# note and again instruct *Sonic Pi* to wait for 1 second.

```
  play 63
  sleep 1
```

Finally we close the two loops that make up the project; these are **live_loop do** and **with_fx**.

```
 end
end
```

With the code complete, click on the Play button and then change your focus to the *Minecraft* window. You'll see your player teleported to the location of the towers and a *24*-esque booming noise will sound to the beat.

Project 3 – Crazy platforms

For the last project we will create our own platform game along with a soundtrack. We start with a new workspace and *Minecraft* open with a new world. Our first few lines of code again use the chatbox to introduce the project.

```
mc_chat_post("Welcome to project 3 - can you make it to the end")
mc_chat_post("without falling off?")
```

Next up we level out the playing area by turning a big cube of the world into air.

```
mc_set_area(:air, -12, 8, -81, 3,20,-56)
```

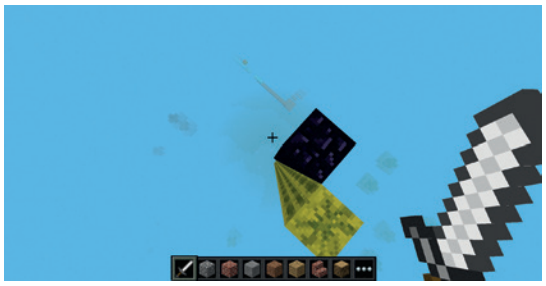
Now we teleport the player to the start of our platform game.

```
mc_teleport(-12,8,-81)
```

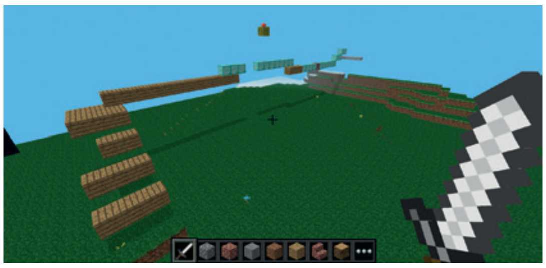
This time we start the live loop process again, renaming the loop to project3. We also create another loop to handle the reverb audio effect and reuse the boom sample from project 2.

```
live_loop :project3 do
 with_fx :reverb, room: 1 do
  sample :bd_boom, amp: 10, rate: 1
```

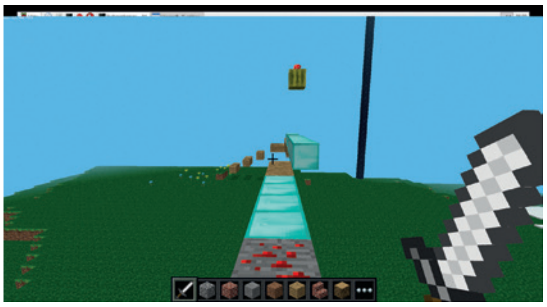
Now we create the first few platforms, these are how our player can jump from place to place. We create them using the **mc_set_area** function that we

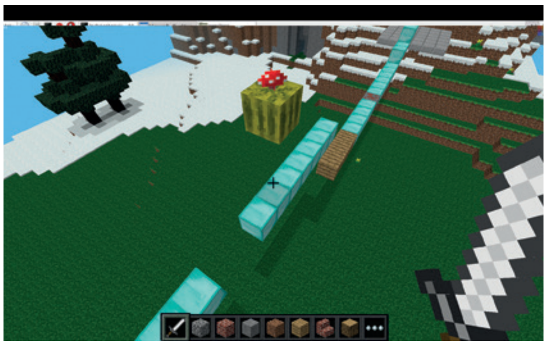


Our towers are so high that they exceed *Minecraft*'s draw distance, creating a fog effect for objects too far away.



This course resembles a typical platform game, with hazards around every corner! It would look rather good with lava underneath it.


A player's-eye view of the course – this is similar to what Mario can see. That platform looks rather narrow!


In classic homage we've included a yellow power up box with a mushroom inside it. Remember kids, don't eat any mushrooms that you cannot identify.

learnt in Project 2. The first platform will be made of grass and be three blocks long; the next two will be made of wood and made to hover in the air by changing their y coordinates.

```
mc_set_area(:grass, -12, 8, -81, -12, 8, -83, )
mc_set_area(:wood_plank, -10, 9, -81, -10, 9, -83, )
mc_set_area(:wood_plank, -8, 10, -81, -8, 10, -83, )
```

To make it a little tricky our next block is only two units long, controlled by the z axis.

```
mc_set_area(:wood_plank, -6, 11, -81, -6, 11, -82, )
```

Our next portion of code creates a larger platform by expanding the x and z coordinates but keeping the y coordinate the same.

```
mc_set_area(:wood_plank, -4, 12, -81, -2, 12, -83, )
```

Now we create a long thin row of wooden blocks.

```
mc_set_area(:wood_plank, -2, 13, -81, -2, 13, -70, )
```

On top of some of the wooden blocks we add some diamond blocks to create a raised platform. Then we create a gap between them by ending the first row at z -69 and starting the next at z -66. We also create some air blocks in between them.

```
mc_set_area(:diamond_block, -2, 14, -71, -2, 14, -69 )
mc_set_area(:diamond_block, -2, 14, -66, -2, 14, -60 )
mc_set_area(:air, -2, 14, -68, -2, 14, -67 )
```

Now we create a yellow block and mushroom as an homage to the Mario games. This will appear above our diamond path.

```
mc_set_area(:melon, -2, 18, -65, -2, 18, -65 )
mc_set_area(:mushroom_red, -2, 19, -65, -2, 19, -65 )
```

We now create a platform to the right of the original path, requiring the player to side step off at their peril.

```
mc_set_area(:wood_plank, -3, 13, -59, -3, 13, -61, )
mc_set_area(:diamond_block, -3, 13, -58, -3, 13, -55 )
```

We change the block type to redstone and drop a block in the line, we'll use it later on.

```
mc_set_area(:redstone_ore , -3, 13, -54, -3, 13, -54 )
```

A few more diamond blocks to make up a path to the end of the course.

```
mc_set_area(:diamond_block , -3, 13, -53, -3, 13, -48 )
mc_set_area(:diamond_block , -3, 14, -47, -3, 14, -46 )
mc_set_area(:diamond_block , -3, 15, -45, -3, 15, -44 )
```

The goal for our course is the large stone dance floor; we'll reuse the same technique that made the large wooden platform earlier.

```
mc_set_area(:stone_slab , -1, 14, -43, -6, 14, -38 )
```

Remember that block of redstone we used earlier? The next portion of code will change it to a block of air every second, effectively creating a trapdoor – sneaky!

```
sleep 1
mc_set_area(:air , -3, 13, -54, -3, 13, -54 )
sleep 1
```

For our last portion of code we create another **live_loop** to handle the music, which is a mix of an industrial-sounding sample and an electric guitar playing chords. We use the reverb effects to give it an ominous twist.

```
live_loop :industrial do
 with_fx :echo, mix: 0.3, phase: 0.25 do
  #sleep 2
  sample :loop_industrial, rate: 0.5
  sample :guit_e_fifths, rate: 0.5
 end
 sleep 4
end
```

With this code complete, press the Play button in *Sonic Pi* and you will see that the *Minecraft* window will update to show the game. Can you make it to the end without falling off?

And that's it! We have taken our first steps with *Sonic Pi* and *Minecraft*. If you'd like to know more, there's a handy resource provided by the Raspberry Pi Foundation at **http://bit.ly/LV_SonicPi**.

Les Pounder divides his time between tinkering with hardware and travelling the United Kingdom training teachers in the new IT curriculum.

Sonic Pi resources

The Raspberry Pi Foundation's education team have created a series of worksheets that are free to use and download from their website: **www.raspberrypi.org/resources/learn**. Key computing concepts such as loops, variables etc are explained using musical composition, helping children to understand both music and coding, but with a heavy slant upon the musical elements.

The *Sonic Pi* website at **http://sonic-pi.net** is also a great resource, with code snippets and audio samples on the site for you to insert into your own compositions. There are also sections covering the use of *Sonic Pi* in the classroom along with guides for teachers to use when planning lessons.

Sonic Pi can be downloaded for Windows, Mac and Raspberry Pi from the *Sonic Pi* website. At the time of writing there are no prepackaged versions for Linux distributions other than Raspbian, but you can download the source code from **https://github.com/samaaron/sonic-pi** and build it from source.

LINUX VOICE TUTORIAL

GPX PROCESSING: CREATE, EDIT AND SHARE GPS DATA

MARCO FIORETTI

Don't get lost on the way to the shops – keep an (open, editable) record of your movements with GPS.

WHY DO THIS?

- Make sure that all your GPS data will always remain available
- Prepare high-precision digital routes for your future vacations
- Share GPS data with others, or create it to geotag your own photographs

As anyone who's studied the golden age of Elizabeth the first knows, maps are important. The Spanish maps of the time were more or less green blobs, while the English and Dutch maps were superbly accurate for the time, enabling an era of maritime dominance. Those days are gone now, and in the past they must remain, but maps are still important. Remembering the exact path followed during a trekking, for example, lets you give safer advice to friends interested in the same route. Detailed analyses of walks or biking trips may provide useful inputs for staying in good physical shape. Automatic comparison and processing of travel logs from many people are essential in many important activities, from mapping to traffic predictions and planning of public transport.

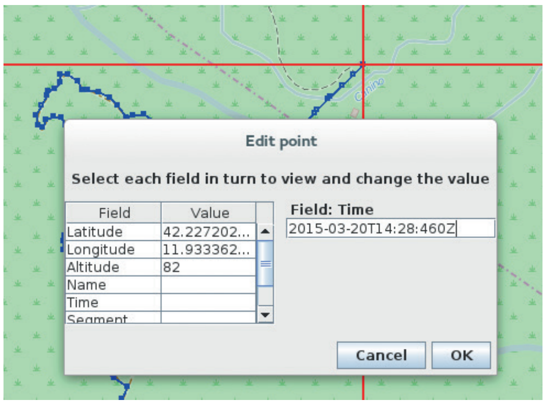
Any scenario like these relies on the same data: sequences of location/time pairs, which can be recorded in real time by Global Positioning System (GPS) devices or generated by software.

This tutorial describes GPX – the GPS eXChange Format. Many products with GPS capabilities store data in their own format. Therefore, an open GPS interchange standard like GPX is the best way to make sure that your GPS data will remain fully reusable with any other GPS software or navigator.

GPX is also very easy to hack, because it's just another dialect of XML, the eXtensible Markup Language: plain text, with each type of data enclosed by appropriate markers. Of course, this doesn't explain why anyone but GPS maniacs would want to mess with GPX files in the first place. The point is that knowing GPX is essential to obtain GPS data good enough to be useful for you, in ways that you may never have imagined. Even where there was no data at all to begin with!

Good GPX tracks are usable for creating custom maps of your trips, or routes for your GPS navigator. They are also essential for geotagging; that, is giving geographical coordinates to photographs and other documents, so they can show up on any digital, interactive map.

Unfortunately, most of the GPS tracks recorded in real time by ordinary navigators and smartphones are not reusable in any meaningful way. Real GPX files often contain redundant data (such as duplicate points where you stopped to enjoy the scenery, or roads already saved), errors (points with bogus coordinates, caused by poor signal reception) and actual holes, where there was no satellite signal at all. This unfortunate fact of life, which has nothing to do with file formats, is why you want to be able to create and edit, not just use, GPX files.



Edit point

Select each field in turn to view and change the value

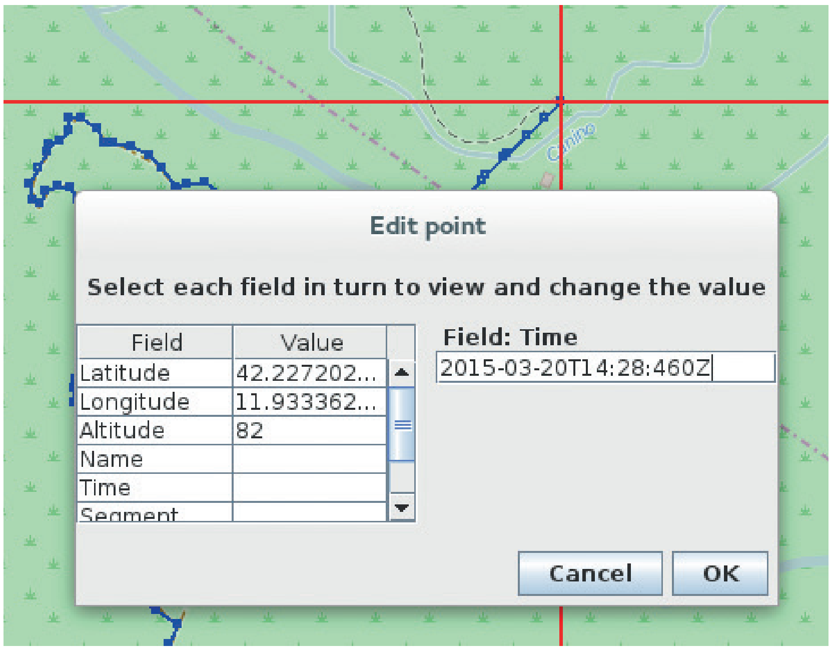
Field	Value
Latitude	42.227202...
Longitude	11.933362...
Altitude	82
Name	
Time	
Segment	

Field: Time

2015-03-20T14:28:460Z

The *GPSPrune* main window, showing how it renders a GPS track downloaded from **gpsies.com**, and how much detail can be hidden in each trackpoint of a GPX file.

If you really want or need to do it, both *GPSPrune* (in the screenshot) and *Viking* will let you edit waypoints directly. These operations, however, can be often done much more efficiently at the command line, with *GPSBabel*.



Edit point

Select each field in turn to view and change the value

Field	Value
Latitude	42.227202...
Longitude	11.933362...
Altitude	82
Name	
Time	
Segment	

Field: Time

2015-03-20T14:28:460Z

The GPX format

Let's start by looking at a very short snippet of a basic GPX file (edited for clarity!):

```
<?xml version="1.0"?>
<gpx version="1.0" creator="Viking"
```

```
xmlns:xsi="http://www..."
xmlns="http://www.topografix.com/GPX/1/0"
xsi:schemaLocation="http://www... ">
<trk>
  <name>mignone-track-1</name>
  <trkseg>
  <trkpt lat="42.202982878" lon="11.882421524">
    <name>Beginning ofthe hike</name>
    <time>2015-03-20T09:09:18Z</time>
    <ele>115</ele>
  </trkpt>
ETCETERA...
```

Not really hard to understand, is it? The initial header may also contain, among other things, a detailed description of the file, author name, or keywords.

After the header, a GPX file may contain any combination of waypoints, tracks and routes. The example shows the beginning of a track called **mignone-track-1**, with name, coordinates, elevation and creation time of the first point (**trkpt**) of its first segment (**trkseg**).

GPX files use the WGS 84 datum (see the box, right, for what this means), storing latitude and longitude in decimal degrees and elevation in metres. Times, in the format YYYY-MM-DDTHH:MM:SSZ, are always in the Coordinated Universal Time (UTC) time zone. A point may also contain data including GPS signal strength and a specific place (the **<extension></extension>** tag) for, you guessed it, data from third-party extensions.

Three great GPX free software tools

There are several programs for GPX editing on Linux. Three of the best are: *GPSBabel* (**www.gpsbabel.org**), *GPSPrune* (**http://activityworkshop.net/software/gpsprune**), and our personal favourite, *Viking* (**http://sourceforge.net/projects/viking**).

GPSBabel is a universal GPS converter and processor that can work with literally hundreds of GPS formats beside GPX. The only way to convert many GPS tracks at once is to use scripts, and the only tool usable with these scripts is *GPSBabel*, so you'd better focus on how it works at the prompt.

The simplest *GPSBabel* commands just convert formats, and all take this form:

```
-gpsbabel -i INPUTFORMAT -f INPUTFILE -o OUTPUTFORMAT -F OUTPUTFILE
```

where the **-f** and **-F** switches define the names of the input and output files, and **-i** and **-o** their respective formats. The real power of *GPSBabel*, however, is its collection of filters, which we will show in a moment.

GPSPrune, which runs anywhere there is Java 1.5 or later, is great for editing already-existing GPX files. *Viking*, on the other hand, lacks some of the niftiest functions of *GPSPrune* (including a real undo feature), but being a native Linux application it fits better with most Linux desktops. Besides, it's great for drawing GPX tracks and routes from scratch, and can create maps with the *Mapnik* toolkit, search tracks by date and plenty of other things.

Micro GPX/GPS glossary

GPS is a complex technology, heavily based on very complicated calculations and mathematical models. In order to use it as we propose here, however, you only need to know the meaning of a few key terms explained here. However, in other GPS-related contexts, several of the terms below can, and do, have additional meanings!

- **Geodetic Datum** A mathematical model of the real surface of Earth, which is not a perfect sphere or ellipsoid. Several datums exist, partly for historical reasons and parthly because each of them describes some part of the planet better than the others. All GPS coordinates are based on the WGS-84 datum surface.
- **Waypoint** A single point, defined by its latitude, longitude and elevation, of the Earth's surface. GPS files and devices store generic landmarks as waypoints, from monuments to restaurants, as well as user-specific locations, like your home or place of work.
- **Route** A series of waypoints that correspond to the exact path one intends, or should follow to go from one point to another, eg from home to office. Internally, a route can be divided in several legs.
- **Track** A series of trackpoints that describe exactly the path that somebody has already followed to go from one place to another, including when he or she moved through each point. In other words, while routes are suggestions of directions for the future, tracks are logs of actual travels, already happened. Tracks can be divided into segments.
- **Trackpoint** Just like a waypoint, but with more data, because it must describe when someone passed there, at which speed, how the GPS signal was received, and other stuff.

Let's now take a quick look at how to increase the quality and usability of the GPX files you already have. Some operations are much faster at the command line with *GPSBabel*, and some with the other tools. Once you start, you'll quickly find out which tool is better at each task, for your actual needs and taste. Here, we give more examples with *GPSBabel* simply because it is the least intuitive program of the bunch. All the examples focus on tracks, because they usually need much more editing, much more often, than waypoints and routes. Most of what follows, however, also applies to those other kinds of data.

Metadata

The best way to make GPX files useful and reusable is to add rich and consistent tags and other metadata. Unfortunately, this is also a task that's better done the "dirty" way – adding them manually with a text editor, or writing shell scripts that perform the same tasks. This is much easier than you may think, since GPX is plain text. And even if there were full-featured GUIs for GPX metadata, you would have to do a lot of manual work inside them anyway. No software may decide autonomously, for example, what it should write as "author" or "licence" if you combine many tracks from different sources. Scripts or manual editing may also be mandatory to remove sensitive data before sharing the files.

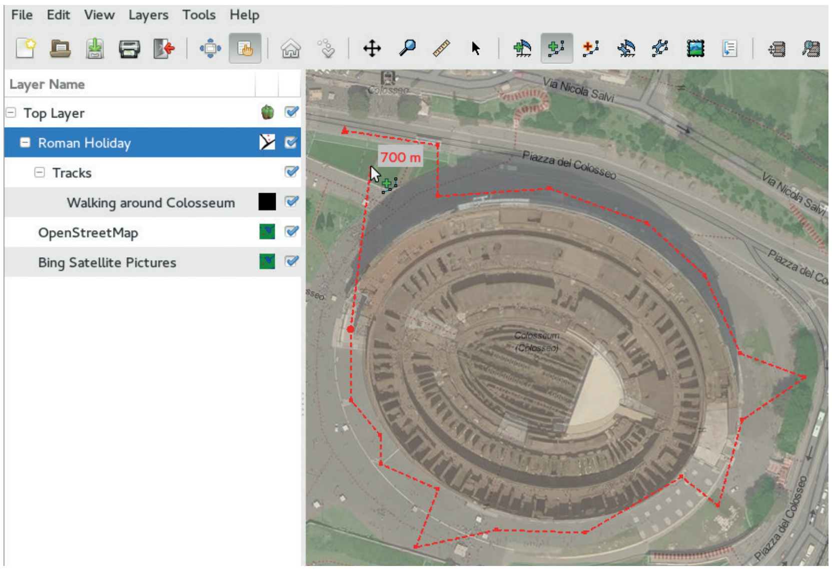
Track transformations

A common need with GPX files is to combine multiple tracks (maybe recorded simultaneously by different GPS navigators) into a single one that fully describes the whole trip. With *GPSBabel*, you can do this with the **merge** or **pack** options, as in these examples that introduce the general syntax of *GPSBabel* filters:

```
gpsbabel -t -i gpx -f track-1.gpx -i gpx -f track-2.gpx -x
track,merge,title="FULL TRIP" -o gpx -F full-trip.gpx
```

LV PRO TIP

It's an absolute must to give each item a coherent name and set of labels, if you want to keep your collection under any semblance of control. This is also true with GPX files, so plan carefully how to organise yours before you start.



