ASTRO PI: SEND YOUR IDEAS INTO SPAAAAAAAAAAAACE!

LINUX VOICE

INSIDE LINUX MALWARE P36

March 2015

FREE SOFTWARE | FREE SPEECH

OPENELEC
BUILD A MEDIA CENTRE
Control the idiot box with this brilliant Linux media setup

EMBEDDED
CODE IN ASSEMBLER
Earn big money by writing tiny, super-efficient code

2015 THE YEAR OF LINUX

Uncover the future of Linux and Free Software – by the experts, developers and geeks who are making it happen.

40+ PAGES OF TUTORIALS

+
ARDUINO Write an interface for cheap hardware
KMAIL Take complete control of your email inbox
BAKE A PI We go inside the Raspberry Pi factory in darkest Wales

FREEDOM!
INSIDE THE EFF
A campaigning digital rights organisation that isn't full of nutters!

INTERVIEW
LENNART POETTERING
The creator of Systemd on flamewars, init systems and why Debian is behind the times

ISSN 2054-3778
March 2015 £5.99 Printed in the UK
9 772054 377001 03

ANDREWS & ARNOLD LTD

will make you the

LINE KING

BE THE MASTER OF YOUR OWN TELEPHONE

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

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

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

HAKUNA MATATA

Welcome to Linuxlandia

The March issue

LINUX VOICE

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

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

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

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

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

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.

This is our twelfth issue, making it one year since our first magazine following our successful Indiegogo campaign. We've produced over one million words about Linux and Free Software, 23+ podcasts, numerous videos and two audio editions. We've sent over 60,000 emails (including those to subscribers) and received more than 30,000 between us. We've served terabytes of data (thanks Bytemark!), we've released two issues for free under a Creative Commons licence, we've helped sponsor conferences and we've spoken to many, many people within the community. It's been a crazy, manic, challenging, exhausting and ultimately rewarding rollercoaster of a year.

Without wanting to diminish the value of the awesome magazine currently in your hands – before you've even got to the contents pages – we've got even better things planned for the next 12 issues, starting with the commencement of our profit sharing scheme next month. We couldn't have done any of this without you, our readership. So… thank you! And here's to 2015!

Graham Morrison
Editor, Linux Voice

SUBSCRIBE ON PAGE 64

What's hot in LV#012

MAYANK SHARMA
"Ben's practical insight into how malware works is fascinating. It's and a must-read if you've any interest in security." **p36**

BEN EVERARD
"Lennart Poettering couldn't be any more 'on-topic', considering the Systemd controversy. He was very candid in our interview." **p42**

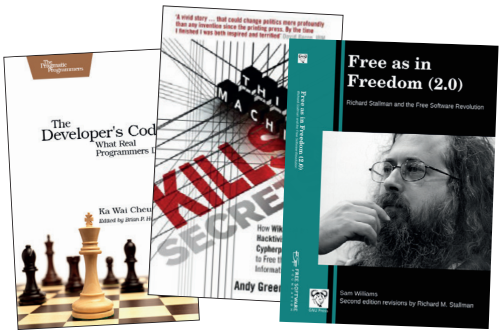
MIKE SAUNDERS
"OLED displays are cheap and brilliant, making our tutorial on hacking them perfect for my Arduino Mario clone." **p100**

CONTENTS

March LV012

A nice Donegal tweed, pipe and comfy armchair. Shall we begin?

SUBSCRIBE ON PAGE 64


The Developer's Code
Ka Wai Cheung
Kill Switch
Free as in Freedom (2.0)
Richard Stallman and the Free Software Revolution
Sam Williams

NEWSANALYSIS

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

Opinion

Liberté, egalité, fraternité

Terrorism must not become an excuse to give up essential liberties.

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

With the rest of the Linux Voice team, I am appalled and horrified by the attack on *Charlie Hebdo* in Paris, which was perpetrated just as this issue was going to the printer. Settling scores with violence is the recourse of ignorant cowards – lower than animals. I am heartbroken for every person affected, and we extend our sympathy to the families, friends and fans of the people murdered.

This was without doubt intended as an act of terrorism. Some politicians seem to act as if "terrorism" means a terrible crime committed by someone who doesn't fit the speaker's own racial and religious profile. Just because something induces terror in some or many people, that doesn't make it terrorism. That diminishes the concept as well as grouping routine crime – for which society has millennia of experience and solutions – into the same bucket as a more subtle and serious phenomenon that preys on the meshed society.

Terrorism isn't just performing a terrifying act. It's provoking society's immune system into attacking itself, making its defence systems attack the values and people they are supposed to be defending. Terrorism is like an autoimmune disorder of democracy. You don't fight terrorism by attacking the virus; you fight it by strengthening the immune system. It is too easy to succumb to the virus. It seeks to provoke us into destroying ourselves, using the very freedoms we value as a weapon to provoke their extinction. I refuse to be terrorised and decline the opportunity to hate. I choose not to succumb to the terrorism disease and I invite you to join me in these choices.

As I write, certain voices are calling for blanket surveillance, for bans on discussion of some topics, for the blocking of certain websites, and more. These reactions are exactly the auto-immune response terrorism seeks to stimulate. Its architects do so because they have already dismissed the possibility (or the value) of democratic society and want the restrictions to go ever deeper as a tool both to damage capitalism and to radicalise more people. Ironically, banning speech that radicalises youth is likely to itself radicalise youth.

But even if the bans were not intrinsically self-defeating, we should still oppose them. To advocate a ban on anything on the internet is to put one's faith in magical thinking. The fabric of the meshed society – the internet, the worldwide web, peer-to-peer protocols, wireless networks – is all designed for resilience. It "treats obstructions as damage" and comes complete with the tools to route round it.

So any requirement for technical measures to impose a ban is a requirement to defeat the resilience of the meshed society itself. Encryption, VPNs and private communications must all be prevented or corrupted. Ultimately that requires general purpose computing devices be eliminated, replaced with devices that can't be modified by their owners. Richard Stallman calls this "treacherous computing"; Cory Doctorow warns us of the "coming war on general-purpose computing".

"To advocate a ban on anything on the internet is to put one's faith in magical thinking."

Beware straw men

As to surveillance: the murderers in France were known to the security services, who had been using existing laws to apply justified surveillance. There are circumstances where it's appropriate in a democratic society for accountable surveillance to be used to prevent crime. But there are limits. Gathering all metadata on all electronic connections is possible in a way the physical equivalents never were before. But is disproportionate democratically even if it is economically feasible.

In particular, it creates a capability for triangulation with other data from other contexts that makes Bentham's panopticon look trivial in comparison. The result is a society in which every form of speech can be cropped out of context and used to ensure conformity. The very existence of blanket surveillance chills democracy.

We should respond to this act of hate, which is as indefensible to anyone who embraces one of the world's religions as to those who reject them all, by ensuring we do not succumb to the self-destructive reactions perpetrators of terrorism want to provoke. That includes apparently reasonable demands for technical measures to prevent publication of hate speech, ban radicalising web sites and increase blanket surveillance. Behind the barbarism of terrorism, a brutal yet seductive subtlety of purpose lurks. We must not succumb.

Ubuntu • Astro Pi • Linux laptops • Devuan • Rocket • FOSS financing

CATCHUP

Summarised: the biggest news stories from the last month

1 Canonical announces Snappy Ubuntu Core

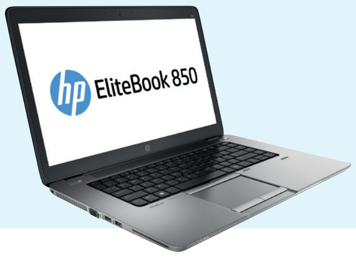
Imagine a trimmed-down version of Ubuntu, geared towards deploying applications via Docker containers. Imagine that it supports transactional updates, so if something goes wrong when during the next round of updates, you can easily revert back to the previously working versions. Well, this is real, and it's called Snappy Ubuntu Core. Canonical certainly has its fingers in many pies at the moment, but this looks promising (see p40 for more). **www.ubuntu.com/cloud/tools/snappy**

2 Raspberry Pis head into space, running code written by kids

You can find Raspberry Pis virtually everywhere these days, and soon a couple of the dinky computers will be sent up to the International Space Station. Even better: they'll be running code to perform experiments written by schoolchildren in primary and secondary schools in the UK. Called Astro Pi, this project should be a fantastic way to get youngsters involved in programming. **www.raspberrypi.org/astro-pi**

3 HP announces EliteBook 850 running Linux

It's rather expensive (starting at $1,709) but it's available with SUSE Enterprise Linux 11. In any case, the more choice we Linux users have when buying hardware, the better. **http://tinyurl.com/kymfuc2**



4 Devuan team publishes first progress report

Yes, the now-famous Debian fork that aims to provide "init freedom" (ie it doesn't use *Systemd*) has lasted longer than many expected. The Devuan team has even delivered its first progress report, which details development going into *Loginkitd* (a standalone alternative to logind, using *ConsoleKit2* as a back-end) and other software projects. According to the financial report, Devuan received €4,500 in donations up to December. **https://devuan.org/newsletter_22dec.html**

5 Red Hat adds speed boosts to Glibc

This might not sound like the most exciting news story in the world, but *Glibc* is a tremendously important piece of software, as it's the base C library that almost every piece of software on a GNU/Linux system links to. Now an engineer at Red Hat has worked on improving the performance of some of *Glibc*'s math routines, making some of the functions up to eight times faster than before. The report is heavy reading, but worth a look: **http://tinyurl.com/m8jrj7x**

6 Snowdrift.coop aims to help fund free software

Many FOSS projects are funded entirely by donations, but only receive occasional or one-off payments from users. A new website at **https://snowdrift.coop** wants to make FOSS projects more sustainable in the long run, by asking users to make monthly pledges to particular projects. And if you're a patron of *FooApp*, for example, making a monthly payment from your account on Snowdrift, you'll also pay a little extra whenever someone else becomes a patron too.

7 Crowdfunded laptop aims to respect your freedom

If you've ever tried to buy a laptop that works perfectly with Linux, you'll know it can be a chore. The Librem 15 is a crowdfunded high-end laptop starting from $1,899, and at the time of writing, the campaign had reached 71% of its goal. The machine will supposedly run on 100% Free Software – no binary blobs required. We'll try to get our hands on a unit for review in a future issue of the magazine. Check it out here: **www.crowdsupply.com/purism**

8 CoreOS team ditches Docker, creates Rocket

Software containers are all the rage at the moment. They let you run applications (typically on servers) in isolated environments with well-defined sets of libraries, so you can update them easily and they can't trample over other programs. Docker has been the big success story so far, but the CoreOS team has decided that Docker's scope has grown too large, and a replacement is needed: Rocket. It's a controversial move, but it's explained well here: **https://coreos.com/blog/rocket**

DISTROHOPPER

Our pick of the latest releases will whet your appetite for new Linux distributions.

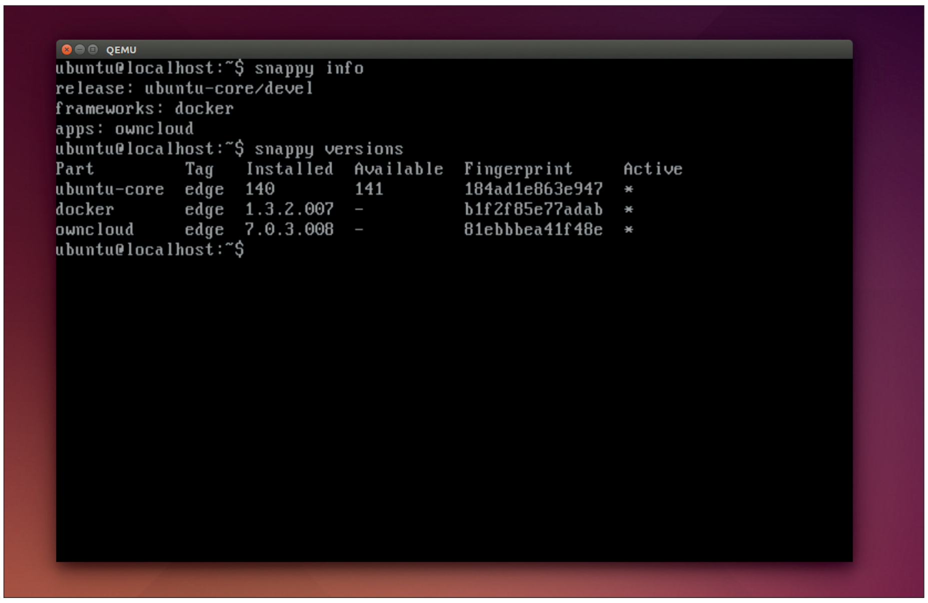
Ubuntu Core

Like Ubuntu Touch, for servers.

Ubuntu Core is mash-up of Ubuntu Server and Ubuntu Touch (the smartphone OS). Yes, we know that sounds weird, but hear us out. Underneath the touch-driven interface, Canonical has been working on the structure of Ubuntu to make it work better on phones. This means the package manager works in a different way. No longer can packages change files and settings across the system. On Ubuntu Touch, they're more self contained and controlled by AppArmor to make sure they're well behaved. The package manager is also transactional, which means that it's easier to roll packages back to previous versions, and you shouldn't have any problems if a package fails halfway through an install.

It turns out that a lot of the stuff that makes a package manager good for a phone also makes it good for a server, so Canonical has adapted the package manager from touch to make Snappy, which powers Ubuntu Core. Snappy isn't a package manager in the usual sense of the word. It's designed to work alongside frameworks that package applications. The first of these is Docker, so through Snappy, you can install Docker, then have access to the whole range of Docker apps. Canonical has promised more Frameworks in the future, but at the time of writing, hadn't given any details about which would be available.

Ubuntu Core makes spinning up virtual servers really easy. A production version isn't ready yet, but you can grab the alpha (maybe beta by the time you read this) from **www.ubuntu.com/cloud/tools/snappy**.



```
ubuntu@localhost:~$ snappy info
release: ubuntu-core/devel
frameworks: docker
apps: owncloud
ubuntu@localhost:~$ snappy versions
Part         Tag   Installed  Available  Fingerprint     Active
ubuntu-core  edge  140        141        184ad1e863e947  *
docker       edge  1.3.2.007  -          b1f2f85e77adab  *
owncloud     edge  7.0.3.008  -          81ebbbea41f48e  *
ubuntu@localhost:~$
```

For more on the logic behind Ubuntu Core and Snappy, turn to our FAQ on page 40.

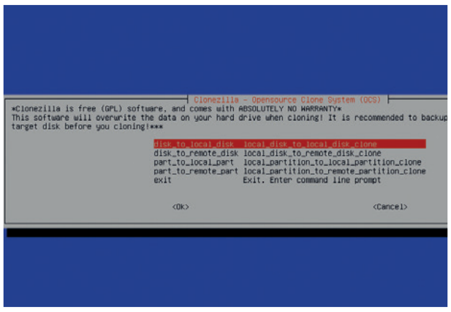
Clonezilla

Create snapshots of hard drives with a live distro.

Do one thing well. That's the essence of the Unix philosophy. It's not usually applied to operating systems themselves, but in the case of Clonezilla it could be. This is a distro for making images of hard drives, and then re-imaging drives with these images (technically that's two things, but they're close enough that we count them as one). This makes it the distro of choice for IT workers who need to set up a large number of identical machines. It can also be used as a backup tool.

It's a live distro that you boot and use alongside some external storage where the image is saved. A simple curses interface guides you through the process, so you don't need to know how to use any command line tools to use Clonezilla. There are two levels of interface, beginner mode where you just get the basic options, and expert mode where you have access to more advanced features. Together these give you enough options to perform most tasks without overwhelming new users.

Clonezilla hasn't really changed much in as long as we can remember, but the latest version brings in a range of improvements to the underlying OS from the latest Debian Sid (on which it's built), including the 3.16 kernel, so it should support more hardware, and run a little faster. We can confidently say that Clonezilla is the best Linux distribution for re-imaging hard disks.



```
Clonezilla - Opensource Clone System (OCS)
*Clonezilla is free (GPL) software, and comes with ABSOLUTELY NO WARRANTY*
This software will overwrite the data on your hard drive when cloning! It is recommended to backup
target disk before you cloning!***

            disk_to_local_disk   local_disk_to_local_disk_clone
            disk_to_remote_disk  local_disk_to_remote_disk_clone
            part_to_local_part   local_partition_to_local_partition_clone
            part_to_remote_part  local_partition_to_remote_partition_clone
            exit                 Exit. Enter command line prompt

              <Ok>                                  <Cancel>
```

The curses interface means you don't have to remember any commands, but you still get you access to a wide range of options.

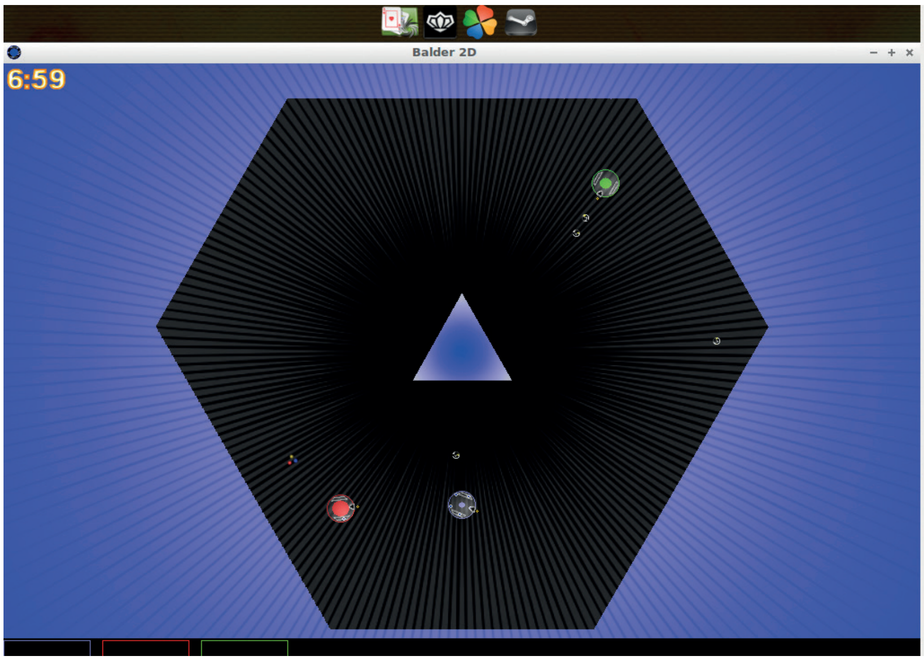
Sparkey: GameOver

Give in and satisfy your inner gamer.

GameOver is a new version of Sparkey designed for gaming. For this, it has just about every option (both closed and open source) waiting to go. There are almost 100 games on the live disc ready to be played. There are also installers for the Steam and Desura clients to give you access to games from these stores. Play on Linux is installed, which makes it easier to run Windows games, and there's a specially designed app to make it easy to install a wide range of emulators (*APTus Gamer*).

We haven't been able to work out just how many games this means there are available for this distro, but we're confident that it will be more than any one person will ever be able to play.