Drawing tracks and routes in *Viking* is a pleasure. Do you see the street names? That's OpenStreetMap layered over right over satellite pictures, to let you see as exactly as possible where each trackpoint should go.

LV PRO TIP

Many GPX operations can only be done in a graphical interface, with a good map as background. For many others, though, shell scripts remain a much more efficient. If you want to become a serious GPX hacker, refresh your scripting skills first!

```
gpsbabel -t -i gpx -f track-1.gpx -i gpx -f track-2.gpx -x track,pack,title="FULL TRIP" -o gpx -F full-trip.gpx
```

In the first case, the **track, merge, title** part tells *GPSBabel* to take all the input tracks, filter them by merging all their points into one sequence sorted by timestamps, and put that sequence into one new track called **FULL TRIP**. The **pack** filter of the second command is more efficient, as it simply appends tracks to one another as they come. For the very same reason, however, it will produce correct output only if applied to tracks that never overlap in time!

Combining multiple tracks is as common a need as its opposite – splitting single GPS tracks. Many such tracks, in fact, contain different independent paths that have no real reason to stay together, like trips to unrelated places. Splitting those tracks, in the same or different files, makes them more searchable and can be done in several ways. *GPSPrune* and *Viking* both have relatively intuitive commands for this, but may be much slower to use with the required accuracy. *GPSBabel* filters, instead, can automatically split tracks according to time and distance, exactly as you need:

```
gpsbabel -t -i gpx -f in.gpx -x track,split,title="LOG # %Y%m%d" -o gpx -F out.gpx
gpsbabel -t -i gpx -f in.gpx -x track,split=4h,title="LOG # %c" -o gpx -F out.gpx
gpsbabel -t -i gpx -f in.gpx -x track,sdistance=0.5k" -o gpx -F out.gpx
```

The first command creates one separate track per day, titled **LOG of YYYYMMDD**. The second starts a new track every time two consecutive points have timestamps at least four hours apart. The **sdistance** filter does the same thing every time the distance between consecutive points is above the given threshold (500 meters in the example). The **split** and **sdistance** filters can be combined (written one after the other) separated by a comma: this makes *GPSBabel* create new tracks only if both time and distance intervals exceed their thresholds.

Two other important transformations are data structure conversion and reversion. Conversion transforms a track or set of waypoints into a route, or *vice versa*. Reversion produces instructions to, so to speak, "get back home", by reverting the order of all the points into a track or route. To transform, for example, a set of waypoints into one route, use:

```
gpsbabel -i gpx -f myfile.gpx -x transform,rte=wpt
```

In the other case, type:

```
gpsbabel -t -i gpx -f myfile.gpx -x reverse -o etc etc...
```

in *GPSPrune* you could do the same, selecting the track as a range of points and then clicking on Reverse Range in the Range menu.

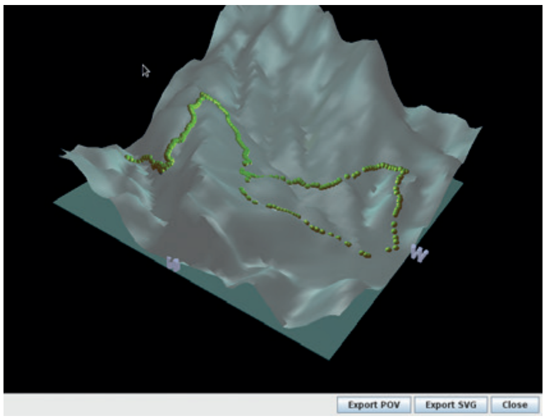
Removing cruft

As we said, it is extremely common for a real GPX track to contain many points that shouldn't be there. Can we find and remove them efficiently?

Viking can only remove points with exactly the same position or time. *GPSPrune* can create a new point whose coordinates are the average of all the points you had previously selected. This program also has four cleaning options under the Track > Compress Track menu option, called Duplicate, Wacky, Singleton and Nearby. Duplicate, as the name suggests, only finds exact duplicates. Wacky and Singleton identify outliers – points too far from the adjacent ones to be genuine. The Nearby filter marks as likely duplicates all points that are closer to each other than a given 'span factor', which is related to the total area covered by the whole track. Whatever option you choose, you must then click on Track > Delete Marked Points to remove the points that were selected.

Interpolation

As well as having to get rid of errant points, you may find that some of your tracks contain too few points to be useful. Points that are one or more mile apart, for example, may be too few to get accurate directions on the ground. The solution in these case is to interpolate – to tell your software to automatically fill the track with more points. This is quite easy in *GPSPrune* (look for 'Interpolate' in the Range menu) and with *GPSBabel*. The latter can interpolate between every



Once you have finished drawing or cleaning up a GPX track or route, getting a 3D version of it that you can rotate as you want only takes a few more clicks in GPSPrune.

two adjacent points that are too far either in distance (10km in the first example) or time (10 minutes in the second one):

```
gpsbabel -i ... -x interpolate,distance=10k -o ...
gpsbabel -i ... -x interpolate,time=600 -o ...
```

Whatever tool you use, it will place the new points along a straight line, at regular intervals. Therefore, you may need to adjust some of those points by moving them manually in the GUI editors.

Going 3D

Since we don't live in a flat world, elevation matters. You can't estimate speed or travelling time, or draw a 3D map of a track if you don't know the altitude of all its points. GPS navigators will add such data automatically, but tracks you drew in your computer need to get them from some other source. The easiest way we found to add elevations to tracks and routes is the Online > Get Altitudes From SRTM function in *GPSPrune*.

How to create tracks from scratch (and why)

Drawing tracks and routes from scratch isn't just fun: sometimes it's absolutely necessary. *Viking* is great for drawing tracks and routes, thanks to its layer system. Here's how to use it: first, pan and zoom the Default Map (OSM) until you are exactly where you want to draw. Next, select Layers > New Map Layer and choose Bing, or any other set of satellite/aerial photographs. Then select the Default Map layer in the left-hand pane, click on Layers > Properties and change its transparency, setting its alpha value to something around 120 or 130.

By doing this, that is, by laying OSM right over actual photographs of the area, the actual drawing will be really accurate and easy: add and select a new 'TrackWaypoint' layer, click on the Create Track button, and start clicking where the track should pass. If you reach the borders of the window, click on the Pan button to move the map, and then again on Create Track. When you're done, click on the track name in the left-hand pane, select Finish Track and save. You can also add existing tracks or waypoints in their own layers, to use them as reference.

What's the point of a good track if you can't show it off? The 3D-View panel of *GPSPrune* (which requires the *j3d* library) can display tracks on a 3D terrain that you can rotate at will and export in various graphic formats. If privacy isn't a problem, uploading a GPX file to **http://utrack.crempa.net** will produce reports with several statistics, and better graphics than those you can get from *GPSPrune* or *Viking*.

Geotagging

Once you have a good GPX track complete with timestamps, you can use it to automatically geotag pictures. *Viking*, *GPSPrune* and photograph managers such as *Digikam* all have interfaces that enable you to write the geographical coordinates into each picture of a given folder the point of the track that has the same, or the closest timestamp.

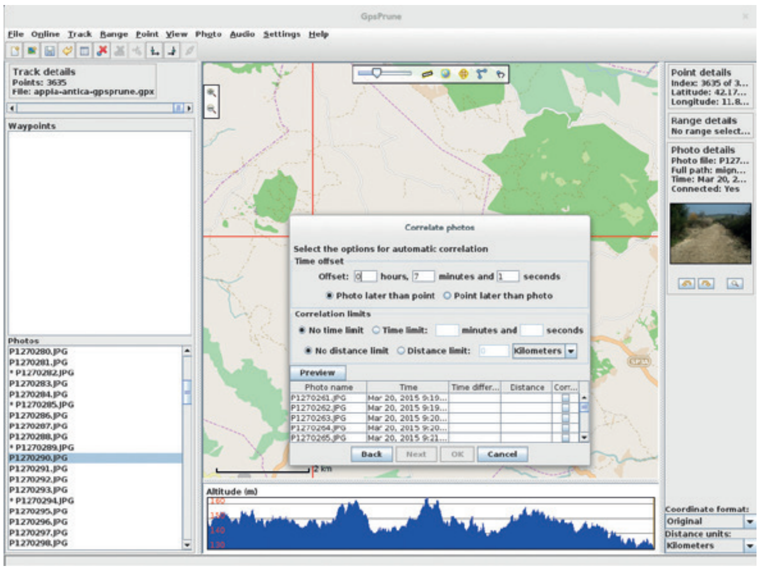
3D images will appear much faster, printouts will be much clearer and geotagging really accurate, only if you've created te best tracks possible. To continue on this path, first read the right stuff – all the *GPSBabel* examples here are samples of the great documentation at **www.gpsbabel.org**. Other must-reads are the GPX homepage (**www.topografix.com/gpx.asp**) and the tutorials at **www.gpsvisualizer.com**, which is also a web-based GPX editor. Invite all your relatives, high school friends and so on to go there and draw the places and walks you did together in the old days, and send you the results. With any luck, and without asking anybody to install any software, you should receive lots of GPX files to combine and reuse as shown in this tutorial!

GeoJson and KML

While GPX is the *lingua franca* of GPS, there are at least two other geographic formats that you should know about: GeoJSON (**http://geojson.org**) and the Keyhole Markup Language (KML, **https://developers.google.com/kml/documentation**). The first is widely used in OpenStreetMap applications such as *Umap*, and the second in Google Maps and Google Earth.

GeoJSON can describe all kinds of location-related data, from country boundaries to the history of tourist locations. You can move from GPX to GeoJson and *vice versa* (see **https://developers.google.com/kml/documentation** and, respectively, **https://github.com/tyrasd/togpx**), but the second procedure often loses information, because GeoJson is much more complex than GPX.

KML is another XML format for waypoints, tracks and routes, with the important difference that, unlike GPX, it was specifically designed for annotating maps. This is why KML files come in two flavours: **.kml** and **.kmz**. The first of these are plain text files, while files ending **.kmz** are ZIP archives that may also contain icons, full text documents or image overlays. *GPSBabel* can, of course, convert data from one format to the other.


GpsPrune
File Online Track Range Point View Photo Audio Settings Help
Waypoints
Altitude (m)
2 km
Preview
Back
Next
OK
Cancel

Track details
Points: 3635
File: appia-antica-gpsprune.gpx

Photos
P1270280.JPG
P1270281.JPG
* P1270282.JPG
P1270283.JPG
P1270284.JPG
* P1270285.JPG
P1270286.JPG
P1270287.JPG
P1270288.JPG
* P1270289.JPG
P1270290.JPG
P1270291.JPG
P1270292.JPG
P1270293.JPG
* P1270294.JPG
P1270295.JPG
P1270296.JPG
P1270297.JPG
P1270298.JPG

Correlate photos
Select the options for automatic correlation
Time offset
Offset: 0 hours, 7 minutes and 1 seconds
[✓ Photo later than point] Point later than photo
Correlation limits
[✓ No time limit] Time limit: minutes and seconds
[✓ No distance limit] Distance limit: 0 Kilometers

Photo name	Time	Time differ...	Distance	Corr...
P1270261.JPG	Mar 20, 2015 9:19...			
P1270262.JPG	Mar 20, 2015 9:19...			
P1270263.JPG	Mar 20, 2015 9:20...			
P1270264.JPG	Mar 20, 2015 9:20...			
P1270265.JPG	Mar 20, 2015 9:21...			

Point details
Index: 3635 of 3...
Latitude: 42.17...
Longitude: 11.8...

Range details
No range select...

Photo details
Photo file: P127...
Full path: mign...
Time: Mar 20, 2...
Connected: Yes

Coordinate format: Original
Distance units: Kilometers

Viking and *GPSPrune* can geotag all the points in a photograph automatically, or let you manually place each one of them in the right place.

Marco Fioretti is a free software campaigner and one of the instigators behind the Digital DIY project – www.didiy.eu.

LINUXVOICE TUTORIAL

LISP: ELEGANT PARENTHESES FOR A MORE... CIVILISED AGE

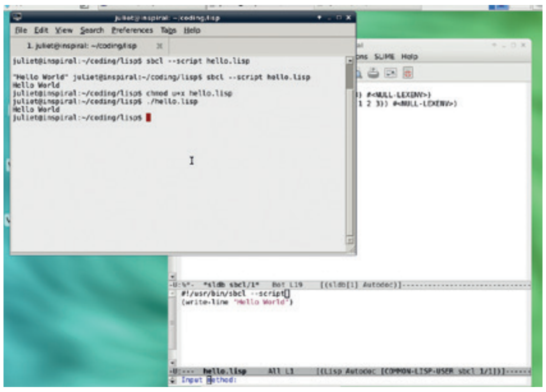
JULIET KEMP

A language created for artificial intelligence research sounds modern – but Lisp comes from 1950s...

Lisp is the third of the big four early languages, a year younger than FORTRAN and about the same age as ALGOL. But unlike ALGOL, Lisp has been in use ever since that its first implementation, and is undergoing something of a revival in recent years. Lisp stands for List Processing, and as you'll see when you read on, lists are what Lisp is all about.

John McCarthy began thinking about an algebraic language in 1956, and he and colleagues spent the next couple of years looking at conditional expressions and the possibility of writing a list processing language in FORTRAN. When McCarthy started at MIT in 1958, he began both working on an artificial intelligence project, and implementing Lisp, based around the vital idea of using a list for both code and data.

Initially, his team postponed the idea of writing a compiler in favour of hand-compiling Lisp functions and subroutines in assembly language, making for a kind of Lisp 'environment' used by stringing functions together. Various simplifying decisions which resulted from this made Lisp into a neat way of describing computable functions, and McCarthy published a paper in 1960 focussing on this. The paper showed



```
juliet@inspiral:~/coding/lisp$ sbcl --script hello.lisp

"Hello World" juliet@inspiral:~/coding/lisp$ sbcl --script hello.lisp
Hello World
juliet@inspiral:~/coding/lisp$ chmod u+x hello.lisp
juliet@inspiral:~/coding/lisp$ ./hello.lisp
Hello World
juliet@inspiral:~/coding/lisp$
```

```
)) #<NULL-LEXENV>)
1 2 3)) #<NULL-LEXENV>)
```

```
-U:%*-  *sldb sbcl/1*   Bot L19   [(sldb[1] Autodoc)]----------
#!/usr/bin/sbcl --script
(write-line "Hello World")

-U:---  hello.lisp    All L1    [(Lisp Autodoc [COMMON-LISP-USER sbcl 1/1])]------
Input Method:
```

Lisp Slime; note the use of the quote evaluator with **car** and **cdr**; when I missed it out the first time, it produced an error. The quote tells Lisp that this is data not code.

that one could build a Turing-complete algorithm language with only a very few simple operators and a function notation. (See the boxout for more detail.)

As part of this paper (**www-formal.stanford.edu/jmc/recursive/recursive.html**), McCarthy described an eval function, written in Lisp. Steve Russell noticed that this would, if implemented, act as a Lisp interpeter, and went ahead and implemented it on the

The theory of Lisp

The basic Lisp building block is an expression, which is either an atom (a sequence of letters, like **name**) or a list (() is a zero-length list; **(a b (foo bar))** is a 3-element list, one of whose members is a 2-element list). (Note that to evaluate these in *Slime* you'll need to quote them; see below.)

Expressions also have values: an expression **e** returns a value **v**. McCarthy established seven basic (axiomatic) expressions. In each case the expression is a list, with the operator as the first element of the list and the arguments the rest of the list.

1 **(quote x)** or **(' x)** returns x. **quote** protects a list from being evaluated, meaning it is treated as data instead of as code. Since in Lisp both code and data are lists, **quote** enables us to tell the difference.

2 **(atom x)** returns **t** if **x** is an atom, or the empty list. If **x** is a list, it returns **()**. In Lisp, **t** represents truth and **()** represents falsity.

3 **(eq x y)** returns **t** if **x** and **y** both evaluate to **()** or to the same value, and **()** otherwise.

4 **(car x)** returns the first element of the list **x**.

5 **(cdr x)** returns everything after the first element of the list **x**. (**car** and **cdr** originate from the initial IBM 704 hand-coding of Lisp, which used two assembly language macros to decompose lists. **car** stood from Contents of the Address Part of Register Number, and **cdr** for Contents of the Decrement Part of Register Number.)

6 **(cons x y)** concatenates the elements of the list **y** after the value of **x**.

7 **(cond (p1e1) ... (pnen))** evaluates each **p** expression in turn until it finds one which evaluates true. The corresponding **e** expression is then returned as the value of the whole **cond** expression.

Finally, McCarthy defined a function expression:

```
((lambda (x y) e) a b)
```

x and **y** are parameters, **e** is an expression, and **a** and **b** are values (which may be expressions that first must be evaluated to give their values). **e** is evaluated with **a** substituted for **x** and **b** substituted for **y** wherever they occur. A function can have as many parameters and values as you like.

Functions can also refer to themselves using the notation **label**, but in Lisp this is usually written as:

```
(defun f (x y z) e)
```

then the function is called as **(f a b c)** with **a** substituted for **x** in **e**, and so on. Paul Graham's excellent paper (**www.paulgraham.com/rootsoflisp.html** – to which I owe thanks for the above) goes on to explain in detail how this setup enables McCarthy to define a function which evaluates any Lisp expression; so Lisp can interpret itself.

Modern Lisp dialects

There are currently three well-known dialects of Lisp (and a collection of less-well-known ones as well).

Common Lisp has a large language standard with lots of built-in types, functions, macros, etc, and an object system. It has also borrowed some features from the Scheme programming language.

Scheme was designed to be clear, simple, and minimalist. It can express numerous different programming styles. It does however have a smaller set of standard features than Common Lisp.

Clojure compiles directly to Java Virtual Machine bytecode, and also targets the Python and Ruby VMs. It's influenced by Haskell and keen on immutability. It provides direct access to Java libraries, to speed up links with Java code.

There are also a handful of Lisp dialects used as scripting languages. The best known is of course *Emacs* Lisp, but others include the embedded *Gimp* Script-fu, and embedded languages in *Audacity*, *AutoCAD*, and other CAD apps. Lisp is used on running on top of JVMs in Clojure, and Common Lisp is used in financial institutions.

Any of the major three dialects can be useful, and any of them will teach you Lisp basics. Here I'll use Common Lisp.

project's IBM 704. (Somewhat to McCarthy's surprise; until Russell succeeded, he didn't believe that the eval in the paper could be translated into machine code.) This was slow, but it was a genuine Lisp interpreter. It was an exciting step forward, but McCarthy has commented that the existence of the interpreter did also have the less-desirable effect of freezing the existing form of the language. The first complete compiler (also written in Lisp) was also implemented at MIT, in 1962, by Tim Hart and Mike Levin.

Lisp development continued, and during the 1960s and 1970s, Lisp was heavily used in AI research, which required significant processor time and vast (for the time) memory space. This was made more difficult because machines were usually optimised for Fortran or assembler, rather than for Lisp. Initially, too, Lisp, being interpreted rather than compiled, ran much more slowly; and garbage collection was a problem, though this was improved by the routines developed at MIT by Daniel Edwards.

Trying to resolve these issues, in 1973, two MIT AI researchers built a machine with some Lisp operations as part of the hardware rather than software. Towards the end of the decade, this led to the creation of Symbolics, a commercially funded company aiming to produce Lisp machines. Various other manufacturers followed suit through the early/mid 1980s. Lisp machines were briefly successful, and they really were very good at what they did, but the advent of the microcomputer meant that by the end of the decade they were largely obsolete. Symbolics no longer exists as a computer manufacturer, but a private company owns its assets and continues to sell the Genera Lisp machine OS, which runs on various Symbolics Lisp machines and DEC Alpha. It also sells the Macsyma computer algebra system. (**symbolics.com** was the first ever .com domain, registered on 15 March, 1985.)

Interest in Lisp declined in the 1990s, but recently there has been an increase of interest again. See the boxout for more on currently available dialects of Lisp.

For a rundown of Lisp's early history from McCarthy himself, check out **www-formal.stanford.edu/jmc/history/lisp/lisp.html**.

Installing and getting started

The most popular Linux implementation of Common Lisp is SBCL (Steel Bank Common Lisp). You can make do with just this (install via your package manager or from the website), but installing Emacs and Slime – the Superior Lisp Interaction Mode for Emacs – as well will give you a more functional dev environment.