It's not just the variety of games that makes GameOver a good distro for playing on. GameOver also includes the Liquorix repository. This is a build of the Linux kernel that's designed to have the best performance under desktop loads, including gaming. You'll need to install the distro to get access to this as it's not used by default on the live version.



Balder, one of the installed games, is like a multi-player version of asteroids played in zero gravity.

Like all Sparkey releases, GameOver is built on Debian, so as well as games, there's also all the software you'd normally expect.

Right now, Sparkey Game Over is probably the quickest way to get a Linux gaming system installed.

Fedora Core 1 The first community release from the world's biggest Linux company.

Way back in the mists of time, you could download Red Hat Linux for free. This, the company decided, was not an optimal business strategy, so Red Hat Linux became Red Hat Enterprise Linux, and was only released to customers paying a hefty fee. The community wasn't abandoned though: Red Hat created Fedora Core, a new distribution that included all the latest software that hadn't yet made it to Red Hat Enterprise Linux, and they made this new distribution available for free (as in beer as well as in speech).

Fedora core 1 (codename Yarrow) came out in 2003 on three CDs. This is a bit more than most modern distros, but back then, many computers didn't have fast internet connections, so downloading new software was a luxury rather than an expectation.

Live distros weren't yet common, so trying Fedora meant installing it (this also makes it easier to split the distro up across multiple CDs). We selected a Personal install (as opposed to Workstation or Server), so we only needed the first two discs to get a desktop. Our memory of Linux installers of this era isn't pleasant, but we found the Fedora installer quite straightforward, although it's a much more involved process than most modern installers (when did boot diskettes stop being a thing?).

Considering it took 2 CDs to install it, Yarrow seems a little short on software. Firefox hadn't yet reached version 1.0, so wasn't considered stable enough. The now-defunct *Mozilla* browser served as the default method for accessing the web. *OpenOffice.org 1.1* provided office functions, and there's a wide variety of games, but that's about it. There's plenty of other stuff in the menus, but it's mostly simple accessories. The open source ecosystem was a lot smaller back then.

Fedora
ben's Home
Start Here
Wastebasket
Fedora - Mozilla
Fedora is now more focused
Less setup, more innovation. Choose a flavor of Fedora streamlined for your needs, and get to work right away.
WORKSTATION
12 years on, Fedora's no longer features a red fedora as the start menu.

GAMING ON LINUX

The tastiest brain candy to relax those tired neurons

FORGET THE SWAN

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

Has it been bothering you how many different package managers you have to use to install games? You have Steam, Desura, GOG, and many other websites to pick your games from, but having so many different places to install and launch games from can be a real nuisance.

We're delighted then to introduce *Lutris*, an open source application built specifically for Linux to help you manage your vast library of games, and it does make a lot of things simpler. *Lutris* doesn't just simplify native games either, but the developers' is to also support installing games with *Wine*, much like the application *PlayOnLinux* does.

Lutris also includes support for browser-based games, and retro emulators as well. Having all of your games launching from one place makes it a nifty application to have, but you still need to manually search to find the executable; once this is done though, launching is a simple click of a button.

Hopefully they will polish up the process and make it as easy as possible in future, as the looks leave a little to be desired.

We mentioned it was open source, so any willing developers can help polish up the experience with the source code, the source has been conveniently placed on GitHub. Check it out here: **https://lutris.net** and be sure to let us know what you think to it. We look forward to seeing this project progress.
http://forums.linuxvoice.com

This War of Mine

As bleak as you can get in a game.

You've played survival games before, sure, but you've never played anything quite as harsh as *This War of Mine*. There are masses of survival games around now, so it's good to see a game try and do things a little bit differently, and succeed at it. Most survival games are all about being "open-world" leaving you to figure everything out for yourself, but *This War of Mine* isn't quite like that, as everything is very simple to do.

The one thing it does share with other survival games is that it drops you right in the thick of it, as you could even start with characters who are already injured and depressed. A key difference is that you have multiple characters to look after.

You start off at your safe house. We use the word 'safe' rather lightly as it's a pretty ruined house that needs some major DIY. You will need to sort through the rubble to find supplies, and craft luxury items like chairs, beds and a radio to keep your little family of survivors happy, which can feel like a real chore at times.

Once night falls, the game becomes completely different. You will need to constantly find supplies like food and weapons, so this is

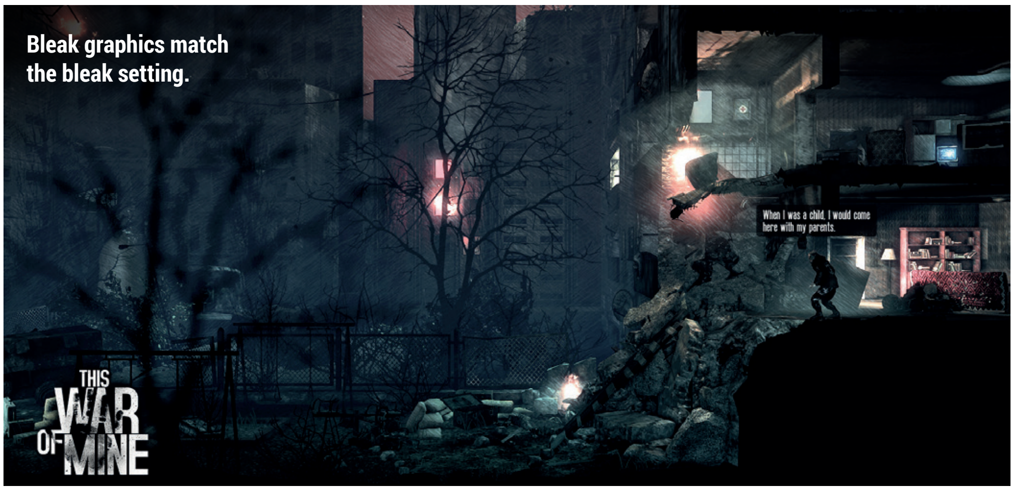
the time to accomplish these essential tasks.

You're given a choice of locations to raid, from completely abandoned houses to burnt ruins full of scared citizens who can attack you.

This is when it gets most fun, as you can choose to sneak around using hiding places to pop out and have a rummage, or you can go in guns blazing and murder those poor people for whatever is in their pockets – but if you decide to murder someone your characters' mental state really will pay for it.

Overall we found this to be one of the most interesting games to come out recently.

Store http://store.steampowered.com/app/282070/
Price £14.99



Bleak graphics match the bleak setting.

"You've played survival games before, sure, but you've never played anything quite as harsh as this..."

Civilization: Beyond Earth

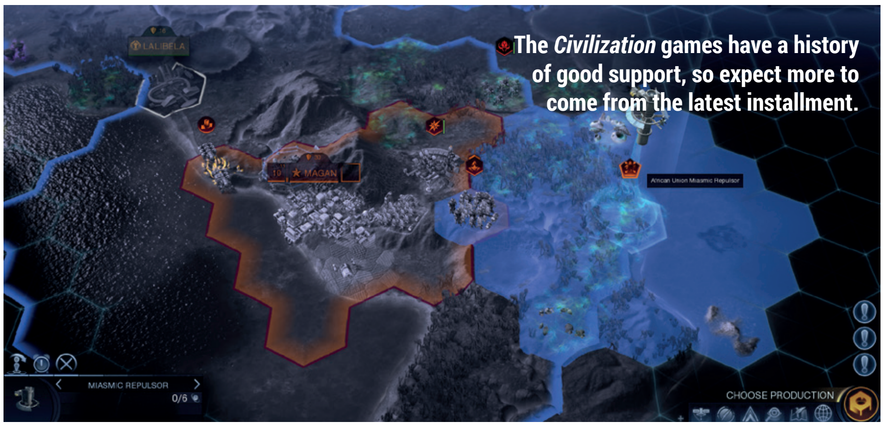
Can you survive this alien world?

The latest generation of *Civilization* has arrived for Linux, courtesy of porting house Aspyr Media, which ported the previous *Civilization* and the *Borderlands* games to Linux.

In *Beyond Earth*, instead of conquering a map on planet earth; you're on a completely alien world full of wondrous mysteries and strange creatures. That makes it similar in certain ways to the old *Alpha Centuri* games (they really need to do a new one!) for the setting, but game-play wise it's very similar to the previous *Civilization* game. In fact a fair amount of people have called it a "re-skin" of *Civilization V*, and it's hard to disagree as most of the mechanics are seemingly the same with different names, but that doesn't stop it being fun at all, as even for seasoned players it will be refreshing.

The setting alone is far more exciting for anyone who is a fan of space science fiction, and it is executed extremely well with the visual style and sound effects. So, if like us space cadets, that's your thing, then you will probably enjoy this version a lot more.

Store http://store.steampowered.com/app/65980 **Price** £29.99



The *Civilization* games have a history of good support, so expect more to come from the latest installment.

SuperTuxKart

Open source racing just got awesome.

This is probably the biggest ever update to our favourite open source racer since it began. This huge update brings with it an overhaul to the "Antartica" game engine, and it's a good 'un.

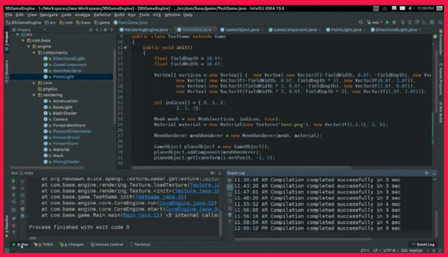
The new engine brings massive improvements to the lighting, physics, and shadows in the game, making everything look just that bit better, but you need decent looking maps to go with it, so they re-did the maps too, and boy are they gorgeous. The old graphics and engine did not age well against other racers, so we hope this helps to breathe new life into a fantastic project.

You don't have to take our word for it; as it's open source you can give it a go any time. Like most open source games it needs contributors, so get in touch with the developers if you fancy helping out.

One thing Linux still lacks is a decent racing game, so could this fill the void? It certainly would be fantastic for a younger audience to muck about on. Sadly, it was released without an easily downloadable file to run on Linux, so until they arrange that you will need to compile it from the source, but they have instructions on their site to follow to enable you to do this.

Website http://supertuxkart.sourceforge.net/ **Price** Free

ALSO RELEASED...



```
public class TestGame extends Game
{
    public void init()
    {
        float fieldDepth = 10.0f;
        float fieldWidth = 10.0f;

        Vertex[] vertices = new Vertex[] {  new Vertex( new Vector3f(-fieldWidth, 0.0f, -fieldDepth), new Vec
                new Vertex( new Vector3f(-fieldWidth, 0.0f, fieldDepth * 3), new Vector2f(0.0f, 1.0f)),
                new Vertex( new Vector3f(fieldWidth * 3, 0.0f, -fieldDepth), new Vector2f(1.0f, 0.0f)),
                new Vertex( new Vector3f(fieldWidth * 3, 0.0f, fieldDepth * 3), new Vector2f(1.0f, 1.0f))};

        int indices[] = { 0, 1, 2,
                2, 1, 3};

        Mesh mesh = new Mesh(vertices, indices, true);
        Material material = new Material(new Texture("test.png"), new Vector3f(1,1,1), 1, 8);

        MeshRenderer meshRenderer = new MeshRenderer(mesh, material);

        GameObject planeObject = new GameObject();
        planeObject.addComponent(meshRenderer);
        planeObject.getTransform().setPos(0, -1, 5);
```

```
    at org.newdawn.slick.opengl.TextureLoader.getTexture(Textu
    at com.base.engine.rendering.Texture.loadTexture(Texture.ja
    at com.base.engine.rendering.Texture.<init>(Texture.java:1
    at com.base.game.TestGame.init(TestGame.java:25)
    at com.base.engine.core.CoreEngine.run(CoreEngine.java:52)
    at com.base.engine.core.CoreEngine.start(CoreEngine.java:3
    at com.base.game.Main.main(Main.java:11) <5 internal calls

Process finished with exit code 0
```

```
11:30:48 AM Compilation completed successfully in 3 sec
11:43:20 AM Compilation completed successfully in 3 sec
11:47:01 AM Compilation completed successfully in 3 sec
11:48:20 AM Compilation completed successfully in 3 sec
11:55:52 AM Compilation completed successfully in 4 sec
11:56:05 AM Compilation completed successfully in 3 sec
11:56:16 AM Compilation completed successfully in 3 sec
11:58:54 AM Compilation completed successfully in 4 sec
12:00:10 PM Compilation completed successfully in 3 sec
```

```
27:1  LF  UTF-8  Git: master
```

Godot Engine

The open source game engine and toolkit *Godot Engine* has release its big "1.0" release promising lots of polish and new features.

It looks like it will be a serious contender for anyone wanting to do proper game development on Linux directly, and that's fantastic. It has an impressive feature list, so it should satisfy more than your basic needs.
www.godotengine.org/wp

The Original Strife: Veteran Edition

Who fancies playing a really old school blend of RPG and FPS mechanics wrapped up in a neat open source engine? You actually get a copy of the source with each purchase! The game originally came out in 1996, so a lot of the mechanics still feel dated even with the new engine, but it's still a good bit of nostalgia.
http://store.steampowered.com/app/317040

Reassembly

This is the game that has kept a certain Games Editor glued to his computer, as it is a little addictive. It has some fantastic vector graphics, and an epic soundtrack, so it's not just a treat for the eyes.

It's a space sandbox game where you build you own ships, and the ship editor is so easy it's laughable (in a nice way!).
http://store.steampowered.com/app/329130

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

FIND OF THE FORTNIGHT

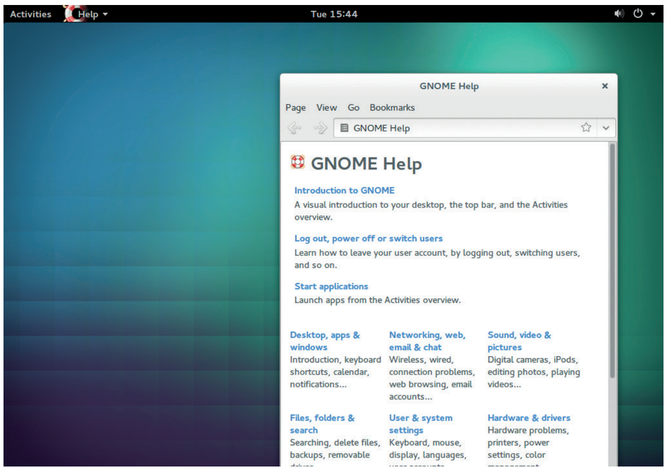
Thanks for the pointer to Tanglu [DistroHopper, LV008]. Even from the live DVD the KDE version was more responsive than I would expect on my Gateway LT4004u with Intel Atom N2600 CPU and 3600 Graphics Media Accelerator (GMA). It just booted and got down to business without needing any intervention from me. Other flavours of OS have been troublesome with this combination of CPU, GMA and Broadcom network interface.

In the past, I have found KDE to be ponderously baroque despite being well-developed. The newer version of KDE delivered with Tanglu looks good on the 10.1–inch LED 1024x600 display.

Installation to my netbook completed in about 30 minutes. Tanglu detected the netbook platform then configured the desktop for the KDE 4 netbook GUI.

System Monitor showed four threads consuming less than 20% of CPU capacity, indicating that Tanglu installed the Intel drivers for the GMA: SolydK failed in this area, resorted to software rendering of the graphics, producing a single-thread CPU load of around 70% capacity – SolydK took over two hours to install too

Andrew Shead



GNOME Help

Introduction to GNOME
A visual introduction to your desktop, the top bar, and the Activities overview.

Log out, power off or switch users
Learn how to leave your user account, by logging out, switching users, and so on.

Start applications
Launch apps from the Activities overview.

Desktop, apps & windows
Introduction, keyboard shortcuts, calendar, notifications...

Networking, web, email & chat
Wireless, wired, connection problems, web browsing, email accounts...

Sound, video & pictures
Digital cameras, iPods, editing photos, playing videos...

Files, folders & search
Searching, delete files, backups, removable

User & system settings
Keyboard, mouse, display, languages,

Hardware & drivers
Hardware problems, printers, power settings, color

Debian, but with with graphical polish and a regular release cycle... sounds familiar... sounds like Ubuntu!

Ben says: I'm glad you like it! We see so many Debian/Ubuntu respins with minimal changes made that it's refreshing to see a distro trying something authentically new.

WE FAIL

A couple of points. Your Arduino tutorial [in LV011] contained an error. The Arduino does not run the Processing language, although it's a commonly held misconception that it does. In fact "sketches" are written in C or C++ as is evident if you examine the code in the article. Processing is a language that runs on another computer and communicates using a serial protocol with an Arduino running a standard firmware "sketch" called Firmata. The following link makes this clear.

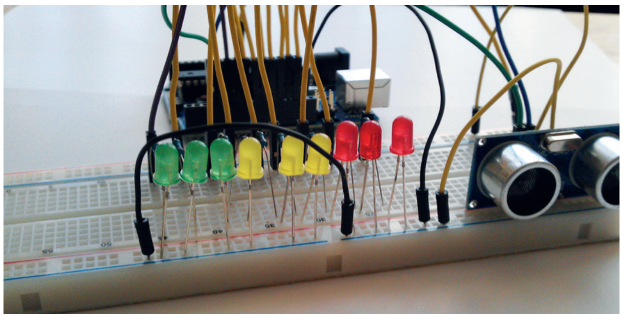
http://playground.arduino.cc/Interfacing/Processing

The use of Processing is entirely optional: for standalone projects where the Arduino is to run autonomously there is no choice other than to use a sketch written in C/++.

Secondly I notice you regularly advocate *DOSBox* rather than *DOSEMU* for running DOS programs. The last time I looked at this – admittedly some years ago – it was evident that *DOSBox* was designed for games, reducing the effective CPU speed accordingly. *DOSEMU* runs DOS programs fast and is a program I use daily. Of course it's possible that *DOSBox* now offers this capability, but it didn't when I tried it. It struck me as a case of horses for courses.

It's a great magazine and I've just renewed my subscription!

Peter Hinch



Wire stuff up, then write code to make cool things happen – that's the simple genius behind the Arduino.