Once you've installed them, add these lines to your **~/.emacs**:

```
;; Set up Common Lisp
(add-to-list 'load-path "/usr/share/common-lisp/source/slime/")
(setq inferior-lisp-program "/usr/bin/sbcl")
(require 'slime)
(slime-setup)
;; Use highlight colors
(global-font-lock-mode t)
```

Start emacs, type M-x slime, and you'll see a CL-USER (or *) prompt. This is your Lisp command prompt. Try typing some things:

```
> (* 2 3)
6
> (car '(1 2 3))
1
> (cdr '(1 2 3))
(2 3)
```

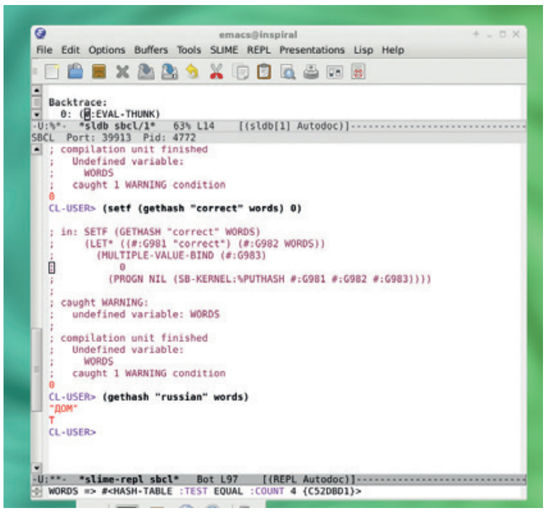
As discussed in the boxout, everything in Lisp is a list, surrounded by brackets. Operators go at the start of a list.

What about Hello World?

```
> (print "Hello World")
"Hello World"
"Hello World"
```

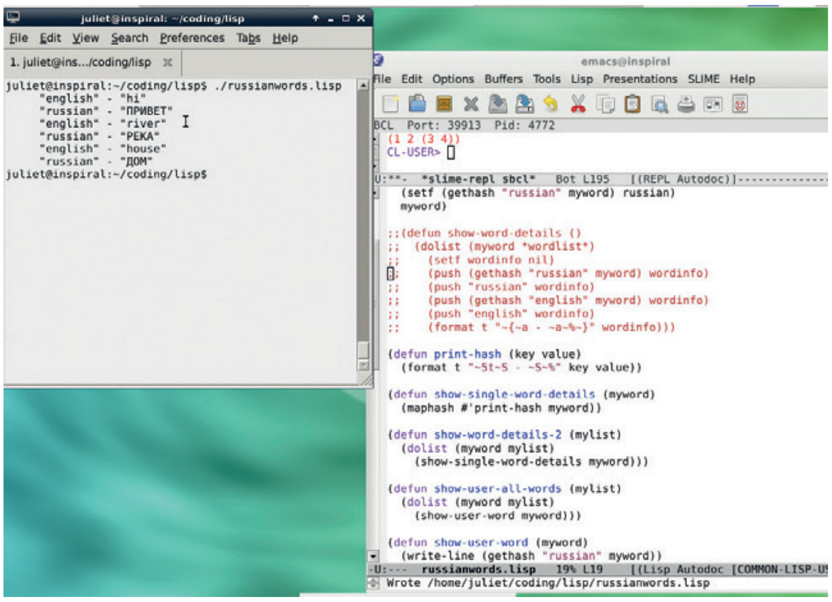
The interpreter prints the string twice, because it executes the function (printing the string), then outputs the return value (which is also the string).

To run a 'real' program (one that won't go away when you quit the interpreter), open up a new file


emacs@inspiral
File Edit Options Buffers Tools SLIME REPL Presentations Lisp Help

```
Backtrace:
  0: ((:EVAL-THUNK)
-U:%*-  *sldb sbcl/1*   63% L14    [(sldb[1] Autodoc)]
SBCL  Port: 39913  Pid: 4772
; compilation unit finished
;   Undefined variable:
;     WORDS
;   caught 1 WARNING condition
0
CL-USER> (setf (gethash "correct" words) 0)

; in: SETF (GETHASH "correct" WORDS)
;     (LET* ((#:G981 "correct") (#:G982 WORDS))
;       (MULTIPLE-VALUE-BIND (#:G983)
;           0
;         (PROGN NIL (SB-KERNEL:%PUTHASH #:G981 #:G982 #:G983))))
;
; caught WARNING:
;   undefined variable: WORDS
;
; compilation unit finished
;   Undefined variable:
;     WORDS
;   caught 1 WARNING condition
0
CL-USER> (gethash "russian" words)
"дом"
T
CL-USER>

-U:**-  *slime-repl sbcl*   Bot L97    [(REPL Autodoc)]
WORDS => #<HASH-TABLE :TEST EQUAL :COUNT 4 {C52DBD1}>
```

Trying out hash tables. Note the undefined variable warning; the code does work anyway, but we'll look at defining variables a bit later.


juliet@inspiral: ~/coding/lisp
File Edit View Search Preferences Tabs Help
1. juliet@ins.../coding/lisp
emacs@inspiral
File Edit Options Buffers Tools Lisp Presentations SLIME Help

```
juliet@inspiral:~/coding/lisp$ ./russianwords.lisp
     "english" - "hi"
     "russian" - "ПРИВЕТ"
     "english" - "river"
     "russian" - "РЕКА"
     "english" - "house"
     "russian" - "ДОМ"
juliet@inspiral:~/coding/lisp$
```

```
BCL  Port: 39913  Pid: 4772
(1 2 (3 4))
CL-USER>
U:**-  *slime-repl sbcl*   Bot L195   [(REPL Autodoc)]--------------
  (setf (gethash "russian" myword) russian)
  myword)

;;(defun show-word-details ()
;;  (dolist (myword *wordlist*)
;;    (setf wordinfo nil)
;;    (push (gethash "russian" myword) wordinfo)
;;    (push "russian" wordinfo)
;;    (push (gethash "english" myword) wordinfo)
;;    (push "english" wordinfo)
;;    (format t "~{~a - ~a~%~}" wordinfo)))

(defun print-hash (key value)
  (format t "~5t~S - ~S~%" key value))

(defun show-single-word-details (myword)
  (maphash #'print-hash myword))

(defun show-word-details-2 (mylist)
  (dolist (myword mylist)
    (show-single-word-details myword)))

(defun show-user-all-words (mylist)
  (dolist (myword mylist)
    (show-user-word myword)))

(defun show-user-word (myword)
  (write-line (gethash "russian" myword))
-U:---  russianwords.lisp   19% L19    [(Lisp Autodoc [COMMON-LISP-US
Wrote /home/juliet/coding/lisp/russianwords.lisp
```

Outputting all the data. Note the DIY loop code commented out in the code window, showing another (less neat) way to do it.

hello.lisp:

```
#!/usr/bin/sbcl --script
(write-line "Hello World")
```

Run it from the command line with **chmod u+x hello.lisp; ./hello.lisp**. Note **s WRITE-LINE** outputs the string with a newline and with no quotes around it, unlike **PRINT**.

A little more complicated

For a more complicated project, let's try a foreign language flashcards game. I've gone for Russian (which means Unicode; see the box). It'll show you a word in Russian, you type in the English equivalent, and it tells you whether you were right or wrong.

The first step is to define a database entry. We're going to use a hash table, which is a data type that contains a set of keys and their associated values. This *Slime* code creates a hash table word, with keys **english** and **russian**:

```
> (setq word (make-hash-table :test 'equal))
> (setf (gethash "english" word) "house")
> (setf (gethash "russian" word) "дом")
```

When making the hash table, passing in **:test 'equal** defines the equality test, so Lisp knows how to test the lookup value against the stored hash keys (in order to return the correct value). **GETHASH** returns a key value, and **SETF** sets it (in this case, creating it first). To return a value, use **GETHASH**, which returns both the value, and **T** (true, because found):

```
> (gethash "russian" word)
"дом"
T
```

This will be a bit time consuming to do by hand for every entry, so let's create a function to handle it:

```
(defvar myword)
(defun make-word (english russian)
 (setf myword (make-hash-table :test 'equal))
 (setf (gethash "english" myword) english)
 (setf (gethash "russian" myword) russian)
 myword)
```

DEFVAR defines the dummy variable **myword** without setting it, so we don't get a warning about it. **DEFUN** shows that we're defining a new function. The next symbol after that is the function name, **make-word**, and the list after that is the parameter list. Everything after that is the body of the function. Mostly, this just consists of putting the parameters into the **SETF** calls we used above. However, note that final line, **myword**. The return value of a Lisp function is the last expression evaluated. We want that to be the new hash table itself, and without that final line, it would be the return value of the last call to **SETF**.

The general form of a function definition, then, looks like this:

```
(DEFUN name (parameter_list)
 (function_body))
```

Like everything else in Lisp, it's a list.

That's a single database entry defined; we want lots of them, so we need a global database variable, and a function to add entries to it:

```
(defvar *wordlist* nil)
(defun add-word (myword) (push word *wordlist*))
```

DEFVAR is the macro that defines a variable (but only if it hasn't already been defined; to override an existing variable use **DEFPARAMETER**), and the Lisp naming standard for global variables is ***name***. The **add-word** function simply uses the **PUSH** macro to add its single parameter to ***wordlist*.**

Add a few words, then (if in *Slime*) type ***wordlist*** to see them all:

```
(add-word (make-word "house" "дом"))
(add-word (make-word "hello" "привет"))
(add-word (make-word "river" "река"))
```

If you're not in *Slime*, you'll want a way to output the list (and if you are in *Slime*, you'll want a way that doesn't just give you lots of hash parameters). The best bet for this is the function **maphash**:

```
(defun print-hash (key value)
 (format t "~5t~S - ~S~%" key value))

(defun show-single-word-details (myword)
 (maphash #'print-hash myword))

(defun show-all-word-details (mylist)
 (dolist (myword mylist)
  (show-single-word-details myword)))

(show-all-word-details *wordlist*)
```

print-hash defines how to output a key/value pair, using **FORMAT**. **FORMAT** controls string format

Unicode

To enter Russian letters (or any other Unicode symbol) in Emacs, type **C-x 8 RET** then the Unicode hex code or full name (Tab-completion works):

```
C-x 8 RET 0434
C-x 8 RET CYRILLIC SMALL LETTER DE
```

both give д. Modern Linux boxes should automatically have Unicode support in *XEmacs* and in the terminal.

output and is therefore, as with all format functions in all languages, somewhat confusing to the untrained eye. **t** outputs to standard out; **~5t** tabs in 5 spaces, **~S** consumes the arguments after the **format** string (here key first and value second), and **~%** is the newline.

show-single-word-details passes the **print-hash** function into **maphash** together with the hash to print. Note that **print-hash** is quoted here, so that it is passed on as data, rather than being evaluated there and then. Finally, **show-all-word-details** uses **dolist** to iterate over the list and show each word in turn.

User input

Finally, we need the bit where it asks you what the word means. Here's the code:

```
(defvar answer)
(defun show-user-word (myword)
 (write-line (gethash "russian" myword))
 (write-line "Enter English translation: ")
 (setq answer (read))
 (cond
   ((equalp (string-trim "\n" answer) (gethash "english" myword))
    (write-line "Correct!"))
   (t (write-line "Wrong!"))))

(defun show-user-all-words (mylist)
 (dolist (myword mylist)
   (show-user-word myword)))

(show-user-all-words *wordlist*)
```

In **show-user-word** we output the Russian part of the hash, output a message to the user, then use **read** to get the user input and **setq** to set it.

cond is a neat function, but it can look a little complicated. Its basic structure is:

```
(cond ((predicate1) (things to do in this case))
      ((predicate2) (things to do in this case))
      (t (else this)))
```

It works, as you can see, a lot like if/then, with the **t** acting as an always-true condition that provides an 'else' (this part is optional; you don't always want an 'else'). However, it has two really useful points:

1. You can have multiple statements in the "things to do" part (just put them all in the appropriate brackets).
2. Lisp doesn't have an **elseif**, so if/then can only have one **if**, one **then**, and an **else**. **cond** can have as many tests as you like.
3. If you have multiple tests, more than one of them may be true for each thing tested (eg testing if a number is less than 10, less than 50, less than 100), and you want to go through all the tests, not drop out at the first 'true', **cond** does this too.

Here it's more straightforward: we test whether the string is equal to the 'english' value from the hash (note the use of **string-trim** to trim off the newline at the end of the user input), and output **Correct** or **Wrong** accordingly.

show-user-all-words then iterates over the list, so the user is asked about each word in turn.

If you want to carry on experimenting with Lisp, here are a few improvements you could try making to this code:

- Ask the words in a random order.
- Keep track of how often a word is asked or is answered correctly. (You'll need to find some way to read data in and out to keep track of this over multiple instances of running the code.)
- Pick out only the words that haven't been asked recently, or which have been answered wrongly in the past. Try using **REMOVE-OR-NOT s**with a lambda function to do this.
- Allow the user to add more words.

For more information on programming Lisp, try the very readable introduction (free online) *Practical Common Lisp*.

Once you get used to the syntax, Lisp has a real elegance, and there are certain classes of problem that it is truly well suited to and by its current resurgence as an active language. Try out Scheme or Clojure if you want a change from Common Lisp; or one of the many other minor variants. For a language that's now coming up on 60 years old, it's impressively evergreen. LV

Lambda functions

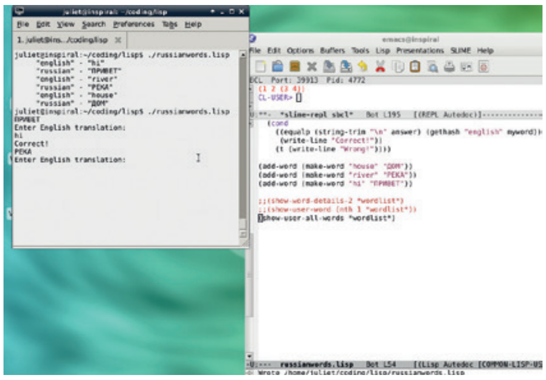
A particularly neat Lisp-ism, and one that has made its way into many other languages, is lambda functions, or anonymous functions. (The word 'lambda' is used after Church's lambda calculus.) Here's a very simple mathematical example:

```
* (remove-if-not #'(lambda (x) (= 3 x)) '(1 2 3 4 5))
(3)
```

REMOVE-IF-NOT simply returns a list of anything that matches its function parameter (removes the things that do match). So (**remove-if-not #'oddp '(1 2 3)**) would return (1 3). Here, the lambda function is effectively a temporary function:

```
(lambda (x) (= 3 x))
```

lambda (x) means "anything that matches 3", and so the function as a whole returns (3). A lambda function, then, is a neat way of creating a function to plug into a bigger function, and you'll find them all over Lisp code as well as in other languages.


juliet@inspiral: ~/coding/lisp
File Edit View Search Preferences Tabs Help
1. juliet@ins.../coding/lisp
emacs@inspiral
File Edit Options Buffers Tools Lisp Presentations SLIME Help

```
juliet@inspiral:~/coding/lisp$ ./russianwords.lisp
     "english" - "hi"
     "russian" - "ПРИВЕТ"
     "english" - "river"
     "russian" - "РЕКА"
     "english" - "house"
     "russian" - "ДОМ"
juliet@inspiral:~/coding/lisp$ ./russianwords.lisp
ПРИВЕТ
Enter English translation:
hi
Correct!
РЕКА
Enter English translation:
```

```
ECL  Port: 39913  Pid: 4772
(1 2 (3 4))
CL-USER>
U:**-  *slime-repl sbcl*   Bot L195   [(REPL Autodoc)]
  (cond
    ((equalp (string-trim "\n" answer) (gethash "english" myword))
     (write-line "Correct!"))
    (t (write-line "Wrong!"))))

(add-word (make-word "house" "ДОМ"))
(add-word (make-word "river" "РЕКА"))
(add-word (make-word "hi" "ПРИВЕТ"))

;;(show-word-details-2 *wordlist*)
;;(show-user-word (nth 1 *wordlist*))
(show-user-all-words *wordlist*)
-U:---  russianwords.lisp   Bot L54   [(Lisp Autodoc [COMMON-LISP-US
Wrote /home/juliet/coding/lisp/russianwords.lisp
```

Testing in action!

Juliet Kemp is a scary polymath, and is the author of Apress's *Linux System Administration Recipes*.

LINUX VOICE TUTORIAL

GET STARTED WITH DOCKER

BEN EVERARD

Make sure your software runs consistently in any environment by packaging it up with Docker.

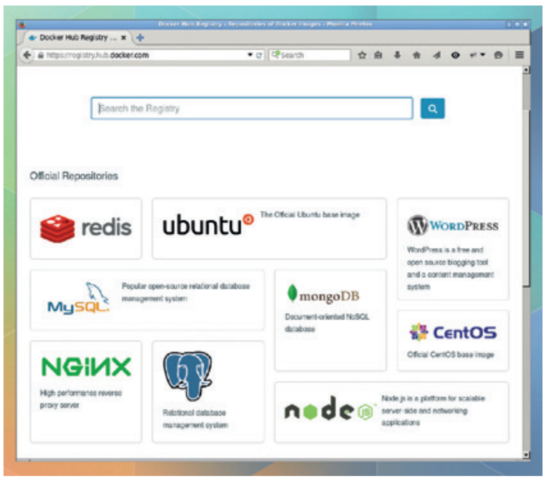
WHY DO THIS?
- Ensure consistency when setting up multiple environments
- Enhance your CV with this great buzzword!
- Automation is always good

Docker enables you to bring together your environment and your code into a single package that you can take and run on any Linux distro. You don't have to worry about which libraries are installed on the host, or which versions of what other software are available: you just start your package with Docker and it will work.

It does this by using containers to encapsulate a separate environment running on top of the host kernel. This environment runs parallel to the host operating system, but is partitioned off by the Docker tools. It has its own filesystem and own software that's nothing to do with the operating system running on the host system.

Let's look at a quick example to see why this is useful. Suppose a company is building a new web app on top of Node.js. They've got a server running Red Hat Enterprise Linux, a developer running Arch Linux and a freelancer running Ubuntu. There are Node.js packages for all of these different Linuxes, so they could all install Node.js and start programming.

The developer using Arch makes a start and sends his work to the freelancer. However, the freelancer finds that it doesn't run properly. After some investigation, they find that it's because Arch has a newer build of Node.js than Ubuntu. The freelancer builds the latest version of Node from source, and they get going.


Docker Hub Registry - Repositories of Docker Images - Mozilla Firefox
Docker Hub Registry ...
https://registry.hub.docker.com
Search the Registry
Official Repositories
redis
ubuntu
The Official Ubuntu base image
WordPress
WordPress is a free and open source blogging tool and a content management system
MySQL
Popular open-source relational database management system
mongoDB
Document-oriented NoSQL database
CentOS
Official CentOS base image
NGINX
High performance reverse proxy server
Relational database management system
node
Node.js is a platform for scalable server-side and networking applications

The Docker Hub holds a wide variety of official and user-contributed images to make it easy to get started.

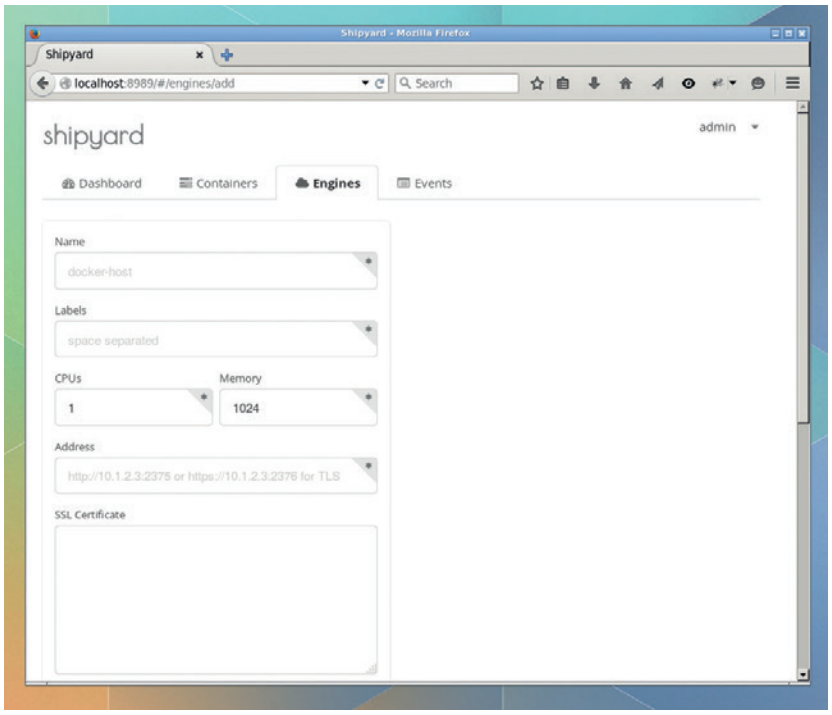
Once they've developed the web app, they deploy it to the server, but again it doesn't work. The package for RHEL doesn't have the features they need. They've dealt with this before, so they try to build Node.js from source. However, it needs newer versions of some libraries than are available in RHEL. They end up having to build several packages from source in order to make the app work. This means that they can't use the package manager to stay up to date, and have to continue re-building the app every time a security patch is released.

The Docker way

Obviously this isn't great, but it's also not necessarily the way it had to work. The developers and server maintainers could have standardised on a distribution from the start – one that they could all have running in a virtual environment if they didn't want to install it natively – and this would have bypassed the issues with versioning. However, there would still be work to make sure that it was set up in the same way, that they had the same repositories available, and have the same software installed.

With Docker, the process works like this:
- Create a Docker image for the project.
- All developers/testers work with this Docker image.
- Once the software is developed, deploy this image to the server.

Shipyard is a Docker web front-end that is conveniently distributed via Docker.



Name: docker-host
Labels: space separated
CPUs: 1
Memory: 1024
Address: http://10.1.2.3:2375 or https://10.1.2.3:2376 for TLS
SSL Certificate

By using the same Docker image, you guarantee that everything is the same on every developer's instance, on every tester's instance, and the final instance that is deployed. There are also a few added benefits, such as that Docker can be used to package software to be released in a distro-agnostic way.

That's enough about why to use Docker, let's get started with how to use it. Once you've installed Docker through your distro's package manager (note: it could be in a package called **docker.io**), you can start a new Docker container with:

```
docker run -it ubuntu /bin/bash
```

The first thing you should notice about this is that it doesn't matter what distro you're running on, this will get and run the latest version of Ubuntu. The download should be fairly small by distro standards (about 200MB). That's because it's just a bare-bones version of Ubuntu, and you'll need to install whatever applications you want on top of it. The **-t** option tells Docker that you want a TTY (console), and **-i** tells Docker to keep STDIN attached (**-it** are the standard options for starting an interactive session) and the final argument tells Docker what command you want to run; in this case, it just starts a *Bash* shell.

Consistent environment

Once Docker's downloaded what it needs, it'll launch an interactive shell in the Ubuntu container. In here you can make changes and install whatever you like. However, any changes you make will only be for that session. If you exit the shell with the **exit** command, then restart it in the same way, you won't get any of your changes. Docker works with images, and it will always start the image that's saved, rather than as it last was. This means you know you're always starting in a consistent environment.

You can save your changes to a new Docker image by committing it. To do this, you need to know two things: the ID of the container and the name you want to commit to.

If you don't have an Ubuntu docker container still running, start it as you did before, then make a change that you can check is saved:

```
cd ~
mkdir test
```

While that container is still running, open another terminal window, and type:

```
docker ps
```

This will list all the currently running Docker containers. It will probably only have one entry so far. That's the container you need. It will have an alphanumeric ID assigned to it, which is the ID you need. The name to save the container to can be any you like – we'll just call ours **testubuntu**. You can save your changes with:

```
docker commit <image-id> testbubuntu
```

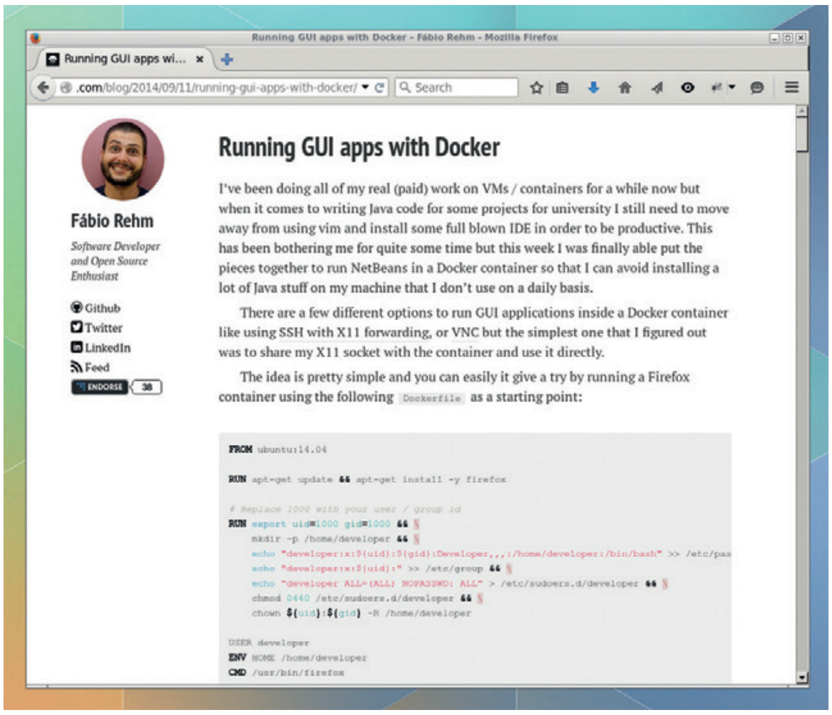
Where **<image-id>** is the ID from the previous **docker ps** output. This saves the current state of that image, so you should have it running in the current state you want it in. Once you've committed it, you can exit the original Ubuntu session, and go back with:

```
docker runt -it testubuntu /bin/bash
```

Using this approach, you can build an image with whatever software you want. However, it's not very convenient. For example, if you want to rebuild your server using Debian rather than Ubuntu, you'd have to manually go through all the steps again. A better way of creating your own Docker images is by building them using Dockerfiles, which state the start image, and then all the steps needed to install everything you want. Docker can build from these file by running through them like a script.

To recreate the same image as above using a Dockerfile, you'll first need to create a directory in which to do the build. The directory name is the

You can run GUI applications in Docker. This is *Firefox* running in Docker showing a web page about running *Firefox* in Docker (**http://fabiorehm.com/blog/2014/09/11/running-gui-apps-with-docker**).



Running GUI apps with Docker - Fábio Rehm - Mozilla Firefox

.com/blog/2014/09/11/running-gui-apps-with-docker/

Running GUI apps with Docker

Fábio Rehm

Software Developer and Open Source Enthusiast

I've been doing all of my real (paid) work on VMs / containers for a while now but when it comes to writing Java code for some projects for university I still need to move away from using vim and install some full blown IDE in order to be productive. This has been bothering me for quite some time but this week I was finally able put the pieces together to run NetBeans in a Docker container so that I can avoid installing a lot of Java stuff on my machine that I don't use on a daily basis.

There are a few different options to run GUI applications inside a Docker container like using SSH with X11 forwarding, or VNC but the simplest one that I figured out was to share my X11 socket with the container and use it directly.

The idea is pretty simple and you can easily it give a try by running a Firefox container using the following `Dockerfile` as a starting point:

```
FROM ubuntu:14.04

RUN apt-get update && apt-get install -y firefox

# Replace 1000 with your user / group id
RUN export uid=1000 gid=1000 && \
    mkdir -p /home/developer && \
    echo "developer:x:${uid}:${gid}:Developer,,,:/home/developer:/bin/bash" >> /etc/pas
    echo "developer:x:${uid}:" >> /etc/group && \
    echo "developer ALL=(ALL) NOPASSWD: ALL" > /etc/sudoers.d/developer && \
    chmod 0440 /etc/sudoers.d/developer && \
    chown ${uid}:${gid} -R /home/developer

USER developer
ENV HOME /home/developer
CMD /usr/bin/firefox
```

Docker Hub The joys of sharing

One of the great features of Docker is the ability to share your images with other people. This means there are loads of ready made images for you to use. There are also official images from many projects including Linux distros (such as Ubuntu and CentOS), web apps (such as WordPress), and server software (such as Node.js or *MongoDB*). All these are hosted through the Docker Registry (**https://registry.hub.docker.com**).

You can install images from these repositories by calling them by name in **docker run** commands (like we did in the main text with the Ubuntu images), or by using **docker pull <image-name>** (which downloads the image but doesn't run it).

As well as the official images, you can get images that users have uploaded. This is done by putting **<user-name>/** in front of the image name. For example, if you don't want to go through the rest of this tutorial and learn how to create an OwnCloud Docker container, you can download one ready-made. A user "l3iggs" has an image called **owncloud** that they keep as the most up-to-date version. You can run this with:

```
docker run -d -p 7000:80 l3iggs/owncloud
```

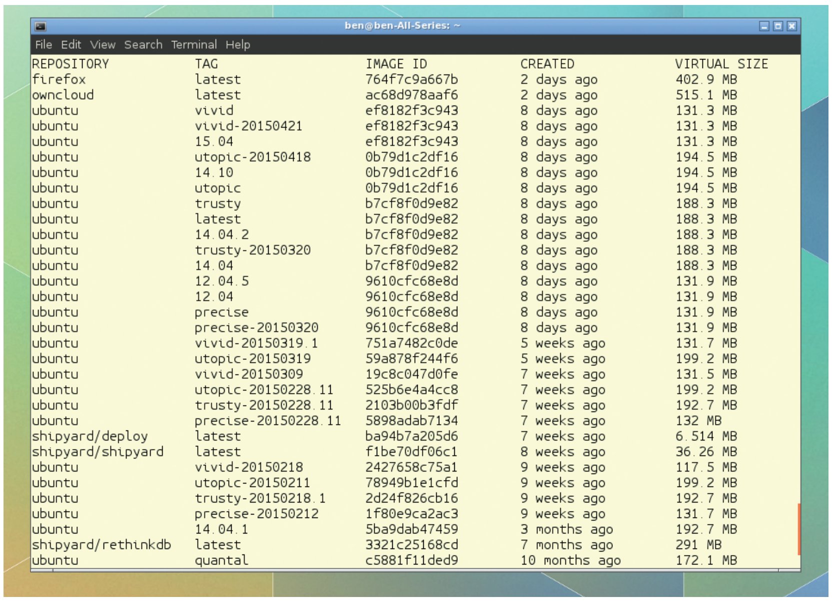
This will download and run the OwnCloud Docker instance, and expose the webserver on port 7000 (you can change this to another port if you wish). If you point your web browser to **localhost:7000/owncloud**, you should see a form inviting you to create an admin password, and then you can complete the owncloud install.

You can upload your containers to the Docker Hub. First you need to create an account at **https://registry.hub.docker.com**, then you can log in your machine with the command:

```
docker login <username>
```

Once you're logged in, you can push images to the Docker Hub with:

```
docker push <username>/<imagename>
```



```
REPOSITORY           TAG                   IMAGE ID            CREATED             VIRTUAL SIZE
firefox              latest                764f7c9a667b        2 days ago          402.9 MB
owncloud             latest                ac68d978aaf6        2 days ago          515.1 MB
ubuntu               vivid                 ef8182f3c943        8 days ago          131.3 MB
ubuntu               vivid-20150421        ef8182f3c943        8 days ago          131.3 MB
ubuntu               15.04                 ef8182f3c943        8 days ago          131.3 MB
ubuntu               utopic-20150418       0b79d1c2df16        8 days ago          194.5 MB
ubuntu               14.10                 0b79d1c2df16        8 days ago          194.5 MB
ubuntu               utopic                0b79d1c2df16        8 days ago          194.5 MB
ubuntu               trusty                b7cf8f0d9e82        8 days ago          188.3 MB
ubuntu               latest                b7cf8f0d9e82        8 days ago          188.3 MB
ubuntu               14.04.2               b7cf8f0d9e82        8 days ago          188.3 MB
ubuntu               trusty-20150320       b7cf8f0d9e82        8 days ago          188.3 MB
ubuntu               14.04                 b7cf8f0d9e82        8 days ago          188.3 MB
ubuntu               12.04.5               9610cfc68e8d        8 days ago          131.9 MB
ubuntu               12.04                 9610cfc68e8d        8 days ago          131.9 MB
ubuntu               precise               9610cfc68e8d        8 days ago          131.9 MB
ubuntu               precise-20150320      9610cfc68e8d        8 days ago          131.9 MB
ubuntu               vivid-20150319.1      751a7482c0de        5 weeks ago         131.7 MB
ubuntu               utopic-20150319       59a878f244f6        5 weeks ago         199.2 MB
ubuntu               vivid-20150309        19c8c047d0fe        7 weeks ago         131.5 MB
ubuntu               utopic-20150228.11    525b6e4a4cc8        7 weeks ago         199.2 MB
ubuntu               trusty-20150228.11    2103b00b3fdf        7 weeks ago         192.7 MB
ubuntu               precise-20150228.11   5898adab7134        7 weeks ago         132 MB
shipyard/deploy      latest                ba94b7a205d6        7 weeks ago         6.514 MB
shipyard/shipyard    latest                f1be70df06c1        8 weeks ago         36.26 MB
ubuntu               vivid-20150218        2427658c75a1        9 weeks ago         117.5 MB
ubuntu               utopic-20150211       78949b1e1cfd        9 weeks ago         199.2 MB
ubuntu               trusty-20150218.1     2d24f826cb16        9 weeks ago         192.7 MB
ubuntu               precise-20150212      1f80e9ca2ac3        9 weeks ago         131.7 MB
ubuntu               14.04.1               5ba9dab47459        3 months ago        192.7 MB
shipyard/rethinkdb   latest                3321c25168cd        7 months ago        291 MB
ubuntu               quantal               c5881f11ded9        10 months ago       172.1 MB
```

The **docker images** command will list all the images you currently have installed on your computer.

name your Docker image will have; we'll call ours **testubuntu2**. Inside this directory, you'll need to create a text file called **Dockerfile**, which will contain all the build instructions. In this case it should contain the following:

```
FROM ubuntu
MAINTAINER Ben Everard <ben@linuxvoice.com>

RUN mkdir ~/test
```

The first line tells Docker what to use as our base image; the second tells anyone working with this who to contact if there's a problem; and the final line tells Docker to run the **mkdir** command in the image before finishing. You can now create the image using the **docker build** command. The format is:

```
docker built -t <image-name> <image-directory>
```

We've used the directory **~/testubuntu2** and we want to call the image **testbuntu2**, so the command is:

```
docker build -t testubuntu2 ~/testubuntu2
```

You can now run the image **testubuntu2** exactly as you did **testubuntu**.

Alternative containers

It should come as no surprise to anyone paying attention to the init wars that the ever-expanding Systemd project includes a container controller. The **nspawn** command enables Systemd to automatically launch and control containers in a very similar way to Docker. What **nspawn** lacks, however, is an elegant system for making and sharing containers, though it is possible to use Docker containers.

CoreOS, a project that's building a minimal Linux distro just for cloud servers, is also working on a container control tool called **Rocket** (aka **rkt**). This works in a similar way to Docker, but it's much more stripped down, and without many of the more complex features of Docker that some people feel are leading to bloat on the original platform.

Rocket is still under heavy development, so not ready for production use just yet. At a lower level, there's also Linux Containers (LXC). This technology enables you to create and manipulate containers. With LXC, you have a lot of control, but it's not as easy to get started as with the other options. There's a new tool from the LXC project called LXD, which aims to provide easier access to LXC containers in a more Docker-like way.

At the time of writing, Docker has far more traction than its competitors both in terms of the number of people using the software, and the number of images available. However, containers are a young technology and few places are entrenched with Docker, so there's still time for one of the other technologies to become the Next Big Thing in containers.

What we've done so far more or less the 'Hello world' of Docker. Hopefully you should have a bit of an idea how it works, but this particular example is a bit useless.

Serving the world

In order to make our container more useful, there are a couple more arguments to **docker run** that we'll need.

- **-d (daemonise)** This runs the Docker container in the background and doesn't put you to a prompt like the first example. This is useful for running servers where you don't want to interact with them, you just want them to start and continue running.
- **-p <external-port>:<container-port>** This is used to connect the Docker container to the real world. It links a particular port on the Docker container to a port on the host.

We'll use these to containerise a website. First, we'll need a new directory to build our image in. We'll call ours **helloweb**. Then, in that, we'll need a Dockerfile to build our website. We'll start with Ubuntu, then we'll install *Apache*, then we'll include a simple web page. The Dockerfile for this is:

```
FROM ubuntu
MAINTAINER Ben Everard <ben@linuxvoice.com>

RUN apt-get update
RUN apt-get upgrade -y
RUN apt-get install apache2 -y

ADD website/* /var/www/html/
ADD apache-config.conf /etc/apache2/sites-enabled/000-
default.conf

EXPOSE 80
CMD /usr/sbin/apache2ctl -D FOREGROUND
```

The first section of this looks roughly like the first Dockerfile we made. It uses **RUN** commands to execute instructions inside the container, but in this case, it uses them to grab the latest packages, and install *Apache*. The **-y** flag on **apt-get** makes it run automatically without asking for input.

The rest of the Dockerfile is a little different. **ADD** commands are used to copy files from the build directory to the Docker container (we'll look at these a little later). **EXPOSE 80** tells Docker to open port 80 (the HTTP port) on the container, and **CMD** is the default command to run when Docker starts. In this case, it just runs *Apache* in the foreground.

As you've seen, this needs extra files for the website and for the *Apache* configuration. The website is in a directory called **website** inside the Docker build directory. The Dockerfile will contain whatever you put in this directory, so it could be as complex as you like, but we've just included a single file called **index.html**, which contains:

```
<html>
<head>
<title>HelloWeb!</title>
</head>
```

Security Keeping containers safe

You may be tempted to think that by running software in different Docker containers, each piece is isolated from the others and therefore a compromise in one Docker container won't allow an attacker to access another. In an ideal world this would be true, but the underlying technologies behind Docker (namespaces, cgroups and others) have not yet been adequately tested from a security point of view to make this claim. There's a high probability that a talented hacker could break out of the Docker container and compromise the rest of the system.

In other words, don't run anything in Docker that you wouldn't run on your system normally. Virtualisation provides a much higher degree of security and has been tested far more thoroughly than containerisation, so this is a better option if you want to run code you don't trust.

```
<body>
<h1>Hello Web!</h1>
</body>
</html>
```

The *Apache* config file sits in the Docker build directory (but not in the website directory) and contains:

```
<VirtualHost *:80>
ServerAdmin ben@linuxvoice.com
DocumentRoot /var/www/html

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
 </VirtualHost>
```

With all those files in place, you can build the Docker container from a terminal with:

```
docker build -t helloweb ~/helloweb
```

Then run it with:

```
docker run -d -p 7777:80 helloweb
```

You can then point your web browser to **localhost:7777** and view your website. Notice that the external port is 7777 despite the fact that *Apache* is listening on port 80 in the container, because when we ran the container, we used the argument **-p 7777:80**, which maps the external port 7777 to the container port 80. This means that you can run the Docker container unmodified on machines with other services running on port 80 and not have conflicts.

pwn the cloud

The previous example built a trivial website, but we can use it to deploy a more complex web app. In this case, it may be better not to copy in files as we did before, but to grab the latest version from an online repository. For example, to install OwnCloud, you could use the following Dockerfile:

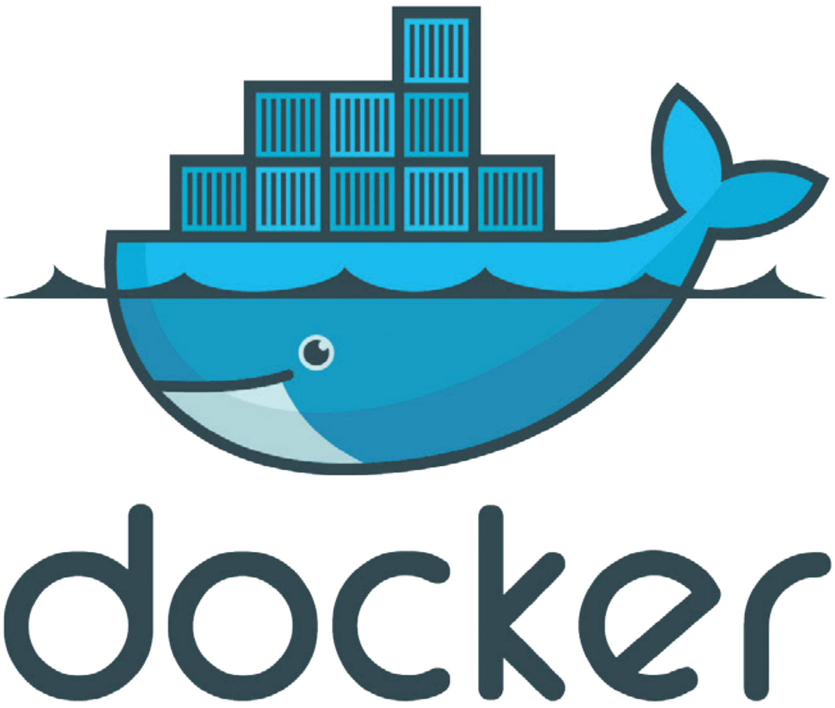
```
FROM ubuntu
MAINTAINER Ben Everard <ben@linuxvoice.com>

RUN apt-get update
RUN apt-get upgrade -y
RUN apt-get install -y  apache2 php5 php5-gd php-xml-parser
php5-intl php5-sqlite smbclient curl libcurl3  php5-curl bzip2
wget  sharutils
RUN wget -q -O - https://download.owncloud.org/community/
owncloud-latest.tar.bz2 | tar jx -C /var/www/html
RUN chown -R www-data:www-data /var/www/html/owncloud
ADD apache-config.conf /etc/apache2/sites-enabled/000-
default.conf
EXPOSE 80
CMD /usr/sbin/apache2ctl -D FOREGROUND
```

Here, we use **wget** to grab the latest version from the OwnCloud website. This is the best option for OwnCloud, but other projects deploy in different ways. For example, you could replace the **wget** line with a **git clone** command to grab the latest version of the some software that's kept on GitHub.