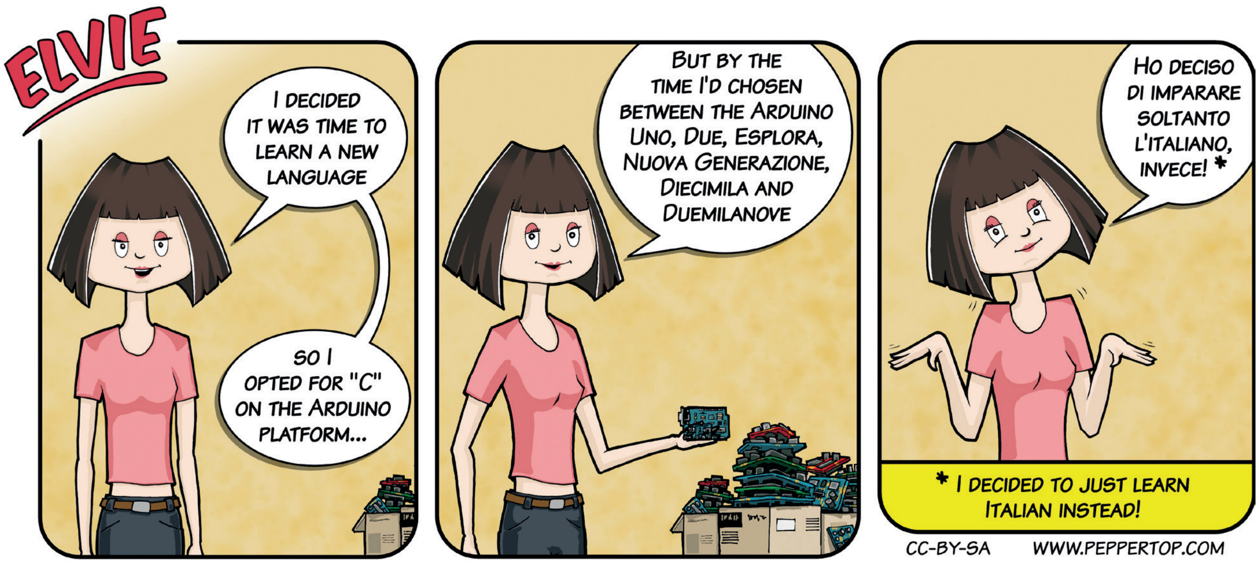
DRINKING

Congratulations on hitting the one year mark! I have gladly renewed my subscription. Any thoughts on allowing subscribers to set up an auto-renewing subscription? I have that set up with your former publication, and it is a very nice convenience.

And, great mugs! Yes, you all do look great in your photographs, but that's not what I'm talking about. I'm talking about the coffee/tea mug (do you even drink tea from a mug?). After having my letter featured in LV002, I had to have one. It looks better in person than it does on your website. I was a bit apprehensive about having a piece of ceramic shipped from overseas, but your shipping department did a fantastic job on the packaging. The form fitting styrofoam container took the trip over the pond in grand style, showing a bit of wear and tear, but protecting its precious cargo without fail. The ability to publish an outstanding magazine AND ship mugs unbroken internationally? What other skills are you hiding?

Paul Olson, Oklahoma, USA

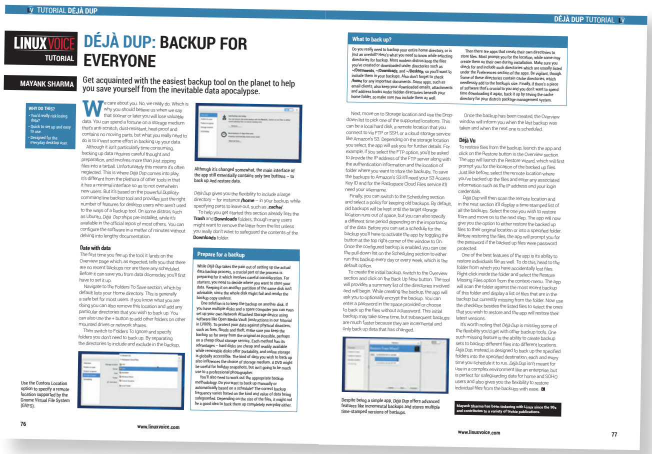
Andrew says: Thanks! We're all amazed by how quickly the year has gone by. It's funny you should mention this: we do have Direct Debit payments enabled for renewals now, and by the time you read this we should have got it running for new subscriptions as well. At first we could only take payments via PayPal, but we're incrementally improving things (incrementally because if anything went wrong it would be up to Ben to fix it, so we're only adding options one at a time). On the mug question: I drink tea from a 50-year-old pint mug that Mike brought me back from Germany. Hidden talents: Graham plays the piano. Mike can juggle. Ben was Worcestershire under-10s freestyle wrestling champion. Americans! Thanks to Graham's packing skills, your mug will reach the shores of liberty whole and unbroken – **http://shop.linuxvoice.com/product/linux-voice-mug**.


ELVIE
I DECIDED IT WAS TIME TO LEARN A NEW LANGUAGE
SO I OPTED FOR "C" ON THE ARDUINO PLATFORM...
BUT BY THE TIME I'D CHOSEN BETWEEN THE ARDUINO UNO, DUE, ESPLORA, NUOVA GENERAZIONE, DIECIMILA AND DUEMILANOVE
HO DECISO DI IMPARARE SOLTANTO L'ITALIANO, INVECE! *
* I DECIDED TO JUST LEARN ITALIAN INSTEAD!
CC-BY-SA
WWW.PEPPERTOP.COM

DISTRO HOPPER

As you might expect when one distro hops, backing up one's data is important. But I had an issue when using *Déjà Dup* [featured in our tutorial section, LV011], after I tried to restore my data. Whilst I can't fully remember the sequence of events now, I think my problem of NOT being able to restore my data was linked to either a change in my computer's host name or a change in which DNS service I used. Anyway, a host name change was my problem and since *Déjà Dup* didn't allow me to specify the host name or even indicate which host name it was using I decided to write my own *Bash* script and using *rsync* do my own backup! It may be that my own script isn't the most efficient at backing up but I know I can restore my stuff. I sometimes wonder when apps are recommended whether the person recommending them have actually used them enough to hit such problems.
G White

Andrew says: I'm pretty sure Mayank, who wrote the *Déjà Dup* tutorial last issue, has been using it for a while now, and his two-page tutorial was more than enough to get new users introduced to it (rather than being an exhaustive look at corner cases – but in this case your best bet would be your distro's forums). On a more profound note, I take my hat off to you for writing your own script! Hackability – that's the key. When the existing solutions don't work for whatever reason, free software gives us the ability to make our own – that's why we like it so much.


LV TUTORIAL DÉJÀ DUP

DÉJÀ DUP TUTORIAL LV

LINUXVOICE TUTORIAL

MAYANK SHARMA

DÉJÀ DUP: BACKUP FOR EVERYONE

Get acquainted with the easiest backup tool on the planet to help you save yourself from the inevitable data apocalypse.

WHY DO THIS?
- You'd really risk losing data?
- Quick to set up and easy to use.
- Designed for the everyday desktop user.

We care about you. No, we really do. Which is why you should believe us when we say that sooner or later you will lose valuable data. You can spend a fortune on a storage medium that's anti-scratch, dust-resistant, heat-proof and contains no moving parts, but what you really need to do is to invest some effort in backing up your data.

Although it isn't particularly time consuming, backing up data requires careful thought and preparation, and involves more than just zipping files into a tarball. Unfortunately this means it's often neglected. This is where *Déjà Dup* comes into play. It's different from the plethora of other tools in that it has a minimal interface so as to not overwhelm new users. But it's based on the powerful *Duplicity* command line backup tool and provides just the right number of features for desktop users who aren't used to the ways of a backup tool. On some distros, such as Ubuntu, *Déjà Dup* ships pre-installed, while it's available in the official repos of most others. You can configure the software in a matter of minutes without delving into lengthy documentation.

Date with data

The first time you fire up the tool, it lands on the Overview page which, as expected, tells you that there are no recent backups nor are there any scheduled. Before it can save you from data doomsday, you'll first have to set it up.

Navigate to the Folders To Save section, which by default lists your Home directory. This is generally a safe bet for most users. If you know what you are doing you can also remove this location and add any particular directories that you wish to back up. You can also use the + button to add other folders on other mounted drives or network shares.

Then switch to Folders To Ignore and specify folders you don't need to back up. By separating the directories to include and exclude in the backup,

Use the Custom Location option to specify a remote location supported by the Gnome Virtual File System (GVFS).

Although it's changed somewhat, the main interface of the app still essentially contains only two buttons – to back up and restore data.

Déjà Dup gives you the flexibility to include a large directory – for instance **/home** – in your backup, while specifying parts to leave out, such as **.cache/**.

To help you get started this section already lists the **Trash** and **Downloads** folders, though many users might want to remove the latter from the list unless you really don't want to safeguard the contents of the **Downloads** folder.

Prepare for a backup

While *Déjà Dup* takes the pain out of setting up the actual data backup process, a crucial part of the process is preparing for it which involves careful consideration. For starters, you need to decide where you want to store your data. Keeping it on another partition of the same disk isn't advisable, since the whole disk might fail and render the backup copy useless.

One solution is to keep the backup on another disk. If you have multiple disks and a spare computer you can even set up your own Network Attached Storage device using software like Open Media Vault (instructions in our tutorial in LV009). To protect your data against physical disasters, such as fires, floods and theft, make sure you keep the backup as far away from the original as possible, perhaps on a cheap cloud storage service. Each method has its advantages – hard disks are cheap and readily available while removable disks offer portability, and online storage is globally accessible. The kind of data you wish to back up also influences the choice of storage medium. A DVD might be useful for holiday snapshots, but isn't going to be much use to a professional photographer.

You'll also need to work out the appropriate backup methodology. Do you want to back up manually or automatically based on a schedule? The correct backup frequency varies based on the kind and value of data being safeguarded. Depending on the size of the files, it might not be a good idea to back them up completely everyday either.

76 www.linuxvoice.com

What to back up?

Do you really need to backup your entire home directory, or is just an overkill? Here's what you need to know while selecting directories for backup. Most modern distros keep the files you've created or downloaded under directories such as **~/Documents**, **~/Downloads**, and **~/Desktop**, so you'll want to include them in your backups. Also don't forget to check **/home** for any important documents. Some apps, such as email clients, also keep your downloaded emails, attachments and address books under hidden directories beneath your home folder, so make sure you include them as well.

Then there are apps that create their own directories to store files. Most prompt you for the location, while some may create them on their own during installation. Make sure you check for and include such directories which are usually listed under the Preferences section of the apps. Be vigilant, though. Some of these directories contain cache directories, which needlessly add to the backup's size. Finally, if there's a piece of software that's crucial to you and you don't want to spend time downloading it again, back it up by saving the cache directory for your distro's package management system.

Next, move on to Storage location and use the Drop-down list to pick one of the supported locations. This can be a local hard disk, a remote location that you connect to via FTP or SSH, or a cloud storage service like Amazon's S3. Depending on the storage location you select, the app will ask you for further details. For example, if you select the FTP option, you'll be asked to provide the IP address of the FTP server along with the authentication information and the location of folder where you want to store the backups. To save the backups to Amazon's S3 it'll need your S3 Access Key ID and for the Rackspace Cloud Files service it'll need your username.

Finally, you can switch to the Scheduling section and select a policy for keeping old backups. By default, old backups will be kept until the target storage location runs out of space, but you can also specify a different time period depending on the importance of the data. Before you can set a schedule for the backup you'll have to activate the app by toggling the button at the top right-corner of the window to On. Once the configured backup is enabled, you can use the pull-down list on the Scheduling section to either run this backup every day or every week, which is the default option.

To create the initial backup, switch to the Overview section and click on the Back Up Now button. The tool will provides a summary list of the directories involved and will begin. While creating the backup, the app will ask you to optionally encrypt the backup. You can enter a password in the space provided or choose to back up the files without a password. This initial backup may take some time, but subsequent backups are much faster because they are incremental and only back up data that has changed.

Despite being a simple app, *Déjà Dup* offers advanced features like incremental backups and stores multiple time-stamped versions of backups.

Once the backup has been created, the Overview window will inform you when the last backup was taken and when the next one is scheduled.

Déjà Vu

To restore files from the backup, launch the app and click on the Restore button in the Overview section. The app will launch the Restore wizard, which will first prompt you for the location of the backed up files. Just like before, select the remote location where you've backed up the files and enter any associated information such as the IP address and your login credentials.

Déjà Dup will then scan the remote location and in the next section it'll display a time-stamped list of all the backups. Select the one you wish to restore from and move on to the next step. The app will now give you the option to either restore the backed up files to their original location or into a specified folder. Before restoring the files, the app will prompt you for the password if the backed up files were password protected.

One of the best features of the app is its ability to restore individuals file as well. To do this, head to the folder from which you have accidentally lost files. Right-click inside the folder and select the Restore Missing Files option from the context-menu. The app will scan the folder against the most recent backup of this folder and display a list of files that are in the backup but currently missing from the folder. Now use the checkbox besides the listed files to select the ones that you wish to restore and the app will restore their latest versions.

It's worth noting that *Déjà Dup* is missing some of the flexibility you'd get with other backup tools. One such missing feature is the ability to create backup sets to backup different files into different locations. *Déjà Dup*, instead, is designed to back up the specified folders into the specified destination, each and every time you schedule it to run. *Déjà Dup* isn't meant for use in a complex environment like an enterprise, but is perfect for safeguarding data for home and SOHO users and also gives you the flexibility to restore individual files from the backups with ease.

Mayank Sharma has been tinkering with Linux since the 90s and contributes to a variety of techie publications.

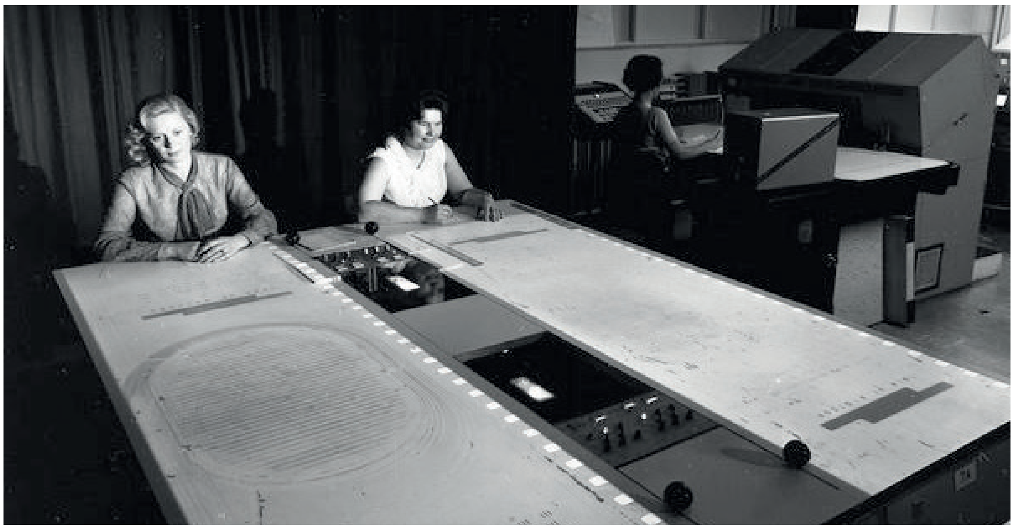
www.linuxvoice.com 77

Déjà Dup is an easy way to perform regular backups – if you aren't backing up already, try it out.

I'M HISTORY!

I have been picking up your magazine since I saw a copy of issue 3 on one of my irregular visits to WHS. I like the mix of articles and tutorials. With my advancing years I look at the coverage you give to older systems and there, on page 99 of issue 10 was the computer room at the Atlas Computer Lab where I worked showing Doug House, Tony Sargent and, I think, Ros Haliwell on the tape decks. I actually missed working on the Atlas as it shut down before I joined the lab in 1976 as a ICL shift engineer on the 1906a mainframe. The Atlas Processor was still there, but was being de-commissioned. An urban myth at the time was that on one occasion some circuit boards were removed, allegedly for an Atlas still working "Somewhere Underground".

My Atlas claim to fame is that in 1978 I was asked by the ICL engineer in charge, Ted Everson, to build a 12-volt power supply so that the Atlas console could be on display at the building Reception Desk with some lights on, so I was pleased to see in your article that the console has been "re-discovered and is back on display".


To have worked on a piece of the UK's computing history such as Atlas must be a great privilege.

I was also at the Lab, latterly renamed "Rutherford Atlas Lab" when a Cray 1 was shipped in, for temporary storage for the UK Meteorological Office at Bracknell. A very impressive piece of hardware, complete with Freon cooling system.

My wife has a history even older as she was a systems analyst working on LEO computers in the 1960s.

After 37 years as an engineer, I am now back at college studying HND computer science, not a sniff of Linux, all Microsoft, Adobe, Access, C#, Visual Basic and Oracle, just a tiny hint of *MySQL*, all this in a college five miles from the Sony factory making the Raspberry Pi. It makes you weep sometimes.
Gordon Ireson, South Wales

Graham Says: Wow! Thanks for sharing that with us! We're incredibly lucky that the history of our field is so accessible – even if you start with single-application analogue machines with valves rather than programmable digital computers, computer science is still young enough that most of us can comprehend the timeline without having to bend our heads too much. And don't worry about the HND – the world will catch up with Linux one day.

YOUR AD HERE

LV

Email andrew@linuxvoice.com to advertise here

LUGS ON TOUR

Pi Wars

Deciding the ultimate Raspberry Pi robot!

The Raspberry Pi's small size, low power requirements and accessible GPIOs make it a great device for robotics. It forms the brains of the robot, and all manner of hardware can be attached to the outside to create everything from sailing boats to flying machines. The vast majority of Raspberry Pi robots though move on solid ground, and it was these land-based Pi-powered robots that came together on a frosty morning at the start of December in Cambridge University's Institute for Astronomy to compete in nine challenges to find out which would be crowned champion.

The events were:

- Three point turn.
- Speed test.
- Obstacle course.
- Sumo.
- Line follower.
- Robot golf.
- Proximity alert.
- Code quality.

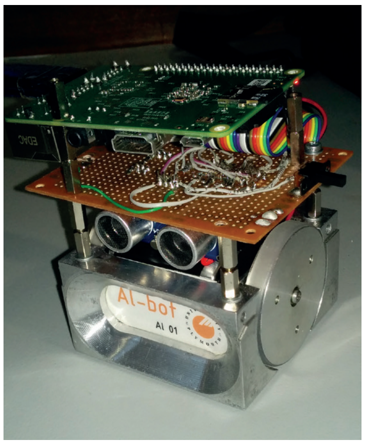
Together, these tested a range of qualities, and no one style of robot dominated all events. Big, powerful robots had the upper hand in sumo, but line-following and proximity alert (getting as close as possible to a wall without touching it) favoured smaller, more deft machines.

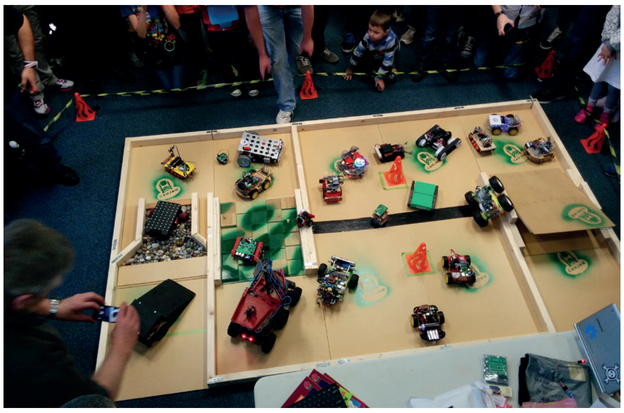
Entries ranged in size from the University of Plymouth's bot, which had the same footprint as a Raspberry Pi A+, to a behemoth decked out as a pirate ship. The only restriction in the rules was that the robots had to have a footprint smaller than a sheet of A3 paper.

The overall results were split into two categories: those that cost under £75 to make, and those that cost more. The competitors were split roughly evenly between those two categories. This kept the playing field even despite people entering from a wide variety of backgrounds. There were schools, university students, professional hardware designers and more present.

When Tim Richardson and Mike Horne (the Pi Wars organisers) totted up all the scores, TractorBot came out on top of the under £75 competition, and Psiclops won the over £75 category. Both of them walked away with a huge selection of Raspberry Pi goodies. There were also prizes for the winners in each event, and a few special prizes such as the most innovative and best-looking. In the end, almost every entrant left with something to show for their efforts.

If there's enough interest, Pi Wars may become a regular competition. Keep an eye on **www.piwars.org** for details of future events.





The range of events meant that every robot had a chance of doing well at Pi Wars. Good show everyone!

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

Literacy On Linux: The Global Learning XPRIZE

Jono Bacon launches a competition from his new position as Senior Director of Community at XPRIZE.

In September the $15 million Global Learning XPRIZE (**http://learning.xprize.org**) launched. The concept of the competition is simple at its core. Teams around the world are challenged to create an application that will run on an Android tablet that will teach a child basic reading, writing, and arithmetic, within 18 months, autonomously. For Linux and Open Source fans though, here is the neat part. The winner of the competition and the four runner-up teams will all be expected to release their code as open source, complete with all of the included assets, unique content and materials.

It doesn't end there. Teams are being asked to build their solutions using the hugely popular Android platform. This means that every solution will be built on an open source platform, using open APIs, and running an entirely open educational codebase. As you can imagine, the impact of this technology could be great.

According to the United Nations Educational, Scientific and Cultural Organisation (UNESCO), there are 250 million children around the world who cannot read. We simply cannot build enough schools or train enough teachers to serve our global literacy needs.

Make the world better

As such, the Global Learning XPRIZE is exploring how technology may be utilised to challenge some of the noted causes of illiteracy due to the lack of access to traditional models of education by creating a bridge to those thought unreachable. This technology could be deployed around the world, bringing learning experiences to children who do not have access to quality education, and supplementing the learning experiences of children who do.

Anyone is welcome to form a team and compete. This is a competition that is open to all, irrespective of approach. As you can imagine, this is a complex challenge that teams are being asked to solve. The competition has been structured with extensive development and field-testing phases. The overall competition will conclude in 2019.

How to enter

Team registration is currently open and closes on 31 March, 2015, then the solution development phase begins on 1 April, 2015 and will run until October 2016. These solutions will be judged and five finalists will be chosen, each of which will be awarded $1 million. The finalists will then enter the field-testing phase of the competition where their solutions will be tested with real kids in rural villages from June 2017 until the end of 2018. Finally, a winner will be picked and awarded the $10 million prize purse in early 2019. For more information or to enter simply visit: **http://learning.xprize.org**.

If you're interested in competing, the first step is to form a team. We recommend that you put together a team that comprises developers, testers, designers, artists, scientists, content developers and others that will help build a comprehensive solution. If you're looking for team members, it might be useful to reach out to your local Linux User Groups, educational groups, universities, visit the forum (**http://forum.xprize.org**) and more.

Next you should take a look at the prize guidelines and rules at **http://learning.xprize.org/about/guidelines** – this will answer many of the questions you may have about the specifics of participating. Finally, go and fill in an Intent To Compete form at **http://learning.xprize.org/about/registration**. This notifies the XPRIZE Foundation of your desire to participate and keeps you up to date with the progress of the prize. You will be notified if your application to participate has been successful.

"This technology could bring learning to children who do not have access to education."

We hope to see some Linux Voice readers creating teams and striving to not just win the $10 million, but to put their stamp in truly making this rock we live on a nicer place for everyone.

Global Learning Homepage – **http://learning.xprize.org**

Put a team together and help create an application that will autonomously teach kids literacy.

UNPROTECTED COMMUNICATION

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

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

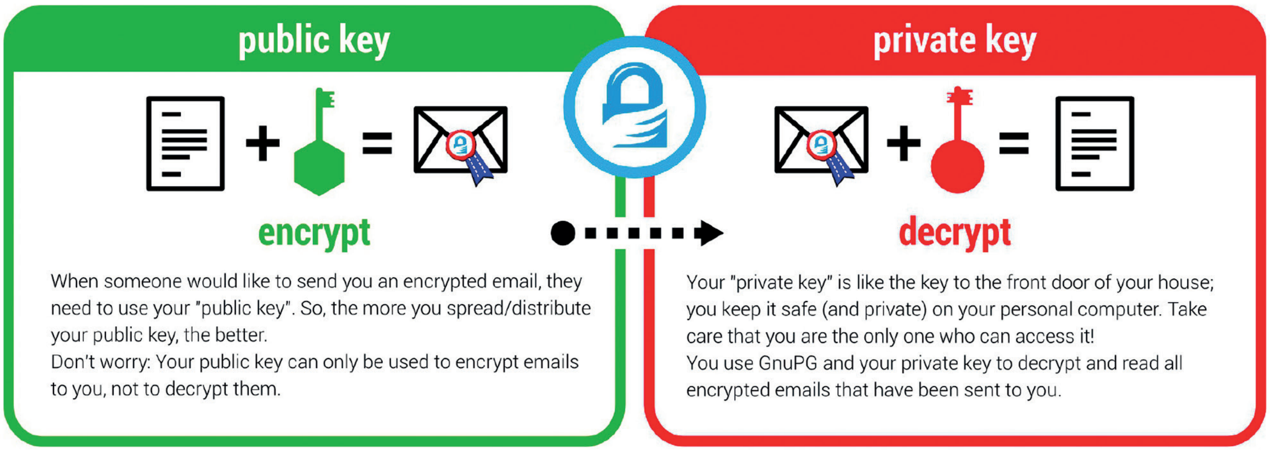
GnuPG is platform independent. That means it works with every email address and runs on pretty much any computer or recent mobile phone. GnuPG is free and available at no charge.

About GnuPG

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

What makes GnuPG secure?

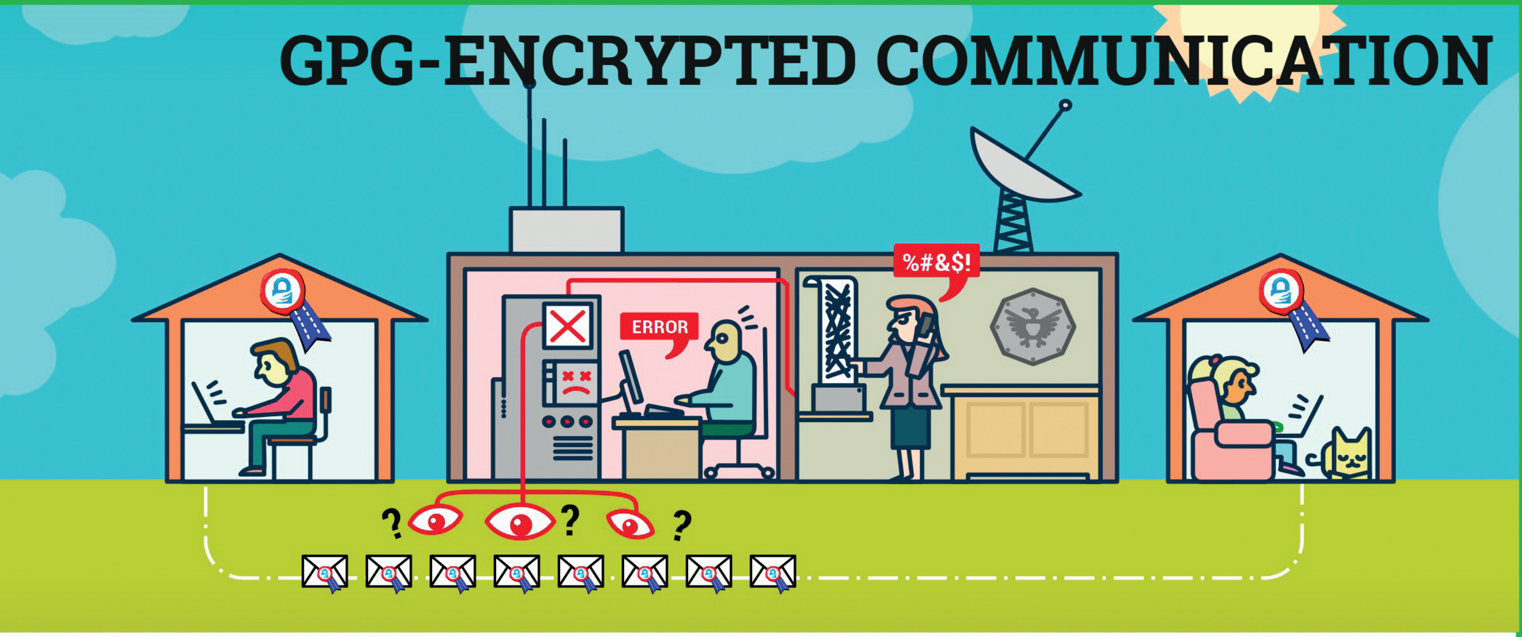
GnuPG is Free Software and uses Open Standards. That is essential to be sure that software can really protect us from surveillance. Because in proprietary software and formats, things might happen beyond your control.
If no one is allowed to see the source code of a program, nobody can be sure that it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



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

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

GPG-ENCRYPTED COMMUNICATION



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

Practical advice

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

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

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

About the fsfe

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

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

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

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

What is Free Software?

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

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

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

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

fellowship of fsfe

2015 THE YEAR OF LINUX

This year is set to be the best ever for Linux – and that's not just our opinion. Some of the biggest players in Free Software think so too…

There's a long-running joke in the Linux world, that the next year will be the year of "Linux on the desktop". At first it was a serious proposition: back in the early 2000s, it looked like the operating system was poised to take a significant chunk of desktop marketshare. Microsoft was struggling with major security problems in Windows, Apple's Mac OS X had barely gotten off the ground, and there was clearly room for something better.

So why didn't Linux dive in and win millions of users? Well, it did, just in another way. Microsoft started to take security more seriously, and OS X emerged as a shiny Unix-like OS that appealed to many geeks. Linux was still the best all-round operating system, we reckon, but the desktop PC market was no longer so appealing. As the end of the decade came closer, users were flocking towards mobile devices such as smartphones and tablets – a market where Linux dominates thanks to Android – and the server space became even bigger thanks to "cloud" computing, software/platform/infrastructure as a service, and the growth of web apps.

So Linux is, today, by far the most prevalent and important operating system in the world. It's a long way from being on every home desktop PC, but the next 12 months have plenty in store, and so for our first issue of the year we want to look ahead at the goodies to come. But we don't just want to wax lyrical about our favourite projects here; we also want to hear from some of the biggest names in Linux and Free Software about what they're most looking forward to.

But before we hear from the horses' mouths, so to speak, let's have a quick look back at 2014, a year crammed with big developments for our favourite operating system…

Highlights of 2014

There were plenty of ups and downs last year.

Last year got off to a corking start, with videogames giant Valve announcing 14 new models of Steam Machines – pre-built computers running the Linux-powered SteamOS platform. For much of its life, Linux hasn't been taken seriously as a gaming OS, but Valve's announcements changed this perception enormously. We're still a while away from seeing Steam Machines in every shop, but the future is looking bright for triple-A titles on Linux.

Meanwhile, Red Hat, one of the big players in Linux for businesses, snapped up the free CentOS distro. For years, CentOS had been providing a free and community-supported rebuild of Red Hat's Enterprise Linux product, and Red Hat had tolerated its existence without saying much about it. But it became clear that CentOS can benefit the company: sure, CentOS users aren't paying any money directly to Red Hat's coffers, but they could upgrade to the paid enterprise product at a certain point. From Red Hat's perspective, it's better if people are using CentOS rather than a completely different distro such as Debian.

In early February, the Debian project had a vote on which init system it should use in the next release of the distro. And guess what? Systemd won, to the surprise of many. Debian is a conservative distro, so the decision to replace a large chunk of the base system caused arguments that extended well into the rest of the year.

March was a great month. The Linux Foundation decided to offer its "Introduction to Linux" online course, which previously costed a whopping $2,400, for free. Another big development was Broadcom's release of the video driver source code for the VideoCore IV GPU, as used in the Raspberry Pi. April had some downers though: Heartbleed, the internet-breaking vulnerability in the widely used OpenSSL library, led some to question whether open source is really more secure. We'd say it is, of course, but just because something is open source, it doesn't mean that lots of eyeballs are going through the code. If only all those big companies affected by Heartbleed had supported OpenSSL development, either with money or patches, maybe things would have been different. In the same month, Canonical cancelled its Ubuntu One cloud storage service.

"The Heartbleed vulnerability led some to question whether open source really is more secure."

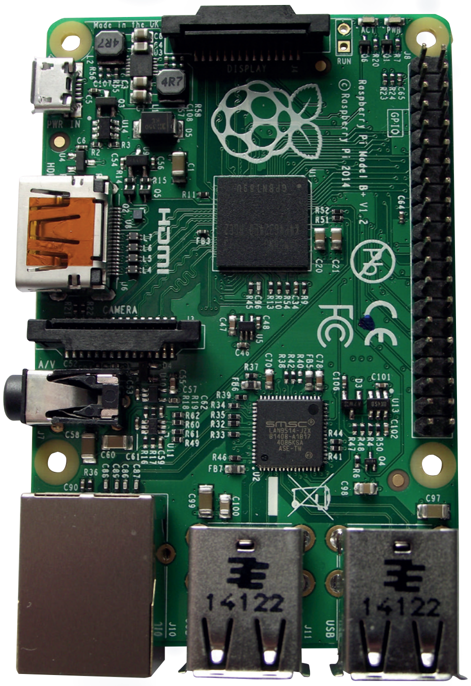
Still, in May the ever-diligent OpenBSD team announced LibreSSL, a fork of the OpenSSL codebase, and immediately started ripping out some of the horrendously old and complicated code chunks. Meanwhile, the Chinese government decided to ban the usage of Windows 8, while the Razor-Qt and LXDE desktops merged to form LXQt.

The Heartbleed OpenSSL vulnerability was such a big deal, it even got its own logo thanks to security company Codenomicon.

Game on

June was a good month for gamers: over 500 games, fully supported on Linux, were now available on Steam. *Krita*, KDE's peerless drawing application, announced a new crowdfunding project to implement new features, and as July came in, we were all greeted by a new Raspberry Pi model, the B+. KDE 5 was released and it was announced that 97% of the world's supercomputers run Linux. Not bad at all!

And it was a good time for open standards: governments, schools and other institutions in Valencia, Geneva and Toulouse announced that they were no longer paying the Microsoft tax, and had moved to Free Software. The UK government surprised us all by demanding that public documents should be in open formats such as ODF.


The Raspberry Pi Model B+ was a very welcome upgrade, with more USB ports, more GPIO pins, and better power management.

In September, the NHS dumped Oracle in favour of an open source solution, while Netflix users could now watch films on Linux (via Chrome and Ubuntu).

But another downer came in October: the Shellshock bug, which affected a *Bash* vulnerability that had been in the code since 1989. It was another sobering reminder that FOSS only works when people are studying the code. Adobe also announced that it was dropping Reader support for Linux, but given the program's bloat and unreliability, few people really cared.

November and December wrapped up the year with some announcements: Mozilla ended its relationship with Google, and decided to move to Yahoo as its default search engine. The Jolla tablet was crowdfunded, Debian was forked into the non-Systemd-requiring Devuan distro, and HP launched an EliteBook laptop with SUSE Enterprise Linux pre-installed.

What the big names say

We asked prominent developers and project leaders in the FOSS world what they thought of 2014, and what they're most looking forward to this year.

Boudewijn Rempt

Lead developer of Krita, a powerful graphics editor for KDE.

What was your biggest highlight of 2014?

BR: Oh gosh! We had so many – we released *Krita* in Valve's Steam app store, we did a really successful Kickstarter for the Krita Foundation, and we had a booth at Siggraph! [computer graphics conference] And then we got a five-star review in an artists' magazine, *ImagineFX*, so *Krita* is getting noticed outside the FOSS world too.

What are you most looking forward to in 2015?

BR: The next release of *Krita*, with all the Kickstarter features... our next Kickstarter, full support for creating animations, and the port to the Qt 5 graphical toolkit. No, I'm lying, I'm not looking forward to that port, but by March it'll be done.

What are the biggest challenges that face GNU/Linux and FOSS in the upcoming year?

BR: GNU/Linux will probably be fine: the kernel is well-funded, has a broad contributor base and, yeah, it'll do fine. FOSS in general too, apart from the usual patents challenge. I think the big challenge is all the divisiveness. All the hate people feel when someone comes up with a new idea. It's taking the fun out of working on FOSS! If only everyone who feels the need to write a hateful mail or post would pledge to also write one love letter to a project of choice, maybe it'd balance out?

Ben Nuttall

Education, development and outreach for the Raspberry Pi Foundation.

What was your biggest highlight of 2014?

BN: Personally speaking, I'd say attending PyConUK. With the Raspberry Pi Foundation running the education track, introducing kids and teachers to open computing; Carrie Anne's keynote thanking the community for its efforts; and the wealth of presentations and lightning talks all rounding up to demonstrate what a great time this is to be involved in making a change through open source in education and industry.

What are you most looking forward to in 2015?

BN: FOSS is more than just a warm tingle, or knowing you're sticking it to The Man, it's a viable alternative. We've seen such growth in activity and community engagement in the FOSS community over the last few years, with more people using Linux in different flavours, and open source software libraries being ever more prevalent. We're seeing fantastic examples of what can be achieved in the free & open source world, and in many areas what we have is pushing the limits of the proprietary alternatives, we're at a tipping point where I see FOSS being the preferable option in more and more cases. That's exciting, and I'm looking forward to what comes in 2015.

What are the biggest challenges that face GNU/Linux and FOSS in the upcoming year?

BN: We still suffer from a lack of choice for consumers. In the high street, people have the choice between buying a Windows laptop and a Macbook, there's just no awareness of an alternative. The addition of Apple is a huge step forward in diversity – now people know there isn't just one type of computer with one interface they should expect everywhere. However in the mobile phone market, when smartphones arrived, Linux (though Android) ended up taking the majority of market share without anyone even noticing.

Proprietary software vendors know they're losing out to FOSS, especially in areas like web servers where Linux is king. They know they're slipping behind and they're worried. What they do to combat that could be good for everyone, if they become more open or start using open standards, or it could be competitive. In recent months we've seen Microsoft open source the .NET core and welcome contributions on GitHub, and we've seen them make Docker available for Windows. A concern could be that with open source tools available, it takes an edge off the attraction for some to use Linux.

Matthew Miller

Works for Red Hat as the Fedora Project leader.

What was your highlight of 2014?
MM: The Fedora 21 release. It was a full year since Fedora 20, and it was my first release as Project Leader, so I was quite anxious. But so many people put in amazing, great work and it all came together into one of the best Linux distro releases ever.

What are the biggest challenges that face GNU/Linux and FOSS in the upcoming year?
MM: Without a doubt, the anti-freedom agenda of media companies. Patent-encumbered software and DRM in HTML standards sets us on a direction that will be very hard to recover from, and all of the real money is on the wrong side in this fight.

What are you most looking forward to in 2015?
MM: Fedora 22 and Fedora 23, of course!