Notice that this also requires the **apache-config.conf** file that we used in the previous Dockerfile, so you'll need to copy this into the same directory for it to run. If you build and start this in the same way as before, but using the external port 7778, you'll find OwnCloud at **http://localhost:7778/owncloud**.

That's all you need to know to use Docker. Its main selling point is that it makes containers easy to use. With just a few lines of easily understood Dockerfiles, you can wrap your software up to make it easy to deploy across any Linux-based platform. It also makes it easier than ever before to install the latest versions of web apps (see boxout on the Docker Hub), so you can test out software without having to first install and configure all the requirements. LV

Ben Everard is the best-selling co-author of the best-selling *Learning Python With Raspberry Pi*, published by Wiley.



According to Docker, the logo communicates expedition, automation, encapsulation and simplification. To us it communicates a whale that's more buoyant than it should be.

"You can wrap your software up to make it easy to deploy across any Linux platform."

LINUX VOICE TUTORIAL

INTERACTIVE PYTHON COMPUTING FOR EVERYBODY

VALENTINE SINITSYN

From trying out code snippets to solving differential equations: IPython is a tool that has something to offer for any problem.

WHY DO THIS?
- Improve your Python shell experience
- Build a computer algebra system completely free
- Create interactive tutorials with live code samples

One thing that's great about Python is its immediacy: you think of something, you type the commands and see if it works or not. There's no compilation, no deployment, no whatever, and it's all great for quick prototyping.

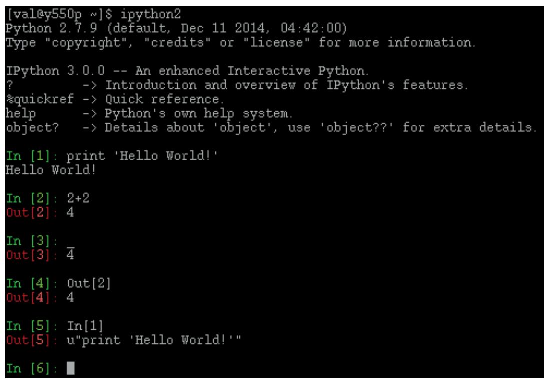
However, for ideas larger than a one-liner, things aren't so good. The Python shell is not very good with multiline history: if you spot a bug in a class definition, you can't fix it easily. There are no writing aids, like code completion. If your idea turns out to be good, you can't easily export it to a script, and there's no straightforward way to share you findings with others.

By itself, *IPython* is just a sophisticated Python shell, but that's only a part of the story. Being free software, it enjoys a vast library ecosystem. With these third-party add-ons, you can turn *IPython* into your testbed for programming ideas, an interactive tutorial or even a computer algebra system (CAS).

Getting ready

As an *IPython* user, you usually work in a console or in a web-based notebook. *IPython* comes with two consoles: traditional and a graphical *Qt*-based one. There are several processes involved in a typical *IPython* session. The most important one is the kernel (not to be confused with an OS kernel) which actually executes your code. For the notebook, there will be also a *Tornado*-based web server. This is known as decoupled process architecture: client and computational core are separated. You can even run them on different machines.

IPython components are glued together with *ZeroMQ*, which is a popular "sockets on steroids" library for high-level network communications. We're not going to cover it here, but you should be aware that *ZeroMQ* lacks built-in security mechanisms. So, if you're really going to run *IPython* in distributed fashion, be sure to read the relevant parts of the official documentation. In a nutshell, securing *IPython* kernel connections boils down to creating SSH tunnels and forcing *IPython* to listen on loopback addresses.



```
[val@y550p ~]$ ipython2
Python 2.7.9 (default, Dec 11 2014, 04:42:00)
Type "copyright", "credits" or "license" for more information.

IPython 3.0.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: print 'Hello World!'
Hello World!

In [2]: 2+2
Out[2]: 4

In [3]: _
Out[3]: 4

In [4]: Out[2]
Out[4]: 4

In [5]: In[1]
Out[5]: u"print 'Hello World!'"

In [6]: █
```

The *IPython* console sports bright colours and many interactivity features.

First steps

With everything in place, your first step into the *IPython* world is straightforward:

```
val@y550p:~$ ipython
Python 2.7.9 (default, Dec 11 2014, 04:42:00)
Type "copyright", "credits" or "license" for more information.

IPython 3.0.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
In [1]:
```

You'll see a banner revealing some of *IPython*'s powers. **%** is the so-called "line magic", because it works on a single line, and can be used inside blocks of code. There are also "cell magics" that start with **%%** and work multiline. Below is a command prompt which behaves as you might expect:

```
In [1]: print 'Hello world!'
Hello world!
```

Things will be different however, if your expression produces a value:

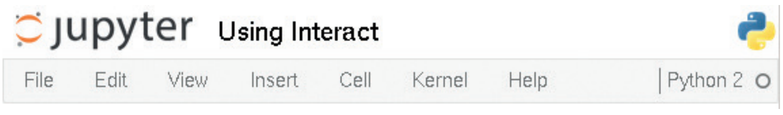
```
In [2]: 2+2
Out[2]: 4
```

Project Jupyter

Some pillars *IPython* stands on are Python-specific, such as *SymPy* or *Matplotlib*. The others, like decoupled protocol or web notebooks, are language-agnostic, so it's natural to extend *IPython*'s beyond Python.

This is exactly what project *Jupyter* does. It takes the *IPython* environment, notebook document format and various other bits and factors them out. *IPython* remains the default Python kernel, but you can switch to Ruby or anything else if you wish.

If you are using *IPython 3.0*, you've probably spotted the *Jupyter* logo in the web notebooks already. Plans are to finish the split by *IPython 4.0*, so version 3.x will likely be the last "monolithic" one.

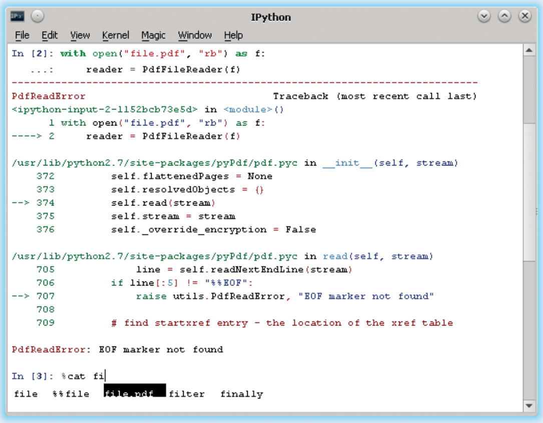


"Project Jupyter" sounds cooler than "IPython without Python".

Graphics covered

IPython sports a thing called "GUI integration". In plain words, it means that you can create *Qt*, *GTK* and some other graphical UI applications straight from the console. It's a useful feature for introspection, especially given the fact that *IPython* can be embedded in your own programs. It also comes handy when you need to quickly create a widget and test its behavior.

To use it, nothing is required on your side. Just execute **%gui qt** (or other toolkit). Don't start the main loop (**QApplication(sys.argv)._exec()** or similar) as it will block the console. Create your widgets and do other things as if it were already running, and *IPython* will take care of the rest.



```
In [2]: with open("file.pdf", "rb") as f:
   ...:     reader = PdfFileReader(f)
---------------------------------------------------------------------------
PdfReadError                              Traceback (most recent call last)
<ipython-input-2-1152bcb73e5d> in <module>()
      1 with open("file.pdf", "rb") as f:
----> 2     reader = PdfFileReader(f)

/usr/lib/python2.7/site-packages/pyPdf/pdf.pyc in __init__(self, stream)
    372         self.flattenedPages = None
    373         self.resolvedObjects = {}
--> 374         self.read(stream)
    375         self.stream = stream
    376         self._override_encryption = False

/usr/lib/python2.7/site-packages/pyPdf/pdf.pyc in read(self, stream)
    705             line = self.readNextEndLine(stream)
    706         if line[:5] != "%%EOF":
--> 707             raise utils.PdfReadError, "EOF marker not found"
    708
    709         # find startxref entry - the location of the xref table

PdfReadError: EOF marker not found

In [3]: %cat fi
 file   %%file   file.pdf   filter   finally
```

PdfFileReader complains about a malformed file, but Tab completion makes fixing it easy.

Here, output appears in its own cell. To suppress this, end the expression with a semicolon (**;**). The last output is always available via the special variable, **_**.

```
In [3]: _
Out[3]: 4
```

object? prints details about an object. *IPython* gathers these from docstrings, introspection and other means, and **??** tries to be even more verbose (though it doesn't necessarily succeed). This feature isn't much help on standard library objects, but is quite wordy with *IPython*'s own ones and various *IPython*-aware third-party libraries.

A double underscore **__** refers to the output before the last one. To get an arbitrary output, use **Out[N]** or **_N**, where **N** is the output's number (2 below):

```
In [4]: Out[2]
Out[4]: 4
```

Note that you can't refer to printed text this way, as **print** yields no value – it just puts characters on the screen. Similarly enough, you can use **_i**, **__i**, and **In[N]** (or **_iN**) to retrieve previous inputs:

```
In [5]: In[1]
Out[5]: u"print 'Hello world!"
```

A magic called **%history** prints all inputs in the current session. This comes in handy when you've tried some idea and decide it's worth putting into standalone script. The magic accepts various options (like the range of inputs to export) which you can examine with **%history?**. For more sophisticated session logging, consider the **%logstart** and **%logstop** magics. They also produce logs that are valid Python code you can readily use.

Writing aids

Session history is also easy to navigate the same way you do it in *Bash*. In fact, both shells relies on the *readline* library for all heavy lifting. Use the Up and Down keys (or Ctrl+P/Ctrl+N) to select previous inputs, or type some words and press Ctrl+R to search history. Unlike the standard Python interactive shell, this works multiline.

```
In [6]: def f(x):
   ...:     return 2 * x
   ...:
```

Note that *IPython* automatically indents code when you press Enter. The first empty line terminates the input. Now, press Ctrl+P: you'll get back to your function and can change it as you like. The only inconvenience is that it is really a long wrapped line, not a multiline text. You can navigate it with the Left and Right keys, Ctrl+A/Ctrl+E (or Home/End), but pressing Up or Down still switches to another history item. For a real multiline editor, do **%edit In[N]**. This opens input **N** in a default text editor (likely *Vim*). *Qt Console* and *Notebook* can edit multiline cells with no external aids.

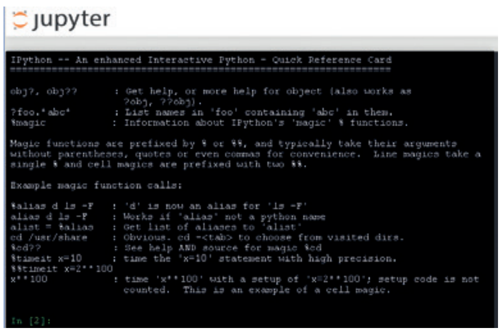
It is also possible to get one of your previous inputs editable at the command prompt. **%recall N** is the magic you need here. It comes in handy when the input you need is buried too deep in the history. If you only need to rerun the input, use **%rerun N**.

IPython also provides some writing aids, like Tab completion. If you forget a method's name, type some first letters then tap Tab to see the options. Even better, you can use shell wildcard syntax and **?** to search through Python objects:

```
In [7]: itertools.c*?
itertools.chain
itertools.combinations
```

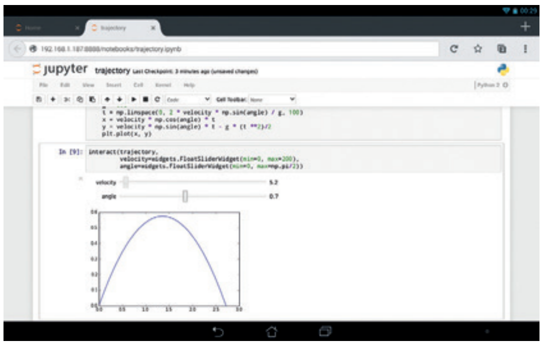
Need more terminals

IPython Notebook 3.0 comes with a built-in terminal, which is like *Chrome Secure Shell* but works in any modern browser. This requires the **python-terminado** package, and you can create terminals in Dashboard the same way as notebooks. It may come in handy for working remotely, but keep in mind that this opens shell access to anyone who connects to your *IPython Notebook* server. Be careful not to leave your system vulnerable.



```
IPython -- An enhanced Interactive Python - Quick Reference Card
================================================================

obj?, obj??      : Get help, or more help for object (also works as
                   ?obj, ??obj).
?foo.*abc*       : List names in 'foo' containing 'abc' in them.
%magic           : Information about IPython's 'magic' % functions.

Magic functions are prefixed by % or %%, and typically take their arguments
without parentheses, quotes or even commas for convenience.  Line magics take a
single % and cell magics are prefixed with two %%.

Example magic function calls:

%alias d ls -F   : 'd' is now an alias for 'ls -F'
alias d ls -F    : Works if 'alias' not a python name
alist = %alias   : Get list of aliases to 'alist'
cd /usr/share    : Obvious. cd -<tab> to choose from visited dirs.
%cd??            : See help AND source for magic %cd
%timeit x=10     : time the 'x=10' statement with high precision.
%%timeit x=2**100
x**100           : time 'x**100' with a setup of 'x=2**100'; setup code is not
                   counted.  This is an example of a cell magic.

In [2]:
```

IPython's web terminal is good enough to run the *ipython* console.



```
t = np.linspace(0, 2 * velocity * np.sin(angle) / g, 100)
x = velocity * np.cos(angle) * t
y = velocity * np.sin(angle) * t - g * (t **2)/2
plt.plot(x, y)

In [9]: interact(trajectory,
                 velocity=widgets.FloatSliderWidget(min=0, max=200),
                 angle=widgets.FloatSliderWidget(min=0, max=np.pi/2))
```

Interactive plots work even in mobile browsers, but you'll need a live kernel connection for them.

```
# skipped for brevity
```

IPython ignores prompts like **>>>**, **In []** or **...** in code you enter. It's very useful for pasting doctests and code snippets you find online.

So far we've discussed the *IPython* console, but all tricks you've learned work in *Qt Console* (**ipython qtconsole**) as well. History in *IPython* is per kernel, so it is shared between consoles. For instance, run **ipython console** in the terminal, then use **%qtconsole** to start *Qt Console* connected to the same kernel. You should be able to see your previous commands. Instead of **%qtconsole**, you could also use the **%connect_info** magic, which displays the kernel connection information and a command line you can run to connect to the existing kernel.

To exit the *IPython* console, press Ctrl+D.

Magic in a shell

IPython provides basic shell command equivalents. You can do things like **cur_dir = %pwd**, **%cd another_dir** and **%cd -** to return back, if you need. Shell scripting language is certainly powerful, but not necessarily intuitive. If you mix and match it with Python, you get best of both worlds.

Think of the following problem. You want to find all PDF reports in your Downloads folder that are less than five pages long, and copy them somewhere. To make it a bit simpler, let our reports have the **report_** prefix in their names.

Open the *IPython* console (*Qt Console* will do as well) and let's begin with:

```
In [1]: reports = !ls ~/Downloads/report_*.pdf
```

! denotes a shell command expression. It passes the command line to the shell, captures standard output and returns it as the result. The actual type of 'reports' is **SList**, as you can easily check with **?**:

```
In [2]: reports?
Type:     SList
...
File:     /usr/lib/python2.7/dist-packages/IPython/utils/text.py
Docstring:
List derivative with a special access attributes.
...
```

SList is IPython's own type that derives from **list**. You can use it as a normal Python list (**reports.l**), a newline-separated string (**reports.n**), a space-separated string (**reports.s**), or a list of Python's path objects (**reports.p**). For our purposes, we'll prefer the latter form. There is also a convenient **SList.grep()** method, which behaves the same way as the shell command.

Next, we'll loop through **reports**, getting the number of pages in each document. It's not straightforward to do in a shell, but the *pyPDF* library makes it a snap. You can probably find it in your distribution's repositories, and the code we need looks like this:

```
In [3]: from pyPdf import PdfFileReader
In [4]: from os import path
In [5]: for p in reports.p:
   ...:     with open(p, "rb") as f:
   ...:         print PdfFileReader(f).getNumPages()
```

Check that it works (it should). Finally, return to the fourth import and change the last line to:

```
In[6]: # original code here
  ...:     if PdfFileReader(f).getNumPages() < 5:
  ...:         !cp "$p" ~/somewhere/"{path.basename(p)}"
```

$p is substituted with the string value of the Python variable **p** before the code reaches the shell. You can also embed Python expressions into shell commands with curly braces. Quotes account for shell-escape characters, such as spaces.

The code we wrote isn't error-proof. If you happen to have a **.pdf** file that **PdfFileReader** can't parse (maybe an incomplete download), it will raise an error. When this happens, note how exceptions are rendered in *IPython*. They're usually quite detailed, but if you want more, try **%xmode verbose**. Exception tracebacks are also coloured so you can grasp them more easily.

LV PRO TIP

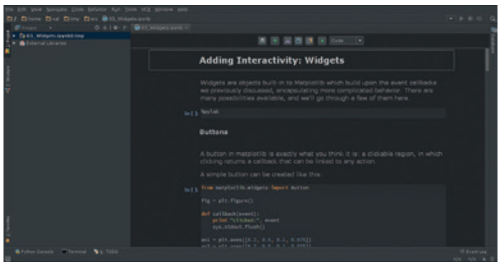
Tab completion in *IPython* also works for files. Press Tab and you'll see suggestions, like in a regular shell.

Notebooks beyond IPython

The more you get used to *IPython*, the more likely is you'll want to share a notebook with someone who doesn't have (and doesn't want) *IPython* installed. You can always use **ipython nbconvert**, but in the cloud era, there's a better way.

NBViewer (**http://nbviewer.ipython.org**) is the closest alternative to an online **nbconvert**. If you have your *IPython* notebook available online (maybe shared in Dropbox) or on GitHub/public Gist, simply paste its URL or Gist ID into the textbox and you're done. You can now send your contact a link that's viewable in any browser. The major limitation is that you must have your *IPython* notebook public, and there's also no kernel connection available.

If you're a professional Python developer, chances are you use the *PyCharm* IDE. It's commercial, but there's a free (as in beer and as in speech) variant called *PyCharm Community Edition*. Both support *IPython* notebooks, so you can enjoy great interactivity with high-end IDE features like code suggestions. However, my experience is that you may have trouble getting *PyCharm*'s *IPython* support to work. *PyCharm* will also use *IPython* as a Python shell if available.



If want a whole block of pure *Bash* code in your *IPython* session, use the **%%bash** cell magic. Everything you enter in that input will go to *Bash* directly, and the output will be printed. Similar cell magics exist for Perl and Ruby.

Pretty plotting

IPython consoles are good for us geeks, but for many others around *IPython* is a synonym for web-based *Notebook*. It's a great tool for interactive tutorials, presentations, and online learning. It also provides scientific computing environment rivalling commercial tools like *Maple* or *Wolfram Mathematica*.

Start the service via **ipython notebook**, and you'll see the web browser window with Dashboard. Dashboard displays *IPython* documents in the current directory and enables you to create new ones. *IPython Notebook* documents (sometimes also called *IPython* notebooks) are just JSON files. You can convert them to other formats like HTML or *Latex* with **ipython nbconvert**. However, in this case you'll lose much of the interactivity.

Notebook itself is a collection of cells. Cells have a type: code cells contain Python code, heading cells give your document a structure, Markdown cells are used for formatted texts and so on. To change a cell's type, use the drop-down under the main menu bar. When you're in command mode (hello *Vim*!), you can move between cells, create, and cut and paste them. However, you can't cut and paste multiple cells, and there is no multi-level undo. Clicking on a cell switches you to edit mode, where you can change the cell's contents.

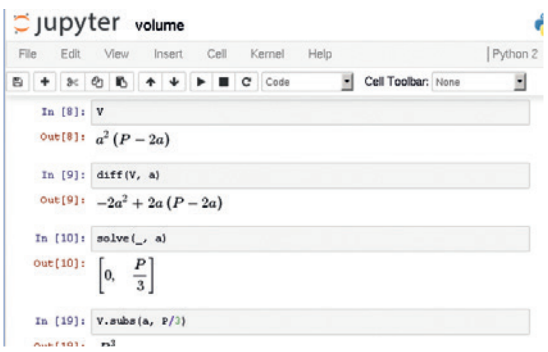
To execute a cell, press Shift+Enter or use the main menu. It will be evaluated, and the output (if any) will appear beneath. All history-related commands you know work in notebooks, too. Cell evaluation may take time, and a circle at the top-right corner indicates whether the kernel is currently busy or not. If it is, the circle will be filled. You can interrupt the current operation via Kernel > Interrupt or Ctrl+M I.