Philip Newborough

Lead developer of CrunchBang Linux.

What was your highlight of 2014?
PN: Gnome 3.14. I've been watching Gnome 3 with a keen interest and I've tried each new release. I found the early releases unusable, but the latest releases have shown real promise. When 3.14 was released in September, I adopted it as my daily desktop. I'm loving how the Gnome developers make the desktop so simple to use, it's minimal, yet still super-powerful.

What are you most looking forward to in 2015?
PN: There are a couple of things that come to mind. Debian Jessie: I'm running it on a couple of machines and it looks like being the best Debian release yet. Windows 10: It's about time that Microsoft brought virtual desktops to the masses.

What are the biggest challenges facing GNU/Linux and FOSS?
PN: Civility. I think the community really struggled in 2014 and at times, I felt embarrassed to be a part of it. Debate is important, but there is no excuse for threatening behaviour and/or abuse. Therefore, I think the biggest challenge will be to keep the debates open, but keep them civil. If things continue as they have in 2014, we'll lose talented developers and we'll stop attracting new ones. That would suck.

Italo Vignoli

Handles media relations at The Document Foundation, makers of LibreOffice.

What was your biggest highlight of 2014?
IV: I can list a few: major release of LibreOffice 4.2, new board of directors in place (with second generation TDF members), UK Government decision to standardise on ODF, major release of *LibreOffice 4.3*, and the Bern LibreOffice Conference.

What are you most looking forward to in 2015?
IV: We are looking forward to other major releases of *LibreOffice*, to the availability of the software on Android, to other significant projects of companies and public administrations migrating to *LibreOffice*, and to a further growth of the ecosystem.

We are also looking forward to another great LibreOffice Conference in the city of Aarhus in Denmark, and to a large number of events in the other geographies.

What are the biggest challenges that face GNU/Linux and FOSS in the upcoming year?
IV: Getting recognised for its real value, not only among people with a decent technical background but also among desktop users.

It's a pity to look at users who have Windows-related problems being stuck because they do not know about the possibility of switching to GNU/Linux or have been discouraged.

FOSS developers should improve their communication strategies and skills, to grow their penetration among basic PC users who are bombarded by proprietary software messages (and who are locked-in without even realising it).

Jos Poortvliet and Frank Karlitschek

FOSS advocates from the KDE and OwnCloud teams.

What was your highlight of 2014?
JP: In the KDE world, we moved to deliver Frameworks 5, which is already having a big impact on the Linux desktop with many projects porting to *Qt* and, we hope, ready to take advantage of KDE technology in the future. Of course, Plasma 5, and just before the year was over, the first KDE applications built on Frameworks 5 also saw the light. These will mature over 2015 and move to more platforms, with *GCompris* leading for Android.

I think the move to a single standard desktop/mobile toolkit for Linux is extremely important for its future as a viable platform.

Note that the toolkit isn't the desktop – Gnome is doing great as a desktop team, moving user experience forward and I applaud them for that! But for app development, it is good if there is one clear choice, with other more specialised options always around to keep some healthy competition and pressure.

What are you most looking forward to in 2015?
FK: *OwnCloud* is moving forward with distributed and federated cloud technology, tackling the very core issues around privacy and security on the web.

Self hosting and control are super important but they always face the challenge of being disconnected – and the power of the web is that it connects people. Many projects work on these issues of course but *OwnCloud* is going fast and big: this technology is being used to connect over 500,000 students from three separate German universities that each run (and control!) their own *OwnCloud* instances.

What are the biggest challenges that face GNU/Linux and FOSS in the upcoming year?
JP: The challenges are big: to put people back in charge of their digital life requires lowering the barrier to using open technologies. Hardware seems to play an ever-increasing role in this, with many projects working to release open hardware. There are significant hurdles to be taken, both for the wider maker movement as well as for projects like *OwnCloud* and KDE. We, the software and the hardware guys and girls, need to work together more, leverage each other, yet remain practical. I think the super extreme projects (like the Purism project and Librem 15 laptop) show the way, pave it even, but it is the more down to earth ones which change the world.

The barrier we fight is twofold: technical difficulties and network effects. Nothing new, of course. Frankly, the legal, political and commercial forces against 'all things open' don't scare me: look at where we are today compared to one or two decades ago! Open platforms are inherently stronger and, over time, seem to win out. There's no reason to assume this will change.

Lucas Nussbaum

Debian Project Leader, and Debian developer since 2005.

What was your highlight of 2014?
LN: During 2014, Debian made a series of hard decisions about how to deal with the transition from our historical init system (sysvinit) to a more modern one (Systemd). While this sounds fairly technical, those decisions required us to question our deep beliefs about what Debian is supposed to be, and how Debian is supposed to work. This resulted in some of the most difficult discussions and decisions in Debian's history.

What are you most looking forward to in 2015?
LN: The release for Debian Jessie, in Q1 2015! I am very confident that this release will reassure all our users that we have made the right decisions, and that Debian is stronger than ever.

What are the biggest challenges that face GNU/Linux and FOSS in the upcoming year?
LN: With the increasing move of our computing to cloud infrastructures, we give up the control of our computing to the managers of our those infrastructures. Our terminals (laptops, desktops) might now be running entirely on Free Software, but this is increasingly irrelevant given that most of what actually matters gets executed on a remote closed system that we don't control. The Free Software community needs to work to help users keep the control of all their computing, by developing suitable alternatives and facilitating their deployment.

Lennart Poettering

Lead developer of Systemd, and formerly of PulseAudio and Avahi fame.

What was your biggest highlight of 2014?
LP: Well, I figure the decision from Ubuntu to adopt Systemd, too, so quickly after Debian made its decision.

What are you most looking forward to in 2015?
LP: Hmm, that's hard to say. I just contributed my share to the Gnome *Builder* Indiegogo campaign, so I guess I am looking forward to the progress *Builder* will make over the year!

What are the biggest challenges that face GNU/Linux and FOSS in the upcoming year?
LP: I figure our biggest challenge is to make sure our way of open source becomes more relevant again, given the threat from "over-the-wall" open source done by Google and similar companies. These companies maintain projects where no real open source community exists, but only the most superficial things you have to do to call something "open source" are done, like they do for Android for example.

We also need to make sure that Linux with its classic userspace can compete with Android, and be an attractive alternative [to regulated app stores] where everybody participating is equal.

Simon Steinbeiß

Lead developer of Xubuntu, the Xfce-based Ubuntu spin-off.

What was your biggest highlight of 2014?
SS: My personal highlight of 2014 was definitely the release of Xubuntu 14.04 Long Term Support. I've been around in the Xubuntu project for several years and to me, this is the best release to date. I'm fully aware of the superlative and that this might sound corny coming from the project leader (although I wasn't leader for 14.04), but an incredible amount of effort that had been put into the Xubuntu project by its team members during previous cycles has come together in this release.

We also decided to take a leap of faith and ship several development versions of Xfce, as the 4.12 release date is still in the not-so-clear future. Some of our team members, including myself, have also helped the Xfce project upstream by submitting patches or maintaining abandoned parts of the desktop environment. While we're still a small team, we have several stable contributors and are able to get our stuff done very effectively. It has to be mentioned though that we'd love to grow and new contributors are more than welcome!

All in all, I would say 2014 was a terrific year for Xubuntu and the overwhelmingly positive feedback we received from the community for the 14.04 release gave us the energy to push forward in 2015.

What are you most looking forward to in 2015?
SS: I'm personally looking forward to the 4.12 release of Xfce, which is sort of overdue. I'm pretty confident it won't take too much longer and then starting the port to *GTK 3* is definitely one of the things I'm looking forward to as well, even if it'll probably mean a loss of stability in the beginning. The instability I see ahead is related to *GTK 3* being a moving target still – the toolkit is changing quite significantly with each release, breaking theming or deprecating function calls – and porting not being an easy task, especially with the larger components like *Thunar* (the file manager) or the panel.

In terms of Xubuntu, I'm looking forward to new contributors joining our ranks and helping us to make Xubuntu (and Xfce) even better.

What are the biggest challenges that face GNU/Linux and FOSS in the upcoming year?
SS: I'm not sure the challenges have changed too much in the recent years, at least from my perspective. There is an ongoing trend of fragmentation, flame wars etc that is very detrimental to the public reception of GNU/Linux or FOSS but also to the developer community itself. We've seen some very extreme controversies take place in 2014 (for instance Systemd) and I can only hope that this will not be among the biggest challenges of 2015.

Other challenges I see ahead are related to Android/Google and its dominance. As it is the dominant operating system in the mobile space, ChromeOS is also gaining traction, and the traditional Linux desktop seems to become less relevant as Google is using the same lock-in mechanism with its ecosystem that made Apple so big and ubiquitous. Obviously ChromeOS doesn't satisfy the daily needs of many (especially power) users, but the seamless integration of mobile systems in the desktop (notifications, SMS replies etc) is a point where Linux will hopefully catch up at some point in the near future.

To come in 2015

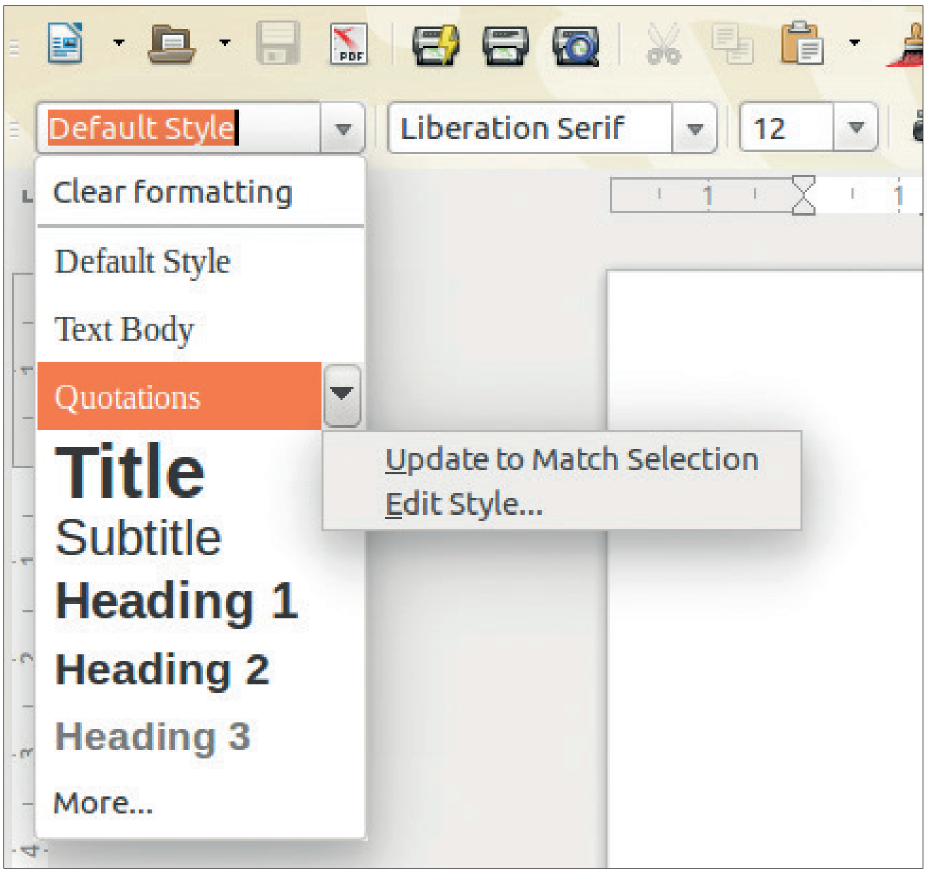
New distro releases, software updates, and much more…

Even if you don't use Ubuntu, a new release of that distribution is always a major event in the Linux world, given how many other distros are based on it. Ubuntu 15.04 (aka the Vivid Vervet) is due to arrive in late April and should include Systemd by default, making it the last of the major desktop-oriented distros to switch. This change hasn't been received well by everyone, but it brings Ubuntu – and its siblings Kubuntu, Xubuntu, Lubuntu and co – into line with OpenSUSE, Fedora, Arch, Debian and others.

Kubuntu users will have KDE Plasma 5 as the default desktop; it was also available in 14.10, but only as a technical preview. It doesn't look like Xfce 4.12 will arrive in time for the next Xubuntu release, nor will LXQt be ready for Lubuntu, so both of those distros will have relatively low-profile releases.

Fedora 22, meanwhile, is scheduled for late May, and will sport *DNF* as a new package manager, replacing *Yum*. It looks very unlikely that Wayland will replace X.org by default though. Debian 8 (codenamed Jessie) isn't pencilled in for any particular date, in true Debian fashion, but we can expect it before summer, all being well. New features include the aforementioned switch to Systemd, the inclusion of the MATE and Cinnamon desktops in the package repositories, and container support via the increasingly popular Docker.

The KDE team might squeeze a Plasma 5.2 into the end of January, featuring a new KWayland Server component, easy configuration of the SDDM display manager, and support for undoing Plasmoid actions (e.g. recovering deleted Plasmoids). Gnome 3.16, later in the year, will build on the superb work that's gone into version 3.14, most notably with the addition of a new application for reading and managing eBooks.



LibreOffice 4.4 will provide a quick way to edit styles, from within the styles menu itself.

So long, Microsoft Office...

Due for release at the end of January (but more likely to slip into February), *LibreOffice*

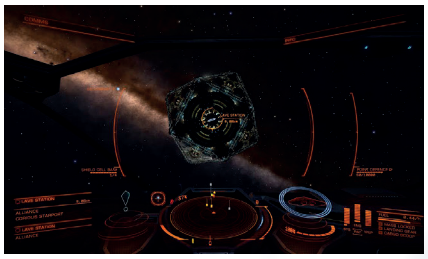
On our wishlist

There's so much that we, at Linux Voice HQ, would love to see this year. Imagine a Raspberry Pi Model C, for instance, with a better CPU and more RAM (512MB is pretty limiting for some jobs). Sure, we appreciate that the Raspberry Pi Foundation doesn't want to clutter up the market with many different models, so that some projects only work on some boards and not on others – and that the Pi has never been marketed as a replacement for a desktop computer. But the Pi makes for such an awesome, silent, low-power server and NAS device (when hooked up to a USB hard drive), and it could run many services in parallel with more RAM.

We'd also love to see a final consumer version of the Oculus Rift virtual reality headset. The current developer model (DK2) has received very positive feedback, so we'd be in heaven if it were made available to the masses before the end of the year. Providing, of course, that it had extensive Linux support, and then there's one game we're absolutely bursting to play on it, which would also need to be ported to Linux… *Elite Dangerous*!

On a more technical level, we'd really like distributions to work together more closely on a cross-distro packaging solution. Lennart Poettering talks a bit about this on page 42. Having different distros with different goals is great – that's software freedom. And having well-checked repositories is also important. But there needs to be a simpler way for third-part app developers to distribute their software than to package up for Ubuntu 14.10, and Fedora 21, and Fedora 22, and OpenSUSE X, and RHEL Y, and so forth. Sure, you can make cross-distro binaries by statically linking in every library, but that's wasteful.

Anyway, what's on your Linux and FOSS wishlist for this year? Are you waiting for a certain distro to be released? A new version of your favourite app? Or a long-standing bug to be fixed? Let us know: **letters@linuxvoice.com**.



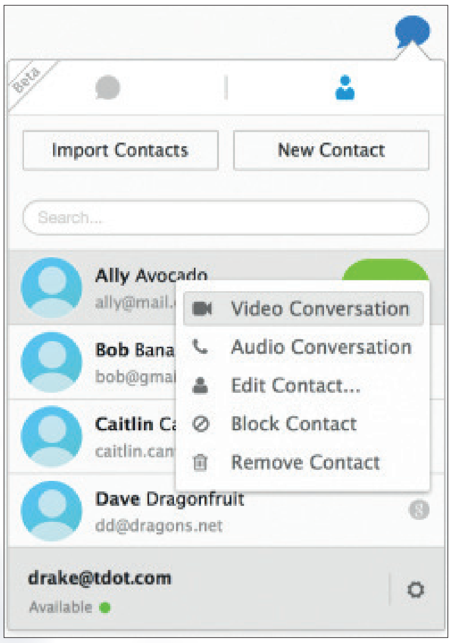
David Braben has given no firm schedule for a Linux port of *Elite Dangerous*, so we may have a while to wait. Dang!

4.4 is one of the most ambitious versions yet, and has seen work across all areas of the suite. From an end-user perspective, one of the biggest changes is the revamped toolbars: in *Writer* and *Calc*, the toolbars have been reorganised to remove lesser-used features, and add newer ones in their place. Not everyone will be happy with this, but on the whole we think it's a good thing, as we've often found ourselves scratching our heads with the default toolbars.

Also in *Writer*, a new Master Document Templates feature is available, which helps in the creation of frameworks for large documents, such as books with multiple chapters. Graphical shape objects can have text boxes embedded inside them – which is much more elegant than the previous, clumsy approach of adding text boxes on top and trying to keep everything together.

Calc, meanwhile, will have a new Statistics Wizard, along with an AGGREGATE function for better *Microsoft Excel* compatibility. The presentation component of *LibreOffice*, *Impress*, will also support password-protected documents for editing – that is, you can make a document read-only unless the viewer knows the password.

Under the hood, much work has been done on file format support: you'll be able to insert media in RealAudio, RealMedia, AC3, ASF and Ogg Opus formats, and import files from *Adobe PageMaker MacDraw*. *LibreOffice 4.4* will also be able to connect directly to *SharePoint 2010* and 2013. An especially useful addition is the ability to digitally sign PDFs as they're generated in the suite, and two new fonts have been added – Caladea and Carlito – which work as drop-in replacements for Microsoft's Cambria and Calibri.



Firefox Hello lets you set up free voice and video calls without accounts – you just share a simple URL.

Kernel goodies

Version 3.19 of the Linux kernel should be ready by the time you read this, and it packs in a bag of improvements across the board. On the graphics front, the DRM subsystem features Intel Skylake graphics support – Skylake being the processor architecture that's due to be the successor to Broadwell later this year. The AMDKFD driver has been merged into the mainline kernel, and support has been added for GM204 Maxwell GPUs, as used in GeForce GTX 970 and 980 graphics cards.

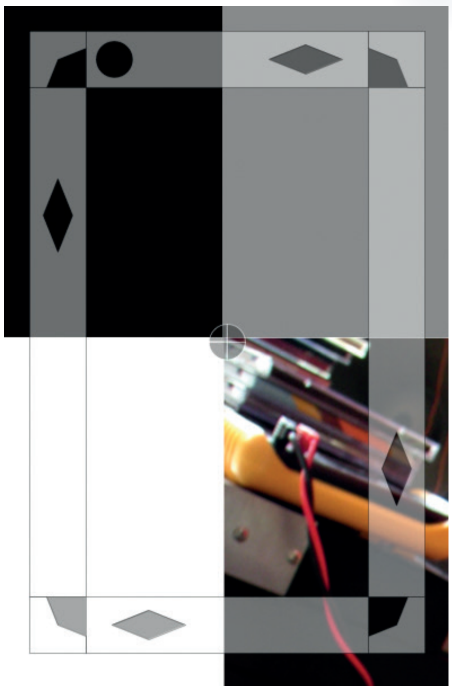
Meanwhile, in filesystem land there have been plenty of improvements: SquashFS, the compressed filesystem, now includes LZ4 compression, which has lower CPU and memory requirements than Zlib compression. F2FS, the Flash Friendly File System originally developed by Samsung for use on solid-state devices, has a new "fastboot" mount option for snappier boot speeds.

Btrfs users will find improved RAID 5 and 6 support, while OverlayFS, the union filesystem merged in kernel 3.18, will support multiple read-only layers. This means you can mount multiple volumes into the same directory. Then there are input driver updates (which should benefit Google Chromebooks), USB audio support improvements, merging of support for ARM's CoreSight debugging architecture, and much more. If you're running a rolling-release distro like Arch, you should get kernel 3.19 fairly swiftly – otherwise it will be in the next round of distro releases.

"Adoption of open formats looks set to continue, and the software is gettng better all the time."

Keep an eye on...

Maybe, just maybe, we'll see a new version of *Gimp* this year. The developers are keen to stress that there's no timeframe for version 2.10, which should fully support GEGL (the Generic Graphics Library, which has been under development since the year 2000) and all the niceties it brings, such as support for higher bit-depth images and non-destructive editing. It will also be possible to preview filter effects directly on the canvas, instead of the little filter window, while a unified transformation tool should make it into the release.



Gimp 2.10's unified transformation tool will let you scale, rotate, shear and add perspective at the same time.

But that's just the start: if we're very lucky, we may even see *Gimp 3.0* before the year is out, which will be ported to *GTK 3*. This will bring it in line with the Gnome 3 and Cinnamon desktops, and make it look much better on HiDPI displays. Other features planned for post-2.10 releases are script recording and playback, better text handling, and automatic layer boundary management. We can't wait!

Also in the realm of desktop software, *Firefox 35* is due out in January or February, and will include improvements to the *Firefox Hello* real-time communication client. It will be easier to start conversations with people over WebRTC and keep track of multiple chats. Also, the browser will default to using HTML5 video when accessing YouTube, which may finally mean the death of Flash for many users. Not that we're complaining – we've all had enough of that bloated, crash-prone, binary blob of evil.

The number of Linux users has never been higher; adoption of open formats looks set to continue; and the software is getting better all the time. The future's bright.

ASTROPI

Ben Everard is definitely not bitter that he didn't get the chance to program space computers when he was in school.

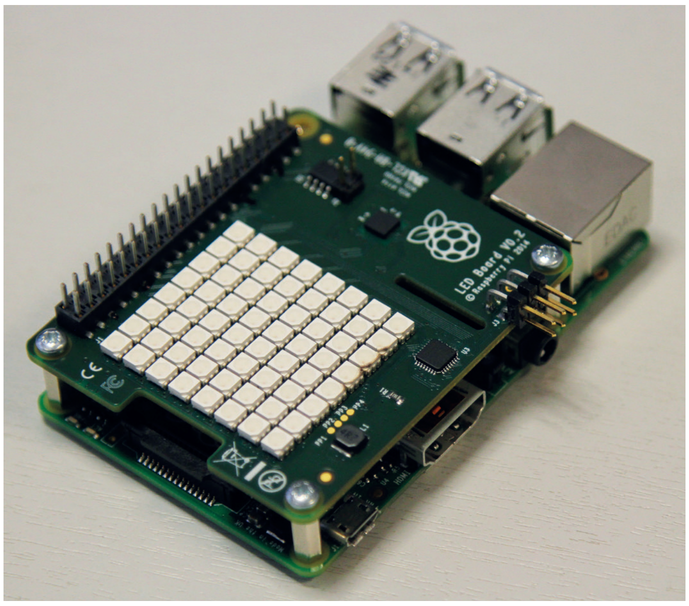
The most expensive structure ever built hurtles around the earth at almost 8km per second. It's 400km above sea level and needs to be occasionally pushed back up to stop it falling back down to earth. It's been mankind's home in the skies for over 14 years, and has provided innumerable scientific advances in that time. However, up until now, the International Space Station (ISS) has been missing one thing: a Raspberry Pi.

Fortunately, all this is about to change. Tim Peake, a British astronaut, will be taking two Raspberry Pis with him when he goes up to the ISS towards the end of 2015, and the Raspberry Pi Foundation is running a competition to see what runs on these boards.

The competition is open to school children in the UK (with separate categories for primary and each key stage of secondary school). Entrants don't have to be expert programmers, as the Pi Foundation is on hand to help implement the plans; the main challenge of the competition is to come up with ideas for what experiments to run on the computers.

Just running software on a Raspberry Pi wouldn't be any different in space to on earth, so in order to take advantage of the extraterrestrial location, the space Pis will be fitted with Astro Pi HATs, which include a whole range of sensors:

- **Gyroscope** In space, there's no up or down. Without gravity, the concepts don't really make sense, so what does this mean for the ISS? Is it spinning, and if so, how fast?
- **Accelerometer** The ISS is moving very fast, but is it's decelerating slowly. This means it's constantly falling towards the earth and occasionally has to be pushed back up into orbit. An accelerometer could be used to measure this falling as well as any irregularities in the station's orbit.
- **Magnetometer** Compasses always point north when you're on earth, but in space it's not quite so simple. With magnetic field fluctuations and solar wind, there's more variation. This sensor could show what's going on.
- **Temperature** Space is cold. Very cold, but above the atmosphere, the sun's rays are very powerful. What does this mean for life on the ISS, which crosses from summer to winter every 45 minutes?
- **Barometric pressure** The ISS is a tiny bubble of air floating through the vast vacuum of space. Only the thin walls stop the whole thing popping and dooming everyone on board to death. How well is the pressure maintained, and should the astronauts fear the bends?
- **Humidity** Humans are constantly breathing out wet air, and perspiring moisture through their skin. In a small enclosed space like the ISS, can this cause a problem?
- **Camera** A bird's-eye view of earth or an image of the stars uninhibited by any atmospheric haze. Which

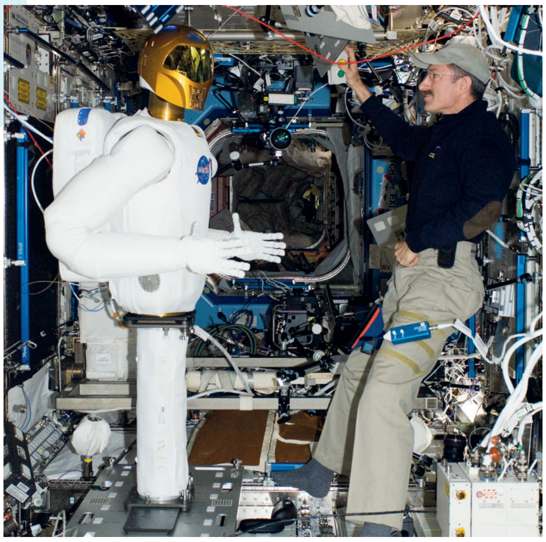

The Astro Pi HAT is the best sensor board on the market, and opens up a whole new world of possible Pi projects

Linux in space Penguins in orbit and beyond

These Raspberry Pis join a long and illustrious line of computers running Linux in space. In fact, the International Space Station is already a Linux-only affair. After a Windows virus outbreak in space (caused by an infected USB stick), all the laptops on board now run Debian.

As well as laptops, there are a large number of embedded Linux systems in space. R2, the humanoid robot on the ISS, is powered by Linux, as are all 71 of Planet Labs' Dove satellites that are swarming in the upper atmosphere and photographing the earth every day (we covered the Planet Labs project on page 32 of LV008).

Linux hasn't just been limited to Earth and its orbit. Linux also powered NASA's Spirit rover that went to Mars (the more recent Curiosity rover used VxWorks RTOS).



Left R2: The humanoid robot in space. The future is here and it's running Linux.
Right We don't know who the judges will be, but Major Peake has a degree in flight dynamics, so working that into a proposal may grab his interest.

would you choose?
- **IR Camera** Infra-red radiation shows different things to visible light, but what does this mean for the view from space?
- **Real Time Clock** As well as keeping tabs on the date and time, this could be used to determine the location of the ISS at any point, since its orbit is highly predictable.

In addition to these sensors, the Astro Pi HATs will have an 8 x 8 matrix of LEDs and some push buttons so the astronauts can interact with them.

We're not part of the team the decides what gets chosen and what doesn't, but if we were, we'd be most interested in projects that combine data from more than one sensor, for example, combining the gyroscope with temperature, humidity and barometric pressure to see how the atmosphere inside the ISS changes as the station spins and moved around the world.

The Astro Pi HAT has a great set of sensors, and obviously could be put to good use down here on Earth. Those of us not lucky enough to be able to design projects that run in space can still get hold of this hardware for terrestrial use. By the time you read this, they may be for sale commercially, as they are promised 'early in 2015'. Keep an eye on **raspberrypi.org** for details.

Although Britain has a strong space industry, traditionally this country has focussed more on hardware than people and few Britons have ever been to space (the first was Dr Helen Sharman, the first Yorkshirewoman in space). Major Peake will be the first to do so with government funding. This project is part of a government drive to get more British people – and school children in particular – interested in STEM subjects. In addition to funding this mission, the UK Space Agency has another £2 million of public money to help fund projects to get people interested in this mission. Hopefully, we'll see more funding like this from the UK government to invest in the future of one of our leading high-tech industries.

"Although Britain has a strong space industry, traditionally we've focussed more on hardware than people."

Not a UK student? More ways to get a computer into space

The Astro Pi competition is only open to school-age children in the UK, but that doesn't mean that other people can't access computing resources in space. California Polytechnic State University has developed the Cubesat system to make it easier to get into orbit. Cubesat is a standard specification for small satellites to make it easy for them to piggyback on launches for a tiny fraction of the normal launch cost. Developing Cubesats is far more involved than the Astro Pi competition, but you also have far more control over your computing.

Even though this is far cheaper than most space launches it can still be expensive, but there are a few programs that can help people get started. Keep an eye on **www.cubesat.org** for details of projects in your area. Alternatively, if you want to go it alone (and have the budget to), companies such as Innovative Solutions in Space (**www.isispace.nl/cms**) can help with everything including getting a launch.

The easiest option for most people is to cheat a little bit and only go as far as the upper atmosphere. Helium weather balloons (or high altitude balloons) can take loads up to about 40km above earth for far less even than the cost of a Cubesat. Dave Ackerman has sent several Raspberry Pis up already, and blogged about the experience on **www.daveakerman.com**. It might not technically be space, but you'd never know that from the photos that come back.

EFFING AWESOME

Take a peek into the origin of the world's leading defender of digital rights and understand what it takes to protect your civil liberties online with **Mayank Sharma**.

When Nelson Pavlosky and Luke Smith received the DMCA takedown notice they weren't really surprised. After all, the Philosophy sophomores at Swarthmore College in Philadelphia had uploaded leaked memos from the billion-dollar voting machine manufacturer Diebold. Pavlosky and Smith weren't voting machine activists. However, they wanted to challenge Diebold in court for abusing copyright law. The teenagers argued that Diebold couldn't claim copyright over documents that revealed possible flaws with the voting machines and possible evidence of wrongdoing.

In an email exchange with us, Pavlosky recounted those times: "We were expecting legal threats from Diebold but we hadn't actually planned out exactly what we were going to do once we received them. Neither of us had sued a corporation before and we were fuzzy on the mechanics of doing so; where we would find the money to pay lawyers." That's when one of their friends pointed towards the Electronic Frontier Foundation (EFF). "The EFF informed us that they were already filing a lawsuit against Diebold on behalf of a non-profit ISP called the Online Policy Group, and they asked us if we'd like to join in on the lawsuit fun." The duo agreed. The EFF then put them in touch with the Stanford Cyberlaw Clinic, which represented them *pro-bono*. After a year, the students came out on top, won the case and set a

EFF's litigious journey

Since its inception in 1990, EFF has undertaken important cases and achieved landmark victories in its self-appointed role as protector of users' digital rights. Its opponents have included the US government, the Federal Communications Commission (FCC), and various entertainment and electronics companies, among others.

The EFF's long history of fighting new bills and amendments in the US courts has resulted in several celebrated cases relating to free speech, file sharing, digital rights management, privacy, patents, NSA spying, and more.

In the Diebold case mentioned in the feature, the EFF set a precedent by fighting on behalf of an ISP, exposing Diebold's bogus copyright claims. Also mentioned in the feature is the Bernstein case, when the EFF took on the US government and got a favourable verdict which forced it to change the draconian laws that treated privacy protection as a potential threat to national security.

In one of the famous victory for free speech the EFF successfully defended a group of online journalists against subpoenas from Apple Computers, which was later ruled to be unenforceable. The foundation also filed a class action lawsuit against Sony BMG for distributing music CDs that contained software to implement copy protection and Digital Rights Management and also covertly allowed the company to spy on the consumer's listening behaviour. Eventually Sony agreed to withdraw the discs and took steps to fix the damage.

Recently, in October 2013, EFF filed a petition at the US Patent and Trademark Office to take a second look at the patent owned by Personal Audio LLC. The company claims its patents cover all forms of podcasting and has already sued a handful of small and large podcasters. In April 2014, the Patent Trial and Appeal Board (PTAB) sided with the EFF and has allowed EFF's petition and ordered a review.

The EFF is also working with the the American Civil Liberties Union (ACLU) as co-counsel to craft an appeal to a ruling in a case filed by an Idaho nurse against President Barack Obama and several US intelligence agencies for violating her Fourth Amendment rights by the various NSA mass-surveillance programs.

legal precedent, which Pavlosky credits to the EFF. "Without the EFF's help, we wouldn't have had a clue what to do, and we wouldn't have become part of a court case that is cited in cyber law textbooks around the country."

The EFF bomb

"The EFF is a non-profit civil liberties organisation that works to defend your rights in the digital world," sums up Shari Steele, EFF's Executive Director, in a video titled "EFF's Five Famous Friends". The video also features Sci-Fi writer Cory Doctorow, who credits the EFF for making sure "that those civil liberties values that informed the best parts of democracy for centuries, make the transition to the 21st century, make the transition to the internet."

EFF was founded in July 1990 as an international digital rights group. The term digital rights, in simple terms, is used to describe the rights that enable users to access, use, create and publish digital media. It also covers the right to access, and use of communication networks, computers and electronic devices.

One of the main catalysts that led to the formation of the EFF was the raid on Steve Jackson Games, a small games book publisher in Texas. As part of a series of raids in early 1990 to track a copied document that described the working of the emergency 911 system, the United States Secret Service seized all electronic equipment and copies of an upcoming game books from Steve Jackson Games. While the equipment was returned when they couldn't find any evidence of wrongdoing, the small-time business was left in tatters.

This event convinced a bunch of Silicon Valley technologists that the authorities weren't informed about emerging forms of online communication and that there was a need for increased protection for civil liberties in this digital era. That's when Mitch Kapor, former president of the Lotus Development Corporation, John Perry Barlow, a poet and musician, and John Gilmore, a prolific free software hacker and co-founder of Cygnus Solutions, got together and founded the EFF in July 1990. Kapor provided the initial funding and Apple co-founder Steve Wozniak also pitched in.

However, forming a foundation wasn't the initial plan. Barlow and Kapor just wanted to hire a law firm and sue the Secret Service on behalf of Steve Jackson Games. In a keynote at the 2008 Digital Freedom Conference in Iceland, Barlow recounted that the plan was to "re-assert the constitution in cyberspace with a couple of surgical legal actions." However, at around that time Barlow received an email from someone in what was still the Soviet Union who had heard of their actions to protect the American constitution on the internet and asked "What about us? We don't have a constitution."

"The EFF is a non-profit civil liberties organisation that works to defend your rights."

Nelson Pavlosky completed his BA in Philosophy and went on to graduate from George Mason Law School.

Richard Esguerra, EFF's Development Director, mc'ing the annual Pioneer Awards. The awards recognise individuals "who have made significant contributions to the empowerment of individuals in using computers".
Image credit: Alex Schoenfeldt

EFF staffers Aaron Jue and Magdalena Kazmierczak talking to a new EFF member.
Image credit: Alex Schoenfeldt

According to Barlow, that's when it "dawned on us that in cyberspace the Bill of Rights is a set of local ordinances. And in fact no rights can be conferred in cyberspace by anybody." That's when they decided to create an organisation with a much wider mandate.

EFF's first order of business was to sue the US Secret Service on behalf of Steve Jackson Games and several of the company's bulletin board users whose personal messages had been accessed and deleted by Secret Service agents. Since none of the EFF's founding members were lawyers, the tech press of the time didn't buy the argument that it was trying to safeguard the civil liberty issues of the falsely accused and labelled the EFF a "defence fund for hackers".

However, a couple of years later Sam Sparks, the US District Judge hearing this case, not only reprimanded the Secret Service for its actions but also held that electronic mail deserves at least as much protection as telephone calls. It is because of this case that law enforcement agencies (in the US, at least) must now produce a warrant that particularly describes all electronic mail messages before going through them.

The EFF next came to the rescue of Dan Bernstein, a University of California mathematics PhD student, who was prevented by the US Department of Justice from publishing his encryption program on the internet. The EFF triumphed again, and Judge Marilyn Hall Patel ruled, for the first time ever, that written software code is speech protected by the First Amendment. The case also forced the government to change its regulations and allowed everyone to publish encryption software on the internet without prior permission from the US government.

By the time the verdicts were delivered in these cases, the EFF had established itself as an authority in the civil libertarian community and was a bubbling wellspring of information on electronic civil rights and privacy issues.

Electronic freedom fighters

Fast forward a couple of decades and the EFF today is an expansive organisation. It is particularly interested in issues related to fair use, free speech and privacy. They'll take on copyright trolls, DMCA and patent abusers, help protect bloggers' rights, and support initiatives against mass surveillance, by the government and by the internet companies.

The EFF not only provides legal assistance to relevant cases, but also does a lot of advocacy work by organising campaigns to spread awareness and rally support against harmful actions and legislation.

Encrypting the web

EFF's involvement with encryption can be traced back to its origins, when it defended Dr Dan Bernstein and rewrote the rules of distributing encryption programs. Then in 1998 it built the DES cracker, which cracked the DES (Data Encryption Standard) by a brute force attack. It helped prove the lack of security of the DES, which back then was a federal standard.

The EFF is also the author of the popular *HTTPS Everywhere* browser plugin for *Firefox*, *Chrome*, and *Opera* browsers. The extension forces the connection to use the encrypted HTTPS connection when communicating with a website instead of the unsecure HTTP, which is the default on many websites. It also alerts you when you run into encrypted pages with links that take you to an unencrypted website.

Before launching the *HTTPS Everywhere* plugin the EFF began asking some of the largest sites on the web, including Google, Facebook, Twitter and Wikipedia, to start offering HTTPS versions of their sites. They also launched the SSL Observatory project, which investigates the certificates used to secure all of the sites encrypted with HTTPS.

In light of the NSA's surveillance programs, the EFF suggested five best practices for encryption for online companies. It also queried the leading companies about their initiatives to bolster encryption and published the results as part of its Encrypt The Web initiative. Recently, the foundation has announced the Let's Encrypt certification authority (CA) initiative, which will start issuing free-of-cost security certificates to websites in 2015.