Let's write a simple example that draws the trajectory of a small body (like a stone) thrown at some angle above the horizon. For that, we'll need plotting facilities. *Matplotlib* is the *de facto* standard plotting tool in Python. It's modelled after *Matlab* (but provides a fully pythonic API as well) so drawing anything is very easy. First of all, we need to enable *Matplotlib* in the session. To do this, run:

```
In[1]: %matplotlib inline
```

"Inline" here means that graphs we plot will appear in the notebook and not in a separate window.

Two main factors affect trajectory shape: the initial velocity and the angle. The x and y coordinates are then easily found as time functions (consult the physics textbook of your liking). From a mathematical viewpoint, this means the trajectory y(x) is given as parametric function. *Matplotlib* can draw it as well as 2D/3D plots, histograms and more exotic plot types. All we need is to write a simple function:



```
In [8]: V
Out[8]: a^2 (P − 2a)
In [9]: diff(V, a)
Out[9]: −2a^2 + 2a (P − 2a)
In [10]: solve(_, a)
Out[10]: [0, P/3]
In [19]: V.subs(a, P/3)
```

Latex makes formulas look nice even on the web.

```
import numpy as np
import matplotlib.pyplot as plt
def trajectory(velocity, angle):
  g = 9.8
  t = np.linspace(0, 2 * velocity * np.sin(angle) / g, 100)
  x = velocity * np.cos(angle) * t
  y = velocity * np.sin(angle) * t - g * (t ** 2) / 2
  plt.plot(x, y)
```

We simply find out how long the body will go and calculate its x and y coordinates in 100 consecutive moments of time during the flight. Now, if you call **trajectory(5, np.pi/4)**, you'll see a curve. But how would a change of velocity or angle affect its shape? It's a good question, and *IPython* has the answer. You may be surprised, but making the plot interactive is mostly a matter of a one-liner:

```
from IPython.html import widgets
widgets.interact(trajectory,
  velocity=widgets.FloatSliderWidget(min=0, max=20),
  angle=widgets.FloatSliderWidget(min=0, max=np.pi/2))
```

interact takes a function and creates widgets to manipulate its arguments. It tries to guess appropriate widget types (eg uses checkboxes for Booleans), but you can always override the defaults as we do here. You can also apply it as a decorator:

```
@widgets.interact
def trajectory(velocity=1, angle=np.pi/4):
  # function body
```

Play with the plot: for instance, set **velocity** to some fixed value and drag the **angle** slider to see which value gives you the longest flying distance (measured along the x axis). You should get something around 0.78 (**np.pi/4**). Note however that this will work only if you are connected to the kernel, as the function is re-evaluated every time you change its arguments.

IPython encompasses many good technologies, and if you want to know more about *Matplotlib*, *SymPy* or whatever, please drop us a line. In the meantime, have a look at **https://github.com/ipython/ipython/wiki/A-gallery-of-interesting-IPython-Notebooks**. Here, *IPython* authors collect notebooks from various fields that you can use as a starting point for your own work, or just for learning. *IPython* welcomes contributions, and if you do something that's accepted upstream, please share it with us! LV

Dr Valentine Sinitsyn teaches Physics, develops high-loaded services and does other clever things with Python.

LV PRO TIP

Always run **ipython notebook** under *tmux* or *screen* (LV013). This way, you won't interrupt a session if you close the terminal occasionally.

LINUX VOICE TUTORIAL

CODE NINJA: BINARY BOOLEAN LOGIC

BEN EVERARD

Understand True and False, and program using the very essence of computer logic.

WHY DO THIS?
- Master the building blocks of computation
- Improve your programs by manipulating raw bits

If you've done even a little programming before, you'll probably have come across the basic logical operators **AND**, **OR**, **NOT** and **XOR** at some point. They operate on values that can be either True or False. These values are known as Boolean (after the mathematician George Boole). For example, take a look at the following Python code:

```
if first_name == "Ben" or first_name == "Mike":
        print "Cool name!"
```

This uses the variable **first_name** and first compares it to the string **"Ben"**. This comparison will either return True or False depending on whether or not **first_name** contains **"Ben"**. Then it will compare the variable to **"Mike"** and again it will return either True or False. The **OR** operation then looks at the two values returned, and if either of them is True, the whole statement returns True. If neither of them is True, then the whole expression returns False (actually, there's a little white lie in this – see the boxout for more details).

The operators **AND** and **XOR** (exclusive **OR**) work in a similar way. **AND** returns True if both inputs return True, and False otherwise. **XOR** returns True if either one of the inputs returns True, but False if either none or both of the inputs return True. The full range of outputs from all the logical operations is given in the truth tables below.

You can combine these to build up complex expressions. Python's interactive shell is a great place to experiment with these. For example:

```
>>> print False or True
True
>>> print (False and True) or (False or True)
True
>>> print (True ^ True) and (False ^ False)
False
```

In the last example, the   is used to denote **XOR**.

Logical operators

As well as these operators that take two inputs, there's the **NOT** operator, which acts on just a single input and changes it, so if it's True it returns False and if it's False it returns True:

```
>>> print not True
False
>>>print not False
True
```

There are three more logical operators that are created by combining the two-input operators with not. **NOR** is not and **OR**, in other words (**NOT** (a **OR** B)) is the same as (a **NOR** b).

So far, we've been looking at logical operators that work with single values, True or False. At a lower level, usually, a binary 1 represents True and 0 represents False. Since everything in your computer is also stored in binary, you can use logical operators on the bits that make up larger values.

As an example, the number 4, when represented in binary is 100, while the number 3 is 011. If you start with the left-most bit of both numbers, you get 1 (or True) from 4, and 0 (or False) from 3. True **OR** False is True, so the first bit of our output is 1. Then focus on the next bit along. From the 4, this is 0, while from the 3 you get 1. Again, these **OR** together to produce 1, and likewise with the last bit. The output of 4 **OR** 3 is 7 when done bitwise. In Python, this is:

Truth tables give the outputs of logical operators for every possible set of inputs.

OR

First input	Second input	Output
False	False	False
False	True	True
True	False	True
True	True	True

AND

First input	Second input	Output
False	False	False
False	True	False
True	False	False
True	True	True

XOR

First input	Second input	Output
False	False	False
False	True	True
True	False	True
True	True	False

NAND

First input	Second input	Output
False	False	True
False	True	True
True	False	True
True	True	False

NOR

First input	Second input	Output
False	False	True
False	True	False
True	False	False
True	True	False

XNOR

First input	Second input	Output
False	False	True
False	True	False
True	False	False
True	True	True

```
>>> print 4|3
7
```

Notice that we used the operator **|** (the pipe symbol) rather than **OR**. That's because **OR** works on whole values while **|** operates at bit-by-bit level. Bitwise **AND** is done with **&** and bitwise **NOT** is done with **~**. **XOR** is still done with **^**.

Binary Booleans

Using bitwise operators on whole numbers works, but it looks a bit strange. It's easier to see what's going on if you view everything in binary, and Python lets you do this. Firstly, anything starting with **0b** will be treated as binary. Secondly, the **bin(function)** can be used to convert a number to binary. So, to display a bitwise **OR** of 4 and 3, but display everything in binary, you can do:

```
>>> bin(0b100 | 0b011)
0b111
```

This performs exactly the same function as the previous bit of code; the only difference here is that we've told Python to display everything in binary.

The most common use of bitwise operations is applying masks to a value when you are only interested in a few of the bits. The two most common operators for this are **AND** and **XOR**. **AND** can be used when you only want to remove some of the values.

It's quite common to transmit Boolean values in 8-bit sequences that are usually interpreted as numbers. For example, if you're controlling hardware, you might have eight outputs that you can switch on or off. You can send a byte of data to tell the controller which ones should be on and which ones should be off. Each bit would correspond to a particular pin. In order to extract the value for a pin (eg pin 4), you could use the following Python code (where the variable **data_byte** contains the data for which pins to switch on)

```
>>> data_byte & 0b00010000
```

This can be incorporated into an **if** function. For example (where the function **set_pin()** is used to control the hardware):

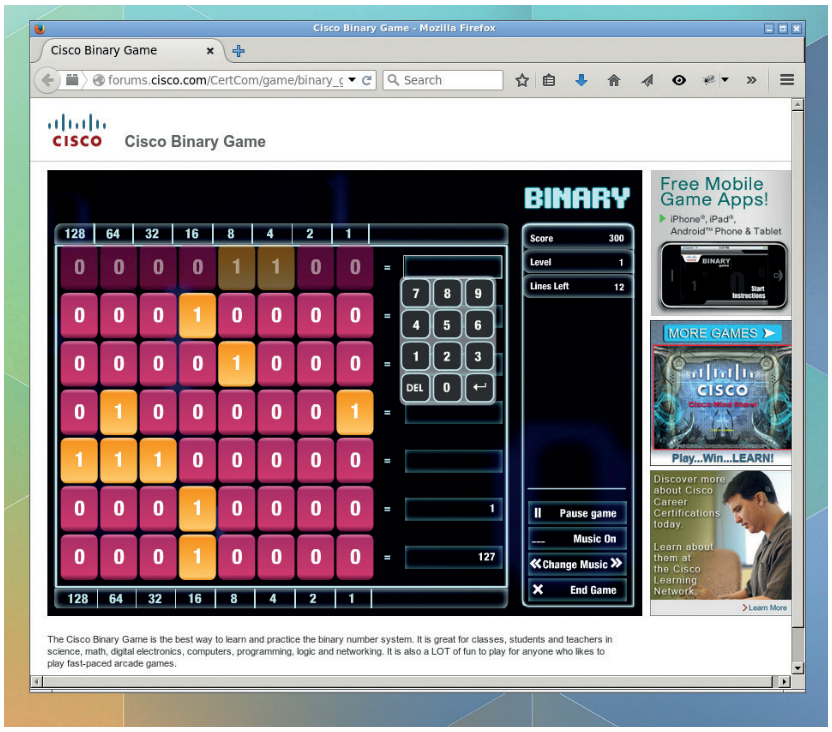
```
if data_byte & 0b00010000:
            set_pin(4, "ON")
```

The **if** statement works because Python considers anything other than 0 to be True.

You can use bitwise **OR** in a similar way because anything **OR**'d with True is always True, and anything **OR**'d with False is its original value. This way you can manipulate individual bits in the byte. For example, to set the fourth bit of **data_byte** to 1 but leave the rest unchanged:

```
>>> data_byte = data_byte | 0b00010000
```

Bitwise **XOR** has a couple of interesting properties. Firstly, anything **XOR**'d with 0 is itself, while anything **XOR**'d with 1 is its opposite. This can be used to selectively apply **NOT** operations to some bits in a byte. For example:


Cisco Binary Game - Mozilla Firefox
forums.cisco.com/CertCom/game/binary_g
Cisco Binary Game
BINARY
Score 300
Level 1
Lines Left 12
Pause game
Music On
Change Music
End Game
Free Mobile Game Apps!
MORE GAMES
Play...Win...LEARN!
Discover more about Cisco Career Certifications today.
Learn about them at the Cisco Learning Network.
Learn More

The Cisco Binary Game is the best way to learn and practice the binary number system. It is great for classes, students and teachers in science, math, digital electronics, computers, programming, logic and networking. It is also a LOT of fun to play for anyone who likes to play fast-paced arcade games.

If you struggle to understand binary numbers, the Cisco Binary Game is a great way to learn (**http://forums.cisco.com/CertCom/game/binary_game_page.htm**).

```
>>>bin(0b10101010 ^ 0b11110000)
01011010
```

Another feature of bitwise **XOR** is anything **XOR**'d with the same value twice is unchanged. This can be used to create a simple (and not particularly secure) form of encryption. If you pick a key that's the same length as the message (or repeat a shorter key), you can **XOR** the message and the key once to get an encrypted message. You can then bitwise **XOR** the encrypted message with the key again to get the original message. So:

```
message = 0b10101010
key = 0b11001100

encrypted_message = message ^ key
bin(encrypted_message ^ key)
```

Bitwise operations can cause some strange problems because of the way numbers are stored. For example:

```
>>> print ~1
-2
```

"It's common to transmit Boolean values in 8-bit sequences."

This is because Python stores numbers using a method known as two's complement, where the largest binary digit is negative and subtracted from the rest of the numbers. For example, in 4-bit two's complement, 1000 is -8, 1001 is -7 (-8 + 1) and 0111 is 7. In this case, **NOT 0001** is 1110, which is -8 and 6, or -2. Python stores numbers in a slightly unusual way. Instead of allocating a fixed number of bits to a variable (as C does for example), Python increases the number of bits depending on how large the number stored is. This makes two's complement a little unusual, so carefully test any bitwise functions that may operate on negative numbers.

LINUX VOICE TUTORIAL

ASMSCHOOL: EXPAND THE SELF-MADE OS

MIKE SAUNDERS

Part 5: We finish off this series by making system calls accessible to user programs, like in a full operating system.

WHY DO THIS?
- Learn what compilers do behind the scenes
- Understand the language of CPUs
- Fine-tune your code for better performance

Last issue we put together a simple operating system, using the techniques and tricks we explored over the last four issues. This miniature OS consists of a tiny bootloader that loads a kernel from a FAT12 formatted floppy disk, and the kernel then displays a prompt and takes input. If you enter the **ls** command you get a list of files from the disk – or if you enter a filename, the OS will load it and try to execute it.

That's all good and well, but it means our OS is just a glorified program launcher at this stage. Proper OSes provide facilities to programs, however, so that they don't have to implement everything themselves. For instance, every standalone program shouldn't need its own string printing routine – our OS can provide that, among other features. So in this final instalment of our assembly tutorial series, we'll beef up our kernel with routines that programs can call.

1 MAKING SYSTEM ROUTINES AVAILABLE

We'll be working from the codebase used last month, so grab it from **www.linuxvoice.com/code/lv015/asmschool.zip**. As a quick reminder, here's a list of the files included and what they do:

- **boot.asm** The 512-byte bootloader that's injected into the start of the FAT12 floppy disk image.
- **bpb.asm** The BIOS parameter block, a chunk of disk-identification information that is embedded into **boot.asm** during assembly.
- **disk.asm** FAT12 file-reading routines, as used by **boot.asm**.
- **mykernel.asm** Our OS kernel, including a simple command line and program execution facility.
- **lib.asm** A set of useful routines for screen, keyboard and disk handling, which gets included into **mykernel.asm** during assembly.
- **test.asm** A test program that our OS can load and launch – it just prints the letter 'X' and exits.

To assemble the bootloader, use:

```
nasm -f bin -o boot.bin boot.asm
```

To create a virtual floppy disk image called **floppy.img** in MS-DOS format, and inject our bootloader into the first 512 bytes, do this:

```
mkdosfs -C floppy.img 1440
dd conv=notrunc if=boot.bin of=floppy.img
```

To assemble the kernel and test program, and add them to the floppy disk image:

```
nasm -f bin -o mykernel.bin mykernel.asm
nasm -f bin -o test.bin test.asm
mcopy -i floppy.img mykernel.bin ::/
mcopy -i floppy.img test.bin ::/
```

(Note that **mcopy** is part of the *GNU Mtools* package, available in most distros.) Now you boot **floppy.img** in a PC emulator such as *QEMU* or *VirtualBox* – eg **qemu-system-i386 -fda floppy.img**. So, we have our codebase and build instructions. Let's build MyOS 2.0!

> "Every program shouldn't need its own string printing routine – our OS can provide that."

What's our vector, Victor?

Imagine that **test.asm** wants to use the **lib_print_string** routine inside our kernel. What does it need to do? Well, it has to use the call instruction followed by the address in memory of **lib_print_string**. But there's a major problem here: **test.asm** has no idea where **lib_print_string** is. If we disassemble our kernel using **ndisasm mykernel.bin**, and search for "**pushaw**" (the first instruction in the **lib_print_string** routine), we can see that the code for **lib_print_string** starts at position 557 (hexadecimal, in the column on the left).

So, that's a start – **test.asm** could just put a string location in SI and use **call 0x557** to print a string, right? That may indeed work, but here we bump into another problem: the location of **lib_print_string** will change as we make modifications to the kernel. It could move by just a few bytes if we make a minor alteration, or it could jump by several kilobytes if we add loads of code to **lib.asm** before it.

Fortunately, however, there's a way to fix this: using vectors. If we put a series of **jmp** instructions at the beginning of our kernel, which jump to different routines in **lib.asm**, these will always stay in the same place. These jumps are called vectors and redirect call instructions from programs to the appropriate places.

Let's see this in action. Edit **mykernel.asm** and add three **jmp** instructions and a "start" label to the very top so that it looks like this:

```
        jmp start               ; 0x0000
        jmp lib_print_string    ; 0x0002
        jmp lib_input_string    ; 0x0005

start:
        mov ax, 2000h
        mov ds, ax
        mov es, ax

loop:
        mov si, prompt
        call lib_print_string
        ...
```

The hexadecimal numbers in the comments alongside these **jmp** instructions show exactly where they'll be in RAM when the kernel is loaded. So, the **jmp start** instruction is at position 0 (as it's right at the start of the kernel), while **jmp lib_input_string** is at position 0x0002, and the next **jmp** is at 0x0005. (The first **jmp** instruction is only two bytes, as it's jumping to a very close address, just a few bytes ahead, whereas the other **jmp**s use three bytes and go to code further away, and therefore use longer addresses.) You can find out the location of these jump instructions using **ndisasm mykernel.bin** – the locations are in the left-hand column.

Make it easier for other developers

So, we've created three vectors here: if a program wants to use **lib_print_string**, instead of using that routine's exact address in memory, it can simply call the vector at position 0x0002, which then redirects to **lib_print_string**. This vector will always remain at 0x0002, unless we change the order of the vectors – but as Linus Torvalds always says with colourful language, you NEVER break userspace – so our vectors won't change location. We can add more of them, but that doesn't change the location of the previous ones.

Try this out by modifying **test.asm** like so:

```
        ORG 32768

        mov si, string
        call 0x0002
        ret

        string db "Ciao!", 0
```

Assemble the kernel and **test.asm** and then copy them to the disk image as per the previous instructions, and boot up the OS. Run **test.bin** and *voilà*: it uses the kernel's **lib_print_string** routine. It doesn't care where that routine is in RAM, but knows that it can call the vector at 0x0002, which will then jump to the right place.

And the same is true of **lib_input_string** – we've also added a vector for that. You can now go ahead and add other vectors for routines in **lib.asm**, making them available to standalone programs. Yes, our project really is becoming a proper operating system!

More real-hardware fun, this time showing the use of graphics routines we've added to our simple operating system.

But now imagine that you want to get other people to write software for your OS. If you tell them to call various hexadecimal numbers, it all looks a bit hacky, so we can make our pseudo-API more attractive by aliasing proper names to the numbers. Alongside **test.asm**, create a file called **myos.inc** with the following contents:

```
lib_print_string equ 0x0002
lib_input_string equ 0x0005
```

The **equ** here simply means: the string on the left is equivalent to the number on the right. So now you can modify **test.asm** as follows:

```
        %include "myos.inc"

        ORG 32768

        mov si, string
        call lib_print_string
        ret

        string db "Ciao! inc", 0
```

By including **myos.inc** into our **test.asm** file, we can use human-readable names for the routines and not just the hexadecimal numbers (although those will still work, of course). As you add more routines to **lib.asm** and make them accessible via vectors at the start of your kernel code, you can add more **equ** statements to **myos.inc** and make it easier for people to write programs for your OS.

2 FUN WITH GRAPHICS

We've worked hard over the last five issues, so it's time to play hard as well by adding some graphics capabilities to our OS. As with the keyboard, screen and disk routines, the BIOS can help out here – although it's not especially fast. If you had wanted to write lightning-fast games back in the glory days of MS-DOS, you'd have worked with video RAM directly, but in our case we can ask the BIOS to plot pixels for us and do the grunt work.

First up, we need to create three routines: one to enable graphics mode, one to plot a pixel with the specified colour and location on the screen, and one to return to text mode. Open **lib.asm** and add the following routines:

```
lib_gfx_mode:
        pusha
        mov ah, 0
        mov al, 13h
        int 10h
        popa
        ret

lib_text_mode:
        pusha
        mov ah, 0
        mov al, 3
        int 10h
        popa
        ret

lib_plot_pixel:
        pusha
        mov ah, 0Ch
        mov bh, 0
        int 10h
        popa
        ret
```

The first of these, **lib_gfx_mode**, uses the BIOS (**int 10h**, routine 0) to switch to 320x200 pixel mode with 256 colours at your disposal. Sure, it's very low-res by modern standards, but plenty of great games were written in this limited environment! The second routine switches back to text mode, and the third plots a pixel and takes three parameters: the colour in the AL register, the X (horizontal) location in the CX register, and the Y (vertical) location in DX.