ELECTRONIC FRONTIER FOUNDATION
DEFENDING YOUR RIGHTS IN THE DIGITAL WORLD
HOME
ABOUT
OUR WORK
DEEPLINKS BLOG
PRESS
HTTPS Everywhere

You can even install the *HTTPS Everywhere* extension in *Firefox* for Android.

It also publishes white papers and reports that assess and explain threats to the average user from the government as well as the industry. Finally, it also employs programmers who develop tools to keep users safe online.

The EFF keeps its eyes peeled for messages requesting legal assistance on **info@eff.org**. Communications with the EFF asking for legal counsel are protected by the attorney/client privilege regardless of whether or not the EFF takes up a case. Since it has limited resources, the EFF is likely to take on cases that will have a large impact on the law and whose decision will help define how the law is applied in future cases.

Working a lawsuit usually costs a lot of money, and since the EFF has limited resources it only helps those who simply can't afford legal representation. In their last financial period they spent over $120,000 (about £80,000) in legal and professional fees and almost $90,000 (about £60,000) for various litigation expenses. Rebecca Jeschke, the EFF's Media Relations Director and Digital Rights Analyst, told us that the EFF currently has a staff of 63 employees, of whom 20 are legal staff. They are however assisted by others, Jeschke says: "Our activists and technologists and analysts also help on legal cases for background and insight, etc." If the EFF can't take up your case, they'll redirect you to another attorney on their Cooperating Attorneys list, who are passionate about the same things as the EFF.

Advocacy is handled by the 15 activists and analysts employed for this purpose. Then there's the technology products team of seven, which develops tools such as *HTTPS Everywhere*. There are also general technologists who provide support and help build things like web pages for the activism, and folks who answer emails from members.

Support with your wallets

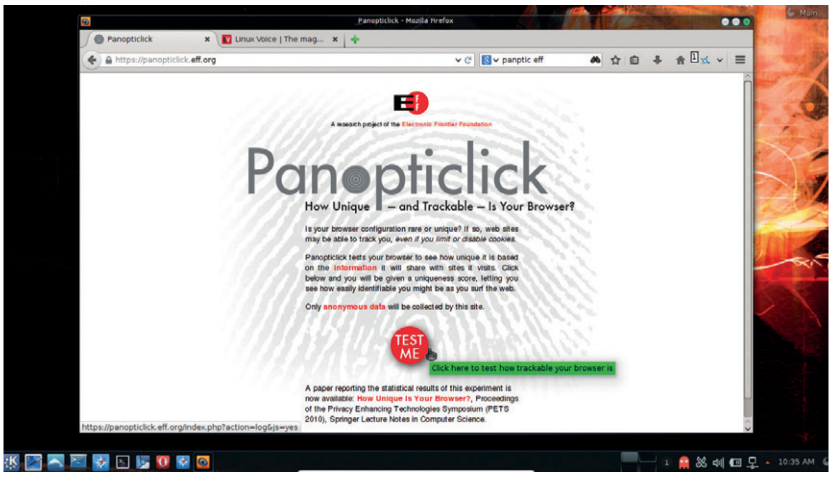
The EFF gets a high proportion of its funds through grants, so the foundation has to employ a bunch of people to help write the grant bids. Of its total income in the last financial year, over $2.5 million (about £1.6 million) were grants from various organisations. Another sizeable chunk of the income, over $2 million (about £1.3 million), were from the sales proceeds of the Humble Bundles. Humble Bundle Inc, shares its income with various charities and organisations including the EFF in a completely transparent fashion.

However, the biggest source of income for the EFF is from direct individual contributions. In the same financial year, the EFF received over $4.3 million (about £2.7 million) from individual supporters like you. Since the EFF is a registered 501(c)(3) non-profit organisation, all donations are tax deductible if you're a US citizen. All donations get you a 12-month membership and discounts on general admission to EFF events.

There are also several other ways you can support the foundation, such as by donating air miles and gifting stocks. Visit the supporters page at **https://supporters.eff.org** for more details of such programs.

We at Linux Voice also appreciate the EFF's efforts in safeguarding our digital rights, and it's our pleasure to include the EFF in our "Giving Profits Back" scheme. You can read about our profit sharing scheme in detail in Issue 10. As promised in our crowdfunding campaign, at the end of every year we'll give 50% of our profits back to Free Software and Linux communities as per our readers wishes and EFF is one of the organisations you can choose to give to.

The Electronic Frontier Foundation is doing important work, and the Snowden revelations show that it has a huge amount of work to keep on top of. Everything the organisation does, in one part or the other, reflects one of its founders, John Perry Barlow's, ambition to be a good ancestor: "If you want to be a good ancestor you want to keep the internet open to anything that flows over it, which might be awful stuff. But as John Stuart Mill said, liberty resides in the rights of that person whose views you find most odious." Keep up the good work, team!


Panopticlick - Mozilla Firefox
Panopticlick
Linux Voice | The mag...
https://panopticlick.eff.org
panptic eff
TEST ME
Click here to test how trackable your browser is
https://panopticlick.eff.org/index.php?action=log&js=yes

A research project of the Electronic Frontier Foundation

Panopticlick

How Unique – and Trackable – Is Your Browser?

Is your browser configuration rare or unique? If so, web sites may be able to track you, *even if you limit or disable cookies.*

Panopticlick tests your browser to see how unique it is based on the information it will share with sites it visits. Click below and you will be given a uniqueness score, letting you see how easily identifiable you might be as you surf the web.

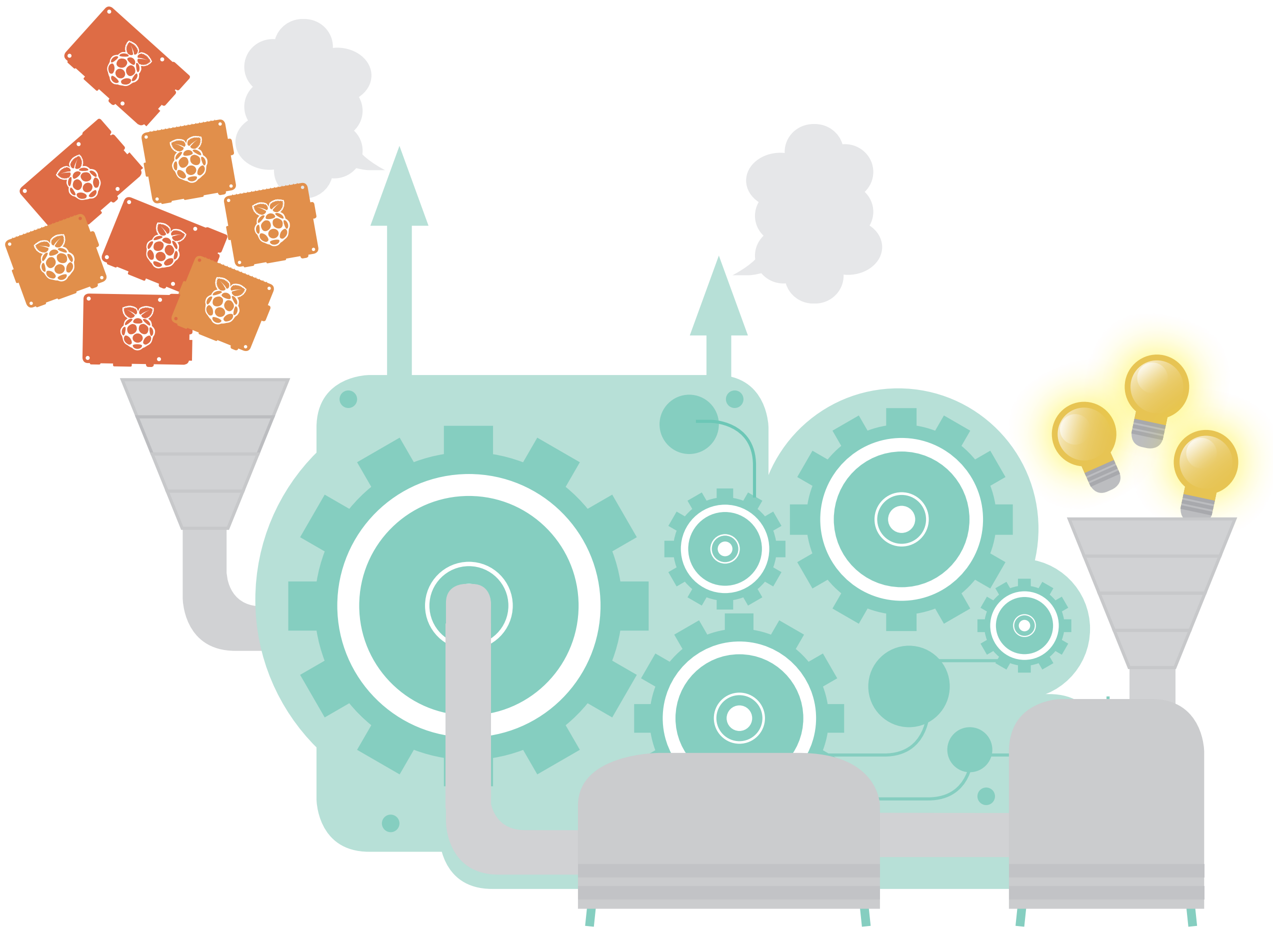
Only anonymous data will be collected by this site.

A paper reporting the statistical results of this experiment is now available: How Unique Is Your Browser?, Proceedings of the Privacy Enhancing Technologies Symposium (PETS 2010), Springer Lecture Notes in Computer Science.

Use Panopticlick (**http://panopticlick.eff.org**) to see how much information your web browser gives away to websites.

"The EFF is interested in issues related to fair use, free speech and privacy ."

Jacob Applebaum, a core member of the Tor project, sits under a dunk tank to raise money for the Electronic Frontier Foundation.
Image credit: Scott Beale / Laughing Squid



INSIDE THE PI FACTORY

Les Pounder bought a Raspberry Pi and inside was a golden ticket to the secret Pi factory.

The Sony UK Technology Centre in Pencoed, Wales, is the home of Sony's manufacturing operations for professional television broadcast equipment, but by sheer volume the biggest product manufactured there is the Raspberry Pi, with some 50,000 units being produced each week – roughly one new Raspberry Pi every 5.5 seconds.

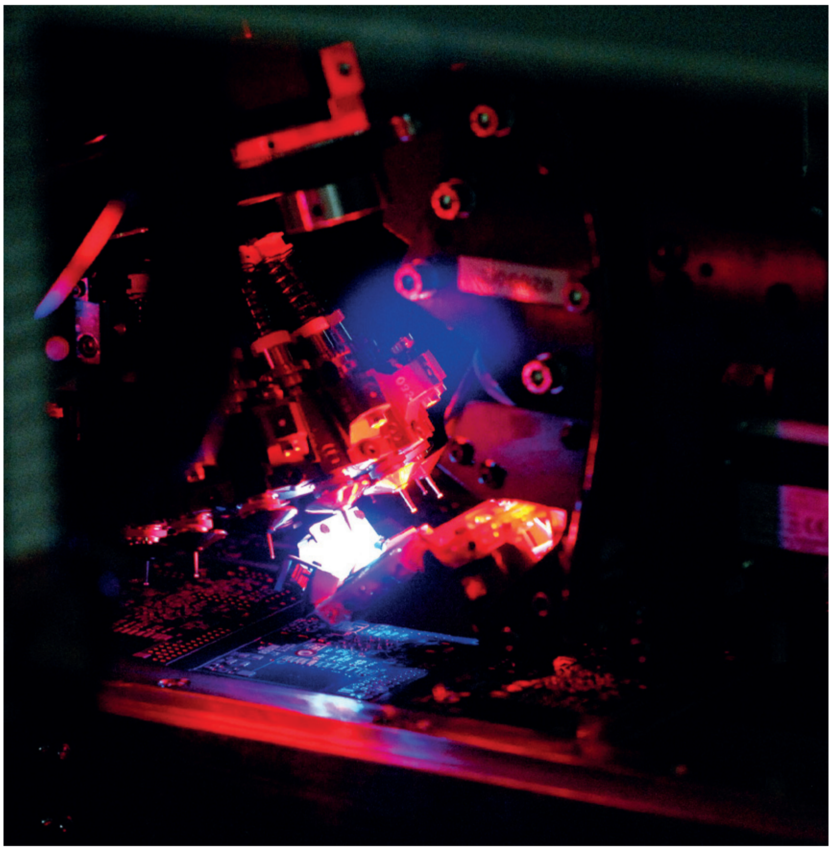
We asked Pete Lomas, the designer of the Pi, to explain how they can make so many models in one factory, including the newly released model A+

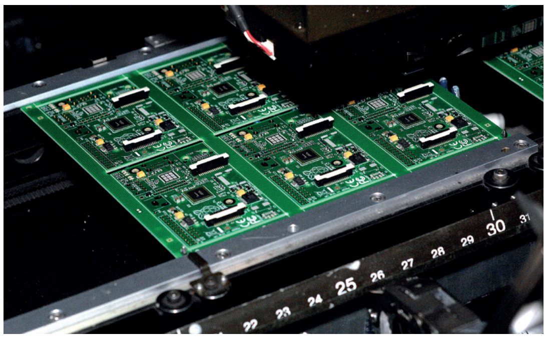
Process 1 and 2 – bottom SMT and top SMT

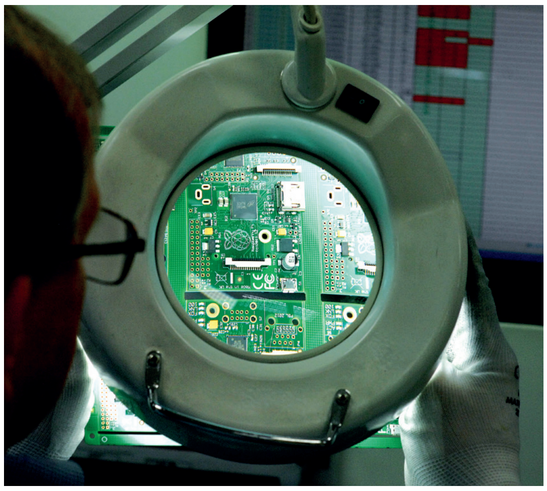
A Raspberry Pi starts life as one of six Pis in a panel for the model B+ (it's 8 per panel for the A+, making it about 30% faster to make) to reduce the handling of the bare PCBs (Printed Circuit Boards) and to group the boards into batches. Each panel is loaded into a hopper and in turn each panel makes its way through the vast line of machinery. First, the boards receive an application of solder paste via an extremely thin steel stencil. An optical check of each board ensures that the application has been done correctly.

With the solder paste in place, the Pis make their way through Sony's specially designed SMT (Surface Mount) mounting machine, which is home to hundreds of reels containing thousands of SMT components such as resistors, capacitors etc. These components are removed from sealed pockets on the reels by advanced pick-and-place machines operating in parallel at tremendous speeds to ensure a smooth operation. These are then gently pressed into the solder paste on the board.

With all the SMT parts in place, the board is then run through the reflow oven to heat the solder paste and the components so that they are bonded together. The SMT process is repeated for the underside and







top of the Raspberry Pi, with special attention being paid to the BCM2835 processor unit, which requires a memory chip to be installed on top of it using a specialist package-on-package (or PoP) applicator that adds a small amount of solder paste to the balls of solder on the bottom of the memory chip; this is then lowered onto the BCM2835 to make the connections.

Process 3 – PTH (through-hole)

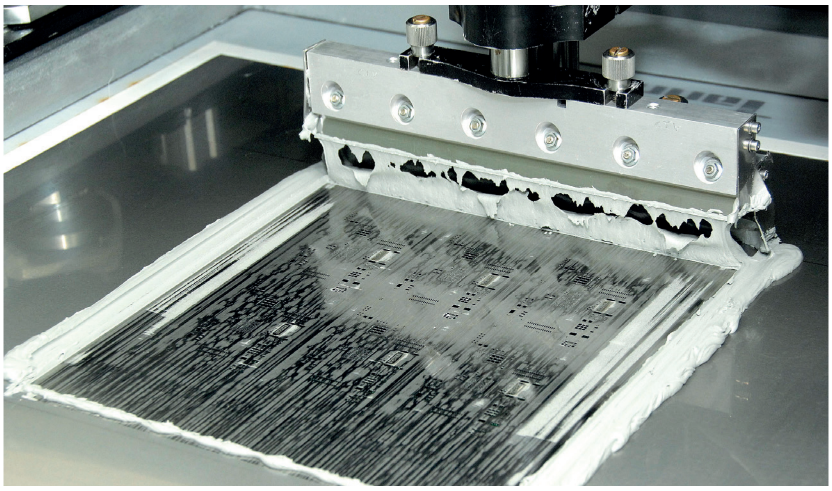
The Raspberry Pi has only a handful of through-hole components (such as the Ethernet jack) which are added to the panel by a team of skilled operatives. The panel is then placed into a heat-resistant frame to protect the SMT components from desoldering in the next stage. After a quick spray of flux to the underside of the Pis the panel is sent through a wave soldering machine, which will solder the PTH components to the Pis via a wave of molten solder. Using the wave soldering station takes a high level of accuracy to get right, as too much power to the wave can cause the molten solder to flow over the board and ruin a batch of Pis.

Once the PTH stage is complete the Pis are broken from their panels and passed to the test teams who perform automated tests of each Pi. For each testing station there are two Pis being tested at any one time and where a defect is found (this is very uncommon) they are placed into a fail box and an operative will investigate to see if any optimisations can be made to the production process to reduce or remove this fault from the process. Sony take these optimisations very seriously and there is a wall of fame out in the 200 metre long corridor where staff are rewarded for their improvements and there were quite a few Raspberry Pi flavoured awards on the board.