Next up, add vectors to these at the start of your kernel. If you haven't added any other vectors for routines in **lib.asm** yet and you only have the three we created earlier, you can add them onto the end like so:

```
        jmp lib_gfx_mode      ; 0x0008
        jmp lib_text_mode     ; 0x000B
        jmp lib_plot_pixel    ; 0x000E
```

(Again, the vectors are three bytes apart; use **ndisasm** to see where all the **jmp**s are in **mykernel.bin** if you're not sure.) Now open **myos.inc** and add the appropriate **equ** lines for these routines, so that other programs can use them, and then edit **test.asm**. Here

What's next?

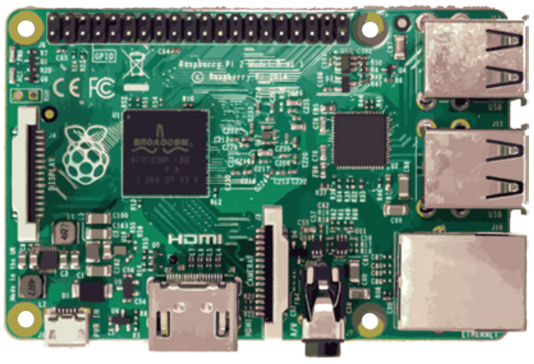
So, where do you go from here? The next step is to explore some other x86 real-mode 16-bit operating systems, such as MikeOS (**http://mikeos.sf.net**). No, that isn't just a cheap plug – MikeOS is very similar to the mini OS we've made here, albeit with many more library routines and kernel features. But the memory layout and use of vectors is the same, the code is well commented, and there's plenty you can nab for your own OS (it's under a BSD-like licence).

After that, you'll want to explore something more powerful. KolibriOS (**www.kolibrios.org**) is a mightily impressive graphical desktop operating system written in assembly language, which features pre-emptive multi-tasking, various filesystem drivers, and other cool features. KolibriOS is a fork of MenuetOS (**www.menuetos.net**), which is open source in its 32-bit incarnation, and includes extra sound and network drivers. It's insanely fast, but still rather limited compared to the likes of Linux and the *BSDs.

Of course, x86 isn't the be-all and end-all of assembly language. In fact, many coders regard it as a very ugly language, due to all the cruft that has been added onto it since the 1980s (the upside of this is excellent backward compatibility). ARM chips are everywhere these days, and while most of them are in embedded or locked-down devices such as phones and tablets, if you have a Raspberry Pi you can start hacking ARM assembly code straight away.

There's a good starting tutorial at **http://tinyurl.com/nsgzq89**, which explains how to assemble and link binaries, and then moves on to the specifics of the ARM chip in the subsequent chapters. Note that while ARM has different instructions and registers from x86, most of the concepts you learn are the same in any assembly language. (Well, unless you start coding for extremely limited embedded chips that don't even have multiply instructions!)

Once you're up to speed on ARM, you could try making your own OS for the Raspberry Pi. This is considerably more involved than our tutorial series, as you have no BIOS to help you out – indeed, you have to write your own video driver early on just to display anything on the screen! But it's good fun, so check out this guide: **www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os**.



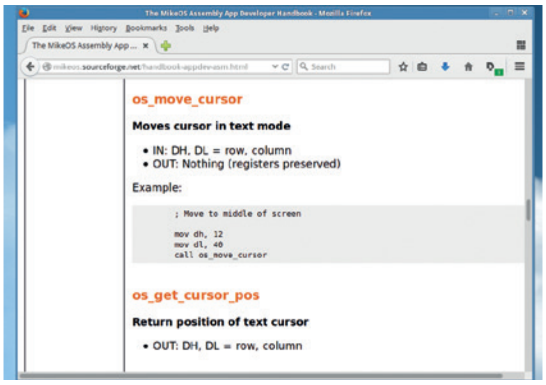
The Raspberry Pi rocks for many reasons, but it's also the ideal platform for learning ARM assembly.

we're going to write a program that draws horizontal lines of pictures on the screen, from top to bottom, changing colour each time. This program makes use of an inner and outer loop, so it's a good way to beef up your assembly knowledge with concepts you've probably come across in higher-level languages. Here's the code:

```
        %include "myos.inc"

        ORG 32768

        call lib_gfx_mode

        mov al, 0
        mov cx, 0
        mov dx, 0

outerloop:
        inc al
        inc dx
        cmp dx, 199
        je $

        innerloop:
                call lib_plot_pixel
                inc cx
                cmp cx, 319
                jne innerloop

        mov cx, 0
        jmp outerloop
```

Here, after switching to graphics mode, we set our colour, X position and Y position registers to zero. Then we begin our outer loop, which bumps up the colour by one and moves on to the next line. If we've reached line 199 (the bottom of the screen), then we execute **jmp $**, which jumps to the same instruction in the code – thereby setting up an infinite loop. (Yes, the only way out of this program is to reset the machine!)

If we're not at the bottom-most line, however, we begin an inner loop. This plots a pixel at the current location and then adds 1 to the CX, moving to the right each time. When we hit the right-hand side (319) the loop ends, so we reset the X position back to zero and return to the outer loop for the next line. Build the OS and boot it, and you'll see results like in the picture – not bad for such a small program, is it?


The MikeOS Assembly App Developer Handbook - Mozilla Firefox
File Edit View History Bookmarks Tools Help
The MikeOS Assembly App...
mikeos.sourceforge.net/handbook-appdev-asm.html
Search

os_move_cursor

Moves cursor in text mode

- IN: DH, DL = row, column
- OUT: Nothing (registers preserved)

Example:

```
        ; Move to middle of screen

        mov dh, 12
        mov dl, 40
        call os_move_cursor
```

os_get_cursor_pos

Return position of text cursor

- OUT: DH, DL = row, column

It's a good idea to document your system calls so that you (and others) know exactly which registers they take and return.

Steve Ballmer was right – it's all about developers, developers, developers, developers! Win them over with an API for your operating system.

Custom lines and shapes

With this knowledge, you can now build some custom routines to draw straight horizontal and vertical lines, and even filled or outlined boxes. But what about arbitrary lines from one point to another? This is a bit more complicated, but one solution is to use Bresenham's line algorithm. This determines the pixels that need to be plotted on the screen when a straight line is drawn from one point to another. The maths involved in this is beyond the scope of this tutorial, but there's some pseudocode at **http://tinyurl.com/k6qsxty**, which will help you to implement it.

The web is full of x86 assembly language implementations of the algorithm, such as **http://bloerp.de/code/asm/x86/line.html**, so do a bit of Googling and see how it's done. You can find similar code chunks explaining how to draw circles and other shapes. If you're feeling especially ambitious, and have plenty of time, you could even try adding mouse support. The BIOS doesn't provide any routines for this, but you can write a simple PS/2 mouse driver and many PCs will emulate USB mice as PS/2 ones (in 16-bit real mode). See **http://wiki.osdev.org/Mouse_Input** to get started – and note the links at the end of the page, which include example code listings.

And that's it! We've come to the end of our journey through the awesome land of assembly language, and hopefully it has whetted your appetite to explore even further. Keep banging those bits and bytes on bare metal, and if you ever happen to land a six-figure salary from an assembly language-related job, this author wouldn't object to a free beer in the post... happy hacking! LV

Mike Saunders is now moving on to coding mostly in 1s and 0s, with the odd 2 thrown in for good measure.

LINUX VOICE MASTERCLASS

Manipulate sound, filter the noise and rub some funk on your masterpiece.

CAPTURE AND TWEAK AUDIO WITH AUDACITY

Your one-stop tool for melodiously enhancing recordings and audio files.

MAYANK SHARMA

Audacity is one of those rare breed of open source software that Linux users love to flaunt. It runs across platforms and packs in more tricks than Gandalf. You can use the incredibly powerful and versatile application for everything from recording music, vocals, instruments and other sounds and editing them into a professional sounding-mix. One of the most popular uses of *Audacity* is to churn out podcasts, though you can also use it to compose tracks for your garage band, record voiceovers, capture audio from apps, make ringtones and do a lot more.

All this robustness hides behind an interface that might appear cryptic to new users, but it's also fairly intuitive and grows on you when you start using it. Grab *Audacity* from your distro's official repository, either with

```
sudo apt-get install audacity
```

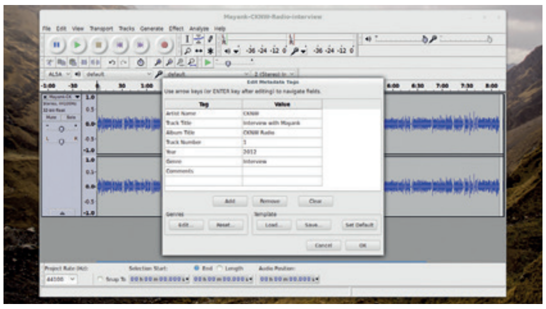
if you use a Debian-based distro or

```
sudo yum install audacity
```

on Fedora-based distros. While *Audacity* is equipped to export Ogg files, if you wish to save your recordings in other formats such as MP4 you'll need to make sure your distro includes the *FFmpeg* libraries. Similarly, for saving as MP3 you'll need the *Lame* encoder. You can install them both from your distro's package manager.

The first thing you should do when you launch *Audacity* is to check your recording hardware. Bring up the pop-up menu next to the microphone icon in the top bar and select the Start Monitoring option. Now speak into the microphone and if the input meters respond, your recording hardware is properly hooked up. Note however that the meters shouldn't hit 0dB. If they do, lower the record volume using the Input Volume slider on the right, or using your distro's Sound Settings dialog.

Before you capture any audio, make sure the audio sampling rate is set higher than 22050Hz. By default the app sets this to 44100Hz (in the bottom-left corner of the interface) and you should leave it as is. Now click the red button and start talking. Use the Pause button to take a break and resume recording from where you left off. When you've finished, click the Stop button. If you press the Record button again, Audacity will record the audio in a new track. Instead, to continue recording from the end of the original track press and hold the Shift key before clicking the Record button again.



Edit Metadata Tags

Use arrow keys (or ENTER key after editing) to navigate fields.

Tag	Value
Artist Name	CKNW
Track Title	Interview with Mayank
Album Title	CKNW Radio
Track Number	1
Year	2012
Genre	Interview
Comments	

While exporting a recording you can also attach various optional metadata with the file.

LV PRO TIP

You can tell if a track is active by the colour of the track menu – an active track's menu is a darker shade than an inactive track's menu.

Enhance with effects

One of the coolest features of *Audacity* are the dozens of effects housed under the Effect menu. Besides the Noise Removal effect that we've demonstrated in these pages, there are several other useful ones that you can use to enhance your recordings. The Change Pitch effect lets you alter the pitch of your voice without changing the tempo. Use the Echo effect to dramatise announcements, or the Reverse effect to play a track back-to-front for fun. There's also the Compressor effect, which equalises volume variations in the track by effectively making the quiet parts louder.

Polish your recordings

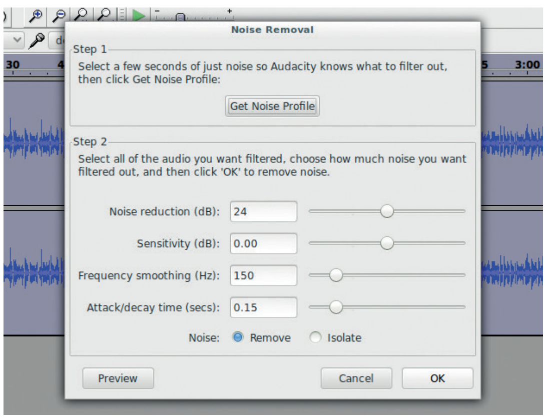
If your cat pounced on you in the middle of a sentence and you let out a PG-13 utterance or an R-rated expletive, you can continue recording and easily take them out later. When you're done recording, select the zoom tool from the Tools toolbar and highlight the portion of the track that contains the aberration. Then activate the selection tool, carefully select just the bit that you want to zap, and hit the Delete key. You can also remove unnecessary portions from the recording using *Audacity*'s trim tools. Suppose a track has a lot of silence at the beginning and the end. In such a case, select the region of the audio that you'd like to keep and click the Trim Outside Selection button from the Edit toolbar to trim all audio not in the selection.

Another common editing task is to remove background noise. Using *Audacity*'s Noise Removal tool, which has been significantly improved in the latest version, you can take out constant background sounds such as fan noise, carrier noise and such. Head over to Effect > Noise Removal, click on the Get Noise Profile button and select a region of the recording that only contains the noise. Then head back to Effect > Noise Removal and experiment with the noise removal parameters. Use the Preview button to listen to a few seconds of audio and tweak the parameters accordingly until you're satisfied.

Sometimes you might want to reposition sections within your recording. For example, if you've recorded a podcast with multiple sections in a particular order and then later decide to rearrange the order of these sections. Instead of re-recording the podcast, you can cut-copy-paste sections of the track just like you would in a text editor. Select the portion you wish to move and press Ctrl+X to cut it. Then move to the position where you want to insert it and press Ctrl + V to paste it.

Mixing tracks

If you wish to record from two microphones, you'll need a soundcard that can record from multiple channels or you can connect the microphones to a mixer which you can then hook up to your computer. If you're on a budget, you can record different sources in multiple tracks and then combine them with *Audacity*. So if you have a band, you can record vocals in one track, the guitar on another and the drums on the third. Once you have the three tracks, start a new project



Noise Removal

Step 1

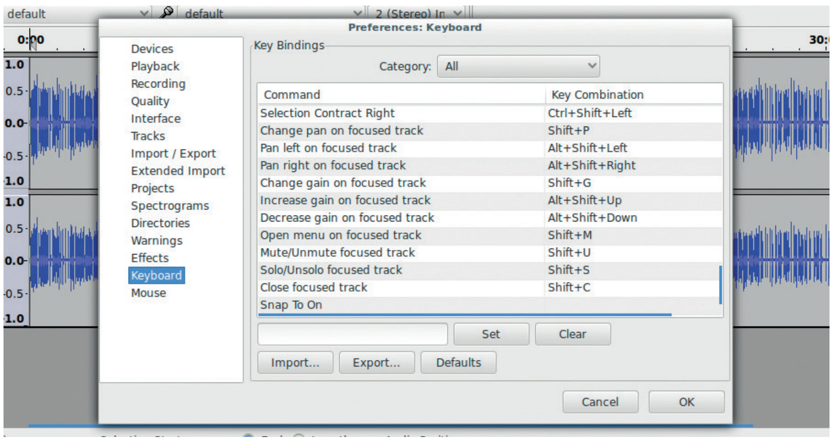
Select a few seconds of just noise so Audacity knows what to filter out, then click Get Noise Profile:

Step 2

Select all of the audio you want filtered, choose how much noise you want filtered out, and then click 'OK' to remove noise.

Noise reduction (dB):	24
Sensitivity (dB):	0.00
Frequency smoothing (Hz):	150
Attack/decay time (secs):	0.15
Noise:	[✓ Remove] Isolate

Don't be overwhelmed by the tweakable parameters of the various options and effects. Most of them can be applied only after trial-and-error.



Preferences: Keyboard

Key Bindings

Category: All

Command	Key Combination
Selection Contract Right	Ctrl+Shift+Left
Change pan on focused track	Shift+P
Pan left on focused track	Alt+Shift+Left
Pan right on focused track	Alt+Shift+Right
Change gain on focused track	Shift+G
Increase gain on focused track	Alt+Shift+Up
Decrease gain on focused track	Alt+Shift+Down
Open menu on focused track	Shift+M
Mute/Unmute focused track	Shift+U
Solo/Unsolo focused track	Shift+S
Close focused track	Shift+C
Snap To On	

If you plan to work with *Audacity* on a regular basis, make sure you familiarise yourself with its keyboard shortcuts.

and head to File > Import > Audio and select the first audio track. Repeat the process to import all three tracks and then hit Play to enjoy your masterpiece.

Another use of the mixing tracks features is when you wish to mix narration with some background music or overlay section announcements on top of a longer podcast. On the left of each track is the Track Control Panel. It presents relevant information about each track, including its sample rate and bit depth. Use the pull-down menu next to the Track title to rename the track.

Then there's the Mute button, which cuts the audio of the track from the mix when you hit the Play button. Conversely, the Solo button cuts out the rest and only plays the audio from the corresponding track. At the bottom is the track's Pan Control, which lets you adjust the track's audio either to the left or right. Above this is the track's Gain Control, with which you can control the relative volume of the track.

In order to blend tracks, begin by first splitting the main track into multiple clips. Use Edit > Clip Boundaries > Split and insert a split where you need to insert audio from another track. You can split a track into as many clips as you want. Similarly split the other track, then use the Time Shift tool from the Tools toolbar, at the top of the *Audacity* interface, make room for the bits from the other track.

If you have places in the track where the music overpowers the voice, you can fade down the music when the voiceover begins and turn it back up at the end. For this, select the Envelope tool, next to the selection tool. When you activate the tool you'll notice horizontal lines at the top and bottom of each track. These are representations of the track's volume, which by default are at 100%. Then in the music track, head over to the area where you need to fade the music and click, hold and drag at either the top or the bottom line to reduce the volume of the music. You can also click inside the music track envelope to add control points and drag the envelope to create a desired shape. Experiment with the shape of the envelope until you're satisfied.

All this might sound a bit confusing at first, but this flexibility enables you to create complex compositions by mixing multiple smaller tracks.

LV PRO TIP

Go to Tracks > Add Label At Selection to mark positions in a track. This is useful for annotating bits in a long piece of audio such as an interview.

LV PRO TIP

Go to Tracks > Mix and Render before exporting your recording. This will merge all your tracks into one single track that you can then clip, normalise, and enhance as a whole.

HACK AUDIO FILES WITH SOX

Surgically manipulate audio with this command-line gem.

MAYANK SHARMA

Sox is the Swiss Army knife of command-line sound editing tools. While it lacks the more powerful features and the graphical interface of *Audacity*, *Sox* is incredibly versatile and has the batch-processing advantage of a command line utility. The tool is adept at several common sound editing tasks such as converting files into different formats and applying sound effects.

Grab Sox from your distro's mirrors either with **sudo apt-get install sox** or **sudo yum install sox** depending on your distro. Also grab the **libsox-fmt-all** package, which includes all of the *Sox* libraries and the ability to work with various file formats including MP3.

Once you have the tool and the libraries, converting files from one format to another is a simple affair. The command **sox some-file.mp3 great-file.ogg** will encode the MP3 file into Ogg. As you can see, *Sox* recognises the formats of the input file and the output file from the extension of the file.

Before you can manipulate a file, you'll need to know its encoding information. The command

```
sox --i some-file.ogg
```

will list various details about the file including its sample rate, file size, bit rate, and more.

If you wish to reduce the size of the audio files on your disk, one way to do so is to reduce their sampling rate. The command

```
sox some-file.ogg -r 16k downsampled-file.ogg
```

will change the sampling rate of the file down to 16kHz. Now check the file again with the **--i** switch and note the changed sampling rate and the reduction in file size.

One of the most popular ways of using *Sox* is to do on-the-fly processing. For example, we use the *PiFM* script, which enables us to use the Raspberry Pi as a low-power FM transmitter. It's very easy to implement and enables us to listen to audio even on older FM players that don't have Wi-Fi. However, the script can only play audio files that are 16-bit 22050Hz mono and in the WAV format. Instead of first converting all our music into WAVs to conform to these restrictions, we use *Sox* to process the file on the fly and then pass it on to the script. The command:

```
sox -t ogg SomeSong.ogg -t wav -r 22050 -c 1 - | sudo ./pifm
- 100.1
```

will translate the Ogg file into a WAV file, change its sampling rate to 22050Hz and down mix the track to a mono channel. The converted audio is sent to the standard output, denoted by the hyphen sign (**-**) and is then piped (**|**) into the standard input of the **pifm** command. The script will then broadcast it on the 100.1 FM frequency.

You don't even need the file on your disk. If you wish to broadcast the Linux Voice podcast, you can do so with the command:

```
sox -t ogg http://www.linuxvoice.com/episodes/lv_s03e05.ogg -t
wav -r 22050 -c 1
```

The only difference between this command and the previous example is that instead of pointing to a local Ogg file, we are now pointing to one that resides online. Similarly, you can use *Sox* to convert an online radio station and pipe it to the *PiFM* script, like so:

```
sox -t ogg http://network.absoluteradio.co.uk/core/audio/ogg/
live.pls?service=a6bb -t wav -r 22050 -c 1
```

Not all stations broadcast Ogg streams. To convert MP3 streams all you need to do is change the type of the input file you are converting with *Sox*, like so:

```
sox -t mp3 http://www.ndr.de/resources/metadaten/audio/m3u/
ndrloop5.m3u -t wav -r 22050 -c 1
```

Audiophiliac remedies

Besides converting music, there are various other ways you can manipulate audio with *Sox*. You can, for example, use it to extract a part of audio with the trim switch. The trim switch accepts two parameters: the starting point of the portion you wish to keep and the second denotes its length. For example,

```
sox original.ogg beginning.ogg trim 0 10
```

will extract the first 10 seconds from the **original.ogg** file. Conversely, if you wish to clip the first 10 seconds and keep the rest, first find the duration of the track in seconds with the

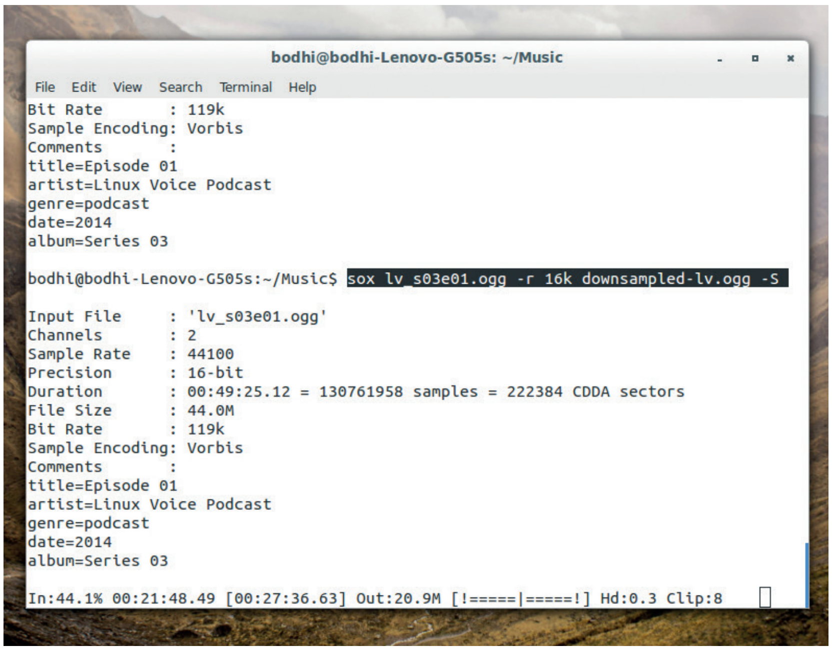
```
sox file.ogg -n stat
```

command. Make a note of the value listed next to the Length (seconds) parameter. Let's assume ours reads 2965.12. Then use the command

```
sox original.ogg silence-clipped.ogg trim 20 2945.12
```

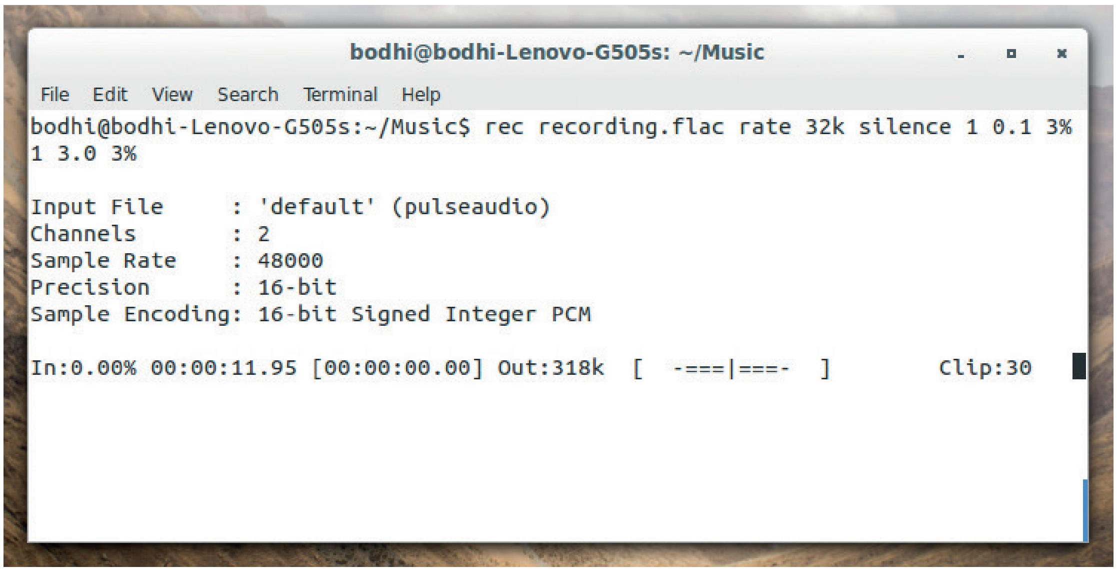
LV PRO TIP

Use the **avg** parameter when converting a multi-channel Stereo track to Mono.


Use the **-S** switch to keep an eye on the progress of ongoing activity.

bodhi@bodhi-Lenovo-G505s: ~/Music
File Edit View Search Terminal Help

```
Bit Rate       : 119k
Sample Encoding: Vorbis
Comments       :
title=Episode 01
artist=Linux Voice Podcast
genre=podcast
date=2014
album=Series 03

bodhi@bodhi-Lenovo-G505s:~/Music$ sox lv_s03e01.ogg -r 16k downsampled-lv.ogg -S

Input File     : 'lv_s03e01.ogg'
Channels       : 2
Sample Rate    : 44100
Precision      : 16-bit
Duration       : 00:49:25.12 = 130761958 samples = 222384 CDDA sectors
File Size      : 44.0M
Bit Rate       : 119k
Sample Encoding: Vorbis
Comments       :
title=Episode 01
artist=Linux Voice Podcast
genre=podcast
date=2014
album=Series 03

In:44.1% 00:21:48.49 [00:27:36.63] Out:20.9M [!=====|=====!] Hd:0.3 Clip:8
```


bodhi@bodhi-Lenovo-G505s: ~/Music
File Edit View Search Terminal Help

```
bodhi@bodhi-Lenovo-G505s:~/Music$ rec recording.flac rate 32k silence 1 0.1 3%
1 3.0 3%

Input File     : 'default' (pulseaudio)
Channels       : 2
Sample Rate    : 48000
Precision      : 16-bit
Sample Encoding: 16-bit Signed Integer PCM

In:0.00% 00:00:11.95 [00:00:00.00] Out:318k  [  -===|===-  ]        Clip:30
```

You can use *Sox* to start recording when it detects audio and stop when it detects a specified length of silence.

Here 20 is the starting point of the recording we wish to preserve, and 2945.12 is the original length minus the first 20 seconds.

An easier way to do this is to specify the length of the track you wish to clip. So if you wish to lop off the first 10 seconds, use

```
sox original.ogg start-clipped.ogg trim 10
```

Similarly you can also clip from the end of the track. The command

```
sox original.ogg end-clipped.ogg reverse trim 5 reverse
```

will reverse the track, trim 5 seconds from the reversed track and reverse it back again, which effectively removes the last 5 seconds of audio.

You can also use *Sox* to merge multiple audio files into one. The command

```
sox -m background-music.ogg commentary.ogg great-show.ogg
```

will combine the **background-music.ogg** file with **commentary.ogg** so that they overlap each other in the **great-show.ogg** file.

Sox also lets you add a fade-in and fade-out effect to a track. The command

```
sox music.ogg with-fade-in.ogg fade 10
```

will slowly raise the volume from zero to maximum in 10 seconds. If you wish to add a fade-out effect as well, you'll need to know the length of the track. Assuming the length of the track to be 400 seconds, the command

```
sox music.ogg with-fades.ogg fade 10 400 5
```

will add a 5 second fade-out at the end of the track in addition to the 10 second fade-in.

There are several other ways you can alter the audio, some more useful than others. For example, you can also speed up or slow down a file by changing its pitch by a particular factor. The default pitch factor of a file is 1.0. The command

```
sox normal.ogg faster.ogg speed 2.0
```

will double its speed and also reduce the duration of the clip by half.

Sox also includes two other command-line utilities. There's the **play** utility, which is basically just a shell script that calls *Sox* to play an audio file over the default audio device. So

```
play some-audio.ogg
```

will play the file. Since it's just a script around *Sox* you can use **play** to hear a file without actually modifying it. So if you wish to boost the bass of a song but aren't sure of the value you want to boost it by, you can use the **play** command. The command

```
play some-audio.ogg bass +3
```

will play the file after boosting the bass 3 notches without touching the original file.

Sox also comes with a recording utility. The command

```
rec voice.ogg
```

will start recording from the default recording device. It'll record until you use the Ctrl + C hot key to abort the operation. The recorded clip will be saved in the **voice.ogg** file. You can also limit the duration of the recording. The command

```
rec record-voice.ogg trim 0 10
```

will record for only 10 seconds. And you can even use other switches as well. The command

```
rec -r 16k -c 1 voice.ogg trim 0 10
```

will record for 10 seconds and then downsample the recording to a single channel audio with a 16k sample rate.

There's so much more that you can do with *Sox*. We'd run out of paper if we attempt to cover everything you can do with the utility. But rest assured that if you need to modify an audio file, chances are you can do it with *Sox* LV.

"If you need to modify an audio file chances are you can do it with Sox."

Mayank Sharma has been finding productive new ways to mess about with free software for years now.

LV PRO TIP

Add a small amount of stereo reverberation effect to a mono file with the -w switch to make it sound more natural.

LV PRO TIP

You lose some information when converting to some formats, such as like MP3, that compress audio. Know that converting back again from such formats will not produce an exact copy of the original.

/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!

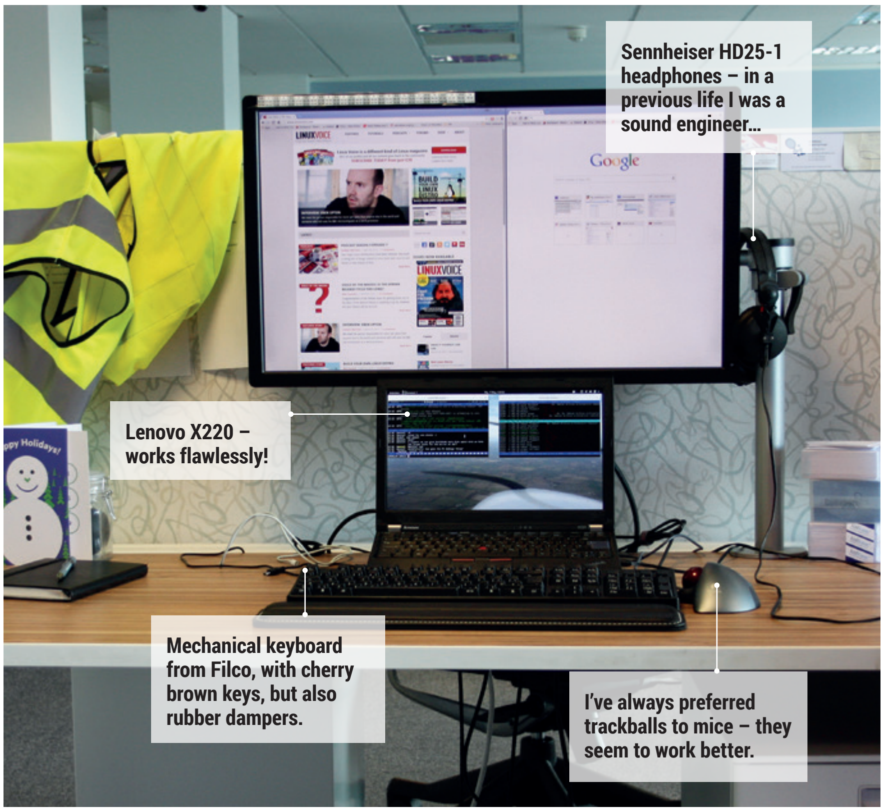
Last year I backed a Kickstarter for a 3D printer. In fact I backed several, but there was one I was particularly keen on seeing. Everything would be straightforward and 'Just Work', etc. So far it's six months late, but hey, I was expecting that.

What I wasn't expecting was the slide on their promises. When I backed the Kickstarter, it proclaimed it would work with Windows/Mac/Linux. I didn't pay that much attention to it at the time, because after all, what idiots would produce a 3D printer without making it compatible with all the industry-leading 3D software out there. Let's face it, home 3D printing was invented (whatever those Makerbot sell-out turncoats would have you believe) by the RepRap project, using open hardware and open source software. In Bath, as it happens.

Fast forward a few seasons, and a website is duly created. Worryingly it now says "Windows/Mac. Linux support is also likely". When the printers start shipping, I discover that they have spent at least 12 months hiring developers and writing their own software! Not their own front-end or anything, but a complete Windows application, helpfully written in something that will be hard to port to Linux.

The developers have kindly enabled an option not to ship the printer before the software is ready for those platforms, but as they can't say when that will be, and haven't even started working on it, I don't want to wait another year...

The worst part of all is that all this info has had to be dragged out of them. The 'we are too busy to respond' excuse is pointless and foolish when you have a website and a forum – it actually saves time! If you are starting a Kickstarter, don't be an idiot. And also, remember to reply to all the backers. Even the idiots.


Sennheiser HD25-1 headphones – in a previous life I was a sound engineer...
Lenovo X220 – works flawlessly!
Mechanical keyboard from Filco, with cherry brown keys, but also rubber dampers.
I've always preferred trackballs to mice – they seem to work better.

My Linux Setup Neil McGovern

The Debian Project Leader shares his hacking environment.

Q What version of Linux are you currently using?

A I'm currently running the latest release of Debian 8 "Jessie", unsurprisingly.

Q And what desktop do you currently use?

A I'm using Gnome 3.14 – I've tried a few different ones, and it seems to be the most complete at the moment.

Q What was the first Linux setup you ever used?

A The first install I did was Mandriva, about 13 years ago. However, I moved on to Debian after about a year, and have been running that ever since.

Q What Free Software/open source can't you live without?

A As most of my job involves communications, and a lot of it, I'll have to go with a combination of *Irssi* for IRC and *Mutt* for email. Text-based interfaces are the only way I'm able to stay on top of things!

Q What do other people love but you can't get on with?

A Not wanting to cause a holy war, but I guess *Emacs*. I'm a *Vim* user, and I just never seem to have enough fingers to make *Emacs* work for me.

Arduino Programming Cheat Sheet

Primary source: Arduino Language Reference
http://arduino.cc/en/Reference/

Structure & Flow

Basic Program Structure

```
void setup() {
  // runs once when sketch starts
}
void loop() {
  // runs repeatedly
}
```

Control Structures

```
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
do { ... } while ( x < 5);
for (int i = 0; i < 10; i++) { ... }
break;  // exit a loop immediately
continue;  // go to next iteration
switch (myVar) {
   case 1:
     ...
     break;
   case 2:
     ...
     break;
   default:
     ...
}
return x; // just return; for voids
```

Operators

General Operators

```
=  (assignment operator)
+  (add)          -  (subtract)
*  (multiply)     /  (divide)
%  (modulo)
== (equal to)     != (not equal to)
<  (less than)    >  (greater than)
<= (less than or equal to)
>= (greater than or equal to)
&& (and)    || (or)    ! (not)
```

Compound Operators

```
++ (increment)
-- (decrement)
+= (compound addition)
-= (compound substraction)
*= (compound multiplication)
/= (compound division)
&= (compound bitwise and)
|= (compound bitwise or)
```

Bitwise Operators

```
&  (bitwise and)      |  (bitwise or)
^  (bitwise xor)      ~  (bitwise not)
<< (shift left)       >> (shift right)
```

Variables, Arrays, and Data

Data types

```
void
boolean   (0, 1, true, false)
char      (e.g. 'a'  -128 to 127)
int       (-32768 to 32767)
long      (-2147483648 to 2147483647)
unsigned char  (0 to 255)
byte           (0 to 255)
unsigned int   (0 to 65535)
word           (0 to 65535)
unsigned long  (0 to 4294967295)
float     (-3.4028e+38 to 3.4028e+38)
double    (currently same as float)
```

Qualifiers

```
static    (persists between calls)
volatile  (in RAM (nice for ISR))
const     (make read only)
PROGMEM   (in flash)
```

Arrays

```
int myInts[6]; // array of 6 ints
int myPins[]={2, 4, 8, 3, 6};
int mySensVals[6]={2, 4, -8, 3, 2};
myInts[0]=42;  // assigning first
               // index of myInts
myInts[6]=12;  // ERROR! Indexes
               // are 0 though 5
```

Constants

```
HIGH  | LOW
INPUT | OUTPUT
true  | false
143          (Decimal)
0173         (Octal - base 8)
0b11011111   (Binary)
0x7B         (Hexadecimal - base 16)
7U           (force unsigned)
10L          (force long)
15UL         (force long unsigned)
10.0         (force floating point)
2.4e5        (2.4*10^5 = 240000)
```

Pointer Access

```
&  (reference: get a pointer)
*  (dereference: follow a pointer)
```

Strings

```
char S1[8] =
  {'A','r','d','u','i','n','o'};
  // unterminated string; may crash
char S2[8] =
  {'A','r','d','u','i','n','o','\0'};
  // includes \0 null termination
char S3[]="Arduino";
char S4[8]="Arduino";
```

Built-in Functions

Pin Input/Output

```
Digital I/O (pins: 0-13 A0-A5)
  pinMode(pin,[INPUT, OUTPUT])
  int digitalread(pin)
  digitalWrite(pin, value)
    // Write HIGH to an input to
    // enable pull-up resistors
Analog In (pins: 0-5)
  int analogRead(pin)
  analogReference(
    [DEFAULT, INTERNAL, EXTERNAL])
PWM Out (pins: 3 5 6 9 10 11)
  analogWrite(pin, value)
```

Advanced I/O

```
tone(pin, freqhz)
tone(pin, freqhz, duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin,
  [MSBFIRST,LSBFIRST], value)
unsigned long pulseIn(pin,
  [HIGH,LOW])
```

Time

```
unsigned long millis()
  // overflows at 50 days
unsigned long micros()
  // overflows at 70 minutes
delay(msec)
delayMicroseconds(usec)
```

Math

```
min(x, y)    max(x, y)    abs(x)
sin(rad)     cos(rad)     tan(rad)
sqrt(x)      pow(base, exponent)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
```

Random Numbers

```
randomSeed(seed) // long or int
long random(max)
long random(min, max)
```

Bits and Bytes

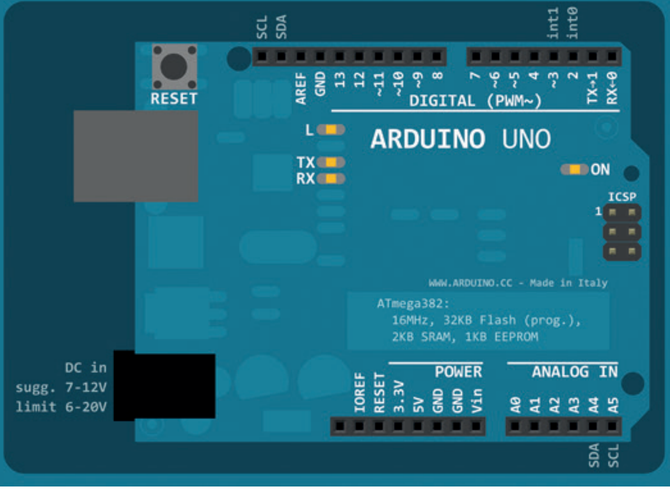
```
lowByte(x)     highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn)  // bitn: 0=LSB 7=MSB
```

Type Conversions

```
char()        byte()
int()         word()
long()        float()
```

External Interrupts

```
attachInterrupt(interrupt, func,
  [LOW, CHANGE, RISING, FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```



Libraries

Serial (communicate with PC or via RX/TX)

```
begin(long Speed) // up to 115200
end()
int available() // #bytes available
byte read() // -1 if none available
byte peek()
flush()
print(myData)
println(myData)
write(myBytes)
SerialEvent() // called if data rdy
```

SoftwareSerial (serial comm. on any pins)

```
  (#include <softwareSerial.h>)
SoftwareSerial(rxPin, txPin)
begin(long Speed) // up to 115200
listen()       // Only 1 can listen
isListening() // at a time.
read, peek, print, println, write
  // all like in Serial library
```

EEPROM (#include <EEPROM.h>)

```
byte read(intAddr)
write(intAddr, myByte)
```

Servo (#include <Servo.h>)

```
attach(pin, [min_uS, max_uS])
write(angle)  // 0 to 180
writeMicroseconds(uS)
  // 1000-2000; 1500 is midpoint
int read()     // 0 to 180
bool attached()
detach()
```

Wire (I²C comm.) (#include <Wire.h>)

```
begin()       // join a master
begin(addr) // join a slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(myByte)             // Step 2
send(char * mystring)
send(byte * data, size)
endTransmission()        // Step 3
int available() // #bytes available
byte receive()  // get next byte
onReceive(handler)
onRequest(handler)
```

debian

OpenStreetMap

KODI

freegle

GnuPG

LibreOffice
The Document Foundation

UKNOF

mySociety

FreeBSD

PROTECT YOUR BITS. SUPPORT ORG
OPEN RIGHTS GROUP

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**

BusTimes.org

Traintimes .org.uk

:BYTEMARK

*Excl VAT

LibreFM