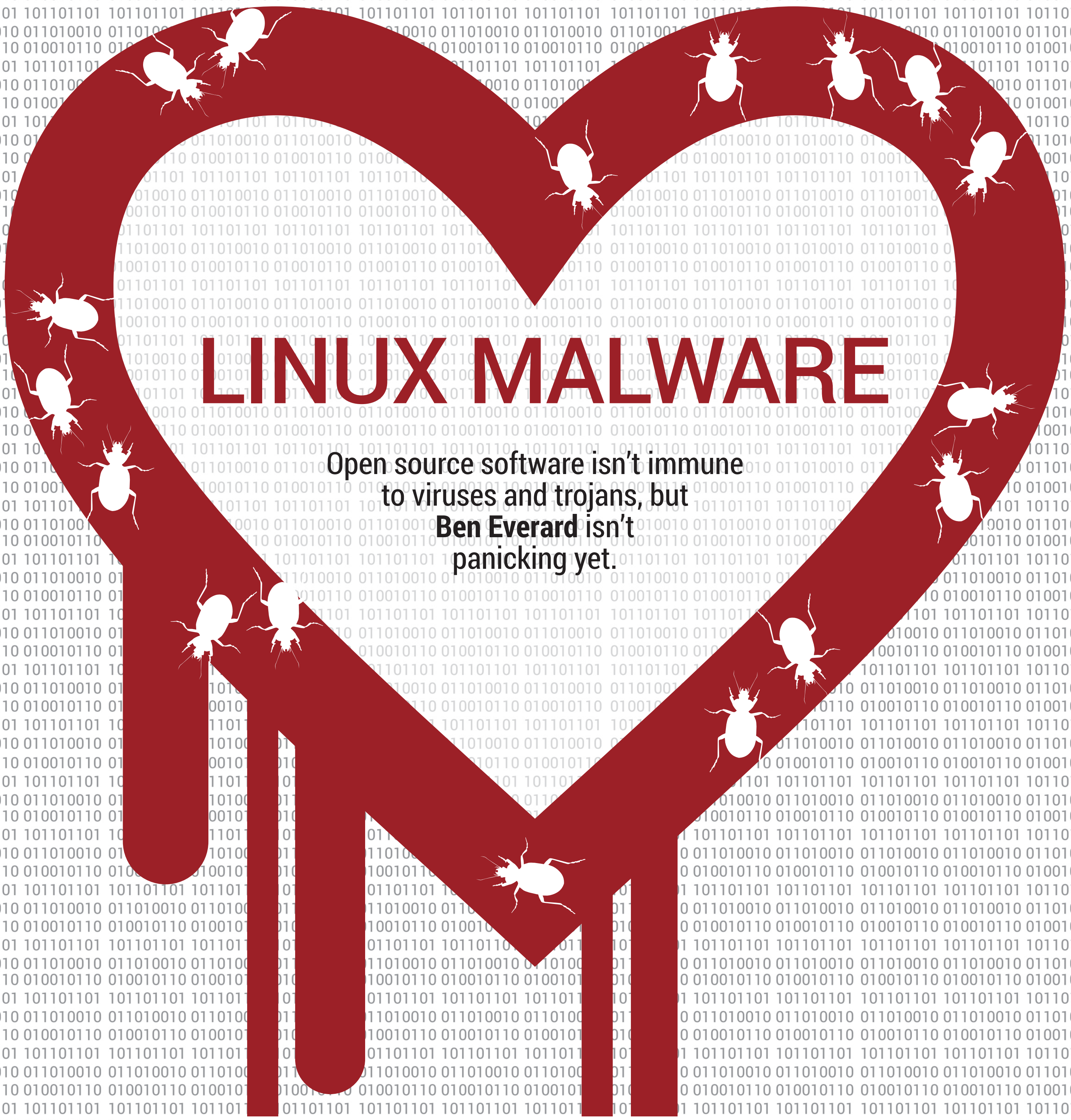
After the tests are completed each of the Raspberry Pis are placed into an anti-static bag and boxed into the correct packaging for the distributor, either RS (Allied) and Farnell (element14). For all of the above processes there are three production lines which are working together to produce around 10,000 Raspberry Pi per day – and all made in the UK.

From here the Raspberry Pi is sent to stores ready for you to purchase, where it will be taken home and used to power projects both big and small, lighting the fire of imagination inside a child's mind and enabling them to enjoy learning not just about computer science but about the worlds of science, engineering and technology that can benefit greatly from this small PCB. LV

Top Placing the SMT components is a precise and rapid part of the process, with thousands of components loaded on to massive reels ready to be used.
Top-right A panel is made up of six Raspberry Pis (eight for the A+) and they will go through all of the processes as a group.



Using a thin steel mask, solder paste is applied to the Raspberry Pi PCB.



LINUX MALWARE

Open source software isn't immune to viruses and trojans, but **Ben Everard** isn't panicking yet.

Linux is generally considered to be virus-free for all practical purposes. No major distros ship with anti-virus software running, and this is not considered a problem. Most Linux users never install any specific security software, and also never run into any issues. However, as Linux becomes more popular, it also becomes a more attractive target for malware creators. In 2013, we saw the release of the first major piece of malware targeting Linux desktop users in the shape of the Hand of Thief banking trojan. In 2014, the trend has continued with several major Linux malware stories in the news. Is it time to rethink the maxim that Linux is secure?

Trojans are pieces of software that hide from the user and steal data from within. Unlike viruses, they don't usually have a mechanism for self-replicating, so they require the user to be tricked into installing them. They're a major problem for people who download software from dubious internet sites. Usually, they attack Windows, but Hand of Thief goes after Linux users. Hand Of Thief is designed to steal information from web browser sessions, specifically login information for internet banking. It can grab data from HTML forms, and other details from the web browser, and relay them back to the attacker. It's also reported to be able to prevent users from accessing anti-virus sites in order to make it harder for them to identify an infection. The Linux version of this trojan has been offered for sale in Russian malware forums for an amount in the region of thousands of pounds, so some people are obviously keen to target Linux users .

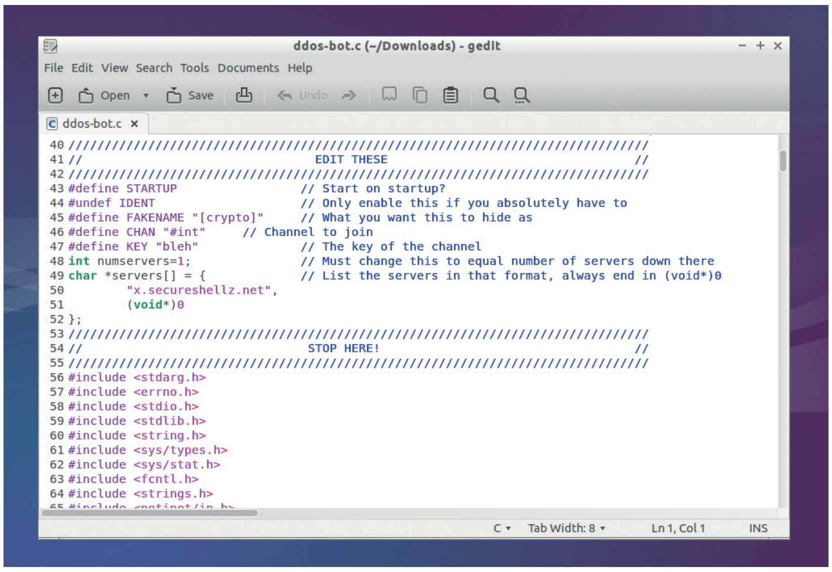
Despite being quite powerful once it's installed, Hand of Thief doesn't have a good method of infection. It requires the user to be tricked into installing it manually. This sort of thing is quite familiar to Windows users. For example, many people have received bogus phone calls from people claiming to work for Microsoft saying something along the lines of, 'we have detected a problem with your computer and you need to install some software to fix it'. They will then talk the unsuspecting user into downloading and installing some trojan. This sort of thing is unlikely to work with Linux users both because they tend to be more knowledgeable about computing and because most users would be suspicious of any software that doesn't come from their package manager.

The hard part of Linux malware isn't controlling the system once you're in; it's infecting the system in the first place. This doesn't mean that we Linux users can disregard malware completely though. In 2014 we found that we too were vulnerable to bugs that allow rapid infection of a large number of machines.

"The hard part of Linux malware isn't controlling the system; it's infecting it in the first place."

Shellshocked

A code execution bug in *Bash* may not immediately seem like a big problem. After all, code execution is the entire point of a shell. However, *Bash* is used to span new sessions in when generating web pages under some server setups. Shellshock – a *Bash* code execution vulnerability announced in September 2014 – allowed attackers to execute code on a server by sending specially crafted HTTP requests that exploited this session spawning.



```
ddos-bot.c (~/Downloads) - gedit
////////////////////////////////////////////////////////////////////////////////
//                              EDIT THESE                                    //
////////////////////////////////////////////////////////////////////////////////
#define STARTUP                         // Start on startup?
#undef IDENT                            // Only enable this if you absolutely have to
#define FAKENAME "[crypto]"             // What you want this to hide as
#define CHAN "#int"             // Channel to join
#define KEY "bleh"                      // The key of the channel
int numservers=1;                       // Must change this to equal number of servers down there
char *servers[] = {                     // List the servers in that format, always end in (void*)0
        "x.secureshellz.net",
        (void*)0
};
////////////////////////////////////////////////////////////////////////////////
//                             STOP HERE!                                     //
////////////////////////////////////////////////////////////////////////////////
#include <stdarg.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <strings.h>
```

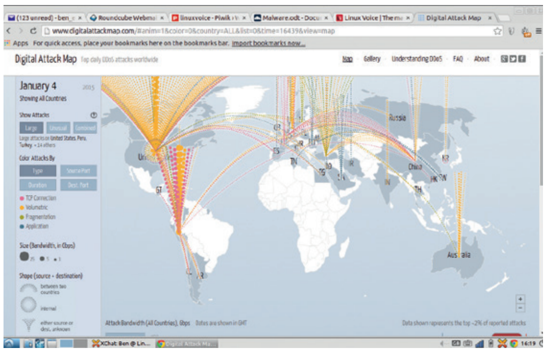
After the Shellshock news, crackers tried to exploit our servers; helpfully, they furnished us with commented source code to their malware. It made analysis much easier.

Within a few hours of Shellshock's announcement, malware writers had adapted their code and were scanning huge swathes of the internet for vulnerable servers, trying to infect them using this as a vector.

We have a couple of servers at Linux Voice: one for the website, and one for our internal use. These both run CentOS and were vulnerable to Shellshock (but our server setup wouldn't allow remote execution). We patched them quickly, and didn't have any problems. However, we could see all the attempts to exploit us, and we kept track of the code that attackers were trying to run on our servers. The vast majority tried to use the vulnerability to use **wget** to download some malware onto our server. Seeing these, we grabbed the malware to see what people were trying to run.

All of the malware that targeted us tried to enlist our servers into Distributed Denial of Service (DDOS) botnets. There were bots written in Perl, Python and C (sent as source code to be compiled on the server), but they all worked in roughly the same way.

An important part of any botnet is the command and control setup. This is the mechanism that enables



You can view a map of DDOS attacks as they happen at **www.digitalattackmap.com**, and even scroll back to big attacks in the past.

Embedded malware Inherent insecurity

Embedded devices pose a new type of threat, and one that could be hard to combat unless there's a radical rethink of the way embedded software is maintained. In the past, devices have shipped with software pre-installed, and normally left with the same software for their entire lifetime. This means few embedded devices ever get patched against bugs, few even have their default passwords changed.

Linux can be secure, but it isn't automatically. Unless manufacturers start taking security seriously, and enable users to update their devices, then the internet of things will continue to be an increasingly attractive target for malware developers, regardless of what OS is running it.

Internet of Things When everything is connected, everything is a target

More and more things besides desktop and server computers are being connected to the internet, and anything connected to the internet can get malware. Phones are the most visible aspect of this internet of things (see Android boxout, below), but they're not the only type of device.

Routers and ADSL modems pose a particularly attractive target to malware developers. They're usually left switched on 24 hours a day and permanently connected to the internet, and they sit between people browsing the web and the internet itself, so they're perfectly placed for performing man-in-the-middle attacks.

The first major attack of this type, known as Hydra, happened in 2008. Hydra was an IRC controlled worm that infected Linux systems running on the MIPS architecture (as many routers do) using both a brute force password guesser and an authentication bypass exploit.

Hydra itself wasn't particularly malicious. Infected routers were just used to infect more routers. However, there are many pieces of malware that have since been built based on the Hydra code which are more malicious.

Not all attacks on routers need malware though. One reason for targeting routers is to launch man-in-the-middle attacks people browsing the web, and this can be done quite effectively by altering the router's Domain Name System (DNS) settings. By altering these, an attacker can send all of a router's data through a machine that the attacker controls allowing them to intercept – and change – any of the data. Recently, attackers have used this to target banks in Brazil and Eastern Europe.

The internet of things is rapidly expanding, and there's seemingly no end to the range of devices some manufacturers try to connect to the internet. It goes far beyond routers to more mundane items like televisions and fridges. Although these aren't positioned on the network in such a way as to allow them to run man-in-the-middle attacks like compromised routers can, they could still be used in DDOS botnets.

The trend for embedded Linux devices is increasing, and they're becoming a more and more attractive target for malware. However, few manufacturers make it easy to keep software up to date, and even fewer users actually do. This means that security bugs often sit around unfixed for years. As we saw with Shellshock, attackers can quickly exploit these.

the attacker to communicate with the bots (and ideally with lots of bots concurrently), but at the same time allows the attacker to remain untraceable when the malware gets discovered.

All the bots that we saw used IRC for this. The bots included the details of servers, channels and passwords to connect with. As soon as they ran, they connected to a particular IRC channel, which they listened to, then they acted depending on what was sent on the channel. The people running the botnet could then connect to the same channel through an anonymising proxy and be untraceable.

Helpfully, all the bots were well commented with instructions for use. Here's the comments describing the IRC commands for one bot:

```
The syntax is:
    !<nick> <command>
You send this message to the channel that is defined later in this
code.
Where <nick> is the nickname of the client (which can include
wildcards)
and the command is the command that should be sent.  For
example, if you
want to tell all the clients with the nickname starting with N, to
send you
the help message, you type in the channel:
    !N* HELP
That will send you a list of all the commands.  You can also
specify an
astrick alone to make all client do a specific command:
    !* SH uname -a
There are a number of commands that can be sent to the client:
    TSUNAMI <target> <secs>      = A PUSH+ACK flooder
    PAN <target> <port> <secs>   = A SYN flooder
    UDP <target> <port> <secs>   = An UDP flooder
    UNKNOWN <target> <secs>      = Another non-spoof udp
flooder
    NICK <nick>           = Changes the nick of the client
    SERVER <server>       = Changes servers
    GETSPOOFS             = Gets the current spoofing
    SPOOFS <subnet>       = Changes spoofing to a subnet
    DISABLE               = Disables all packeting from this bot
    ENABLE                = Enables all packeting from this bot
    KILL                  = Kills the knight
    GET <http address> <save as>  = Downloads a file off the web
    VERSION               = Requests version of knight
    KILLALL               = Kills all current packeting
    HELP                  = Displays this
    IRC <command>         = Sends this command to the server
    SH <command>          = Executes a command
```

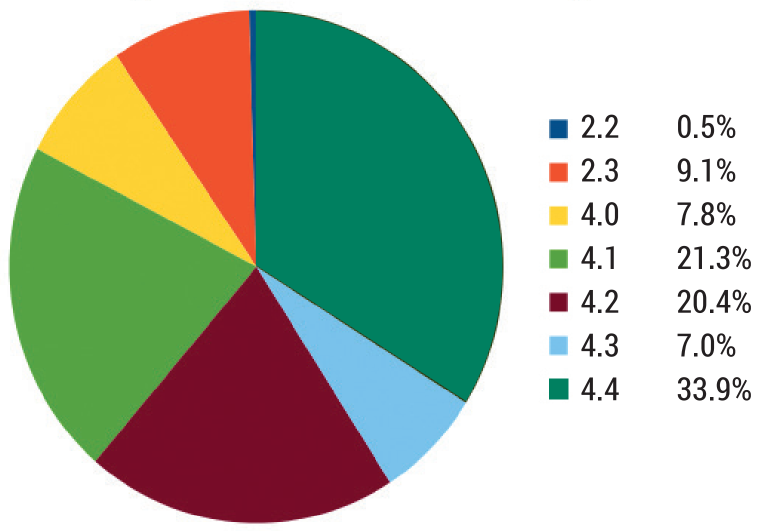
By getting all the bots to connect to a single channel, and allowing wildcards in the commands, the controllers can easily launch an attack using a large number of infected computers – this shows that the bot is designed for DDOSing.

The **SH** command gives the controller the power to execute arbitrary code, so the bots could be used for more than DDOSing even though this appears to be their primary purpose. Amassing network bandwidth is usually the aim of server malware.

The Active Threat Level Analysis System (ATLAS) run by Arbour networks monitors DDOS attacks around the world through ISPs sharing their data (see

Only about 30% of Android devices are running the latest version of the OS, which means many are vulnerable to known security bugs.

Percentage of Android devices by version



https://atlas.arbor.net for more information). In 2012, the largest attack reported by Atlas used a peak bandwidth of 100Gbps. By 2014, that had increased to 325Gbps with attacks over 100Gbps occurring almost every month. In other words, DDOSing is becoming big business, and attackers need larger and larger botnets in order to keep up with the competition. Shellshock provided one such source of new servers, but every vulnerability that can be exploited in this way will be. These botnets are then rented out (often by the hour) to whoever has a desire to take a site offline.

Another common goal of malware on Linux servers is as a vector to infect other machines (typically ones running Windows). In this scenario, when a Linux web server is compromised, the attackers then use it to deliver malware to people who visit the websites hosted on the server through so-called drive-by downloads.

Advanced Persistent Threats

Late in 2014, news surfaced of a newly discovered Linux component of the Turla Advanced Persistent Threat (APT). APTs are targeted malware that's designed to get into an organisation and stay there allowing attackers to steal large amounts of data, or monitor activity for a long time. This is the sort of malware used in industrial espionage, or by nation-state spies.

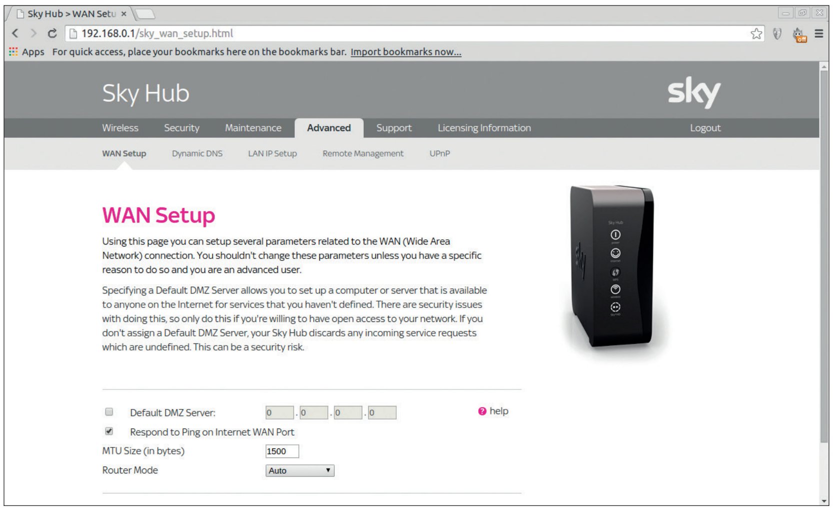
Turla is a suite of APT malware that's been known about for some time. It's thought to be linked to the Russian government, and has been used to spy on governments and militaries around the world. However, up until now, all known components have targeted Windows.

We haven't been able to get hold of a copy of Turla to analyse. However, Turla is based on an older backdoor called cd00r developed by Phenoelit – Turla uses the same method used by cd00r to stay invisible to normal socket monitoring tools while still being contactable from outside for example. A normal TCP connection starts with a three-way handshake. First, the client sends a SYN packet. The server then sends a SYN-ACK packet, and finally, the client responds with an ACK packet. At the end of this handshake, both machines know that they're communicating with a machine that's switched on and working, so they can then start to transfer real data.

From a network security point of view, this means that a computer can always tell which ports on a server are open by sending SYN packets. If they get a SYN-ACK in response then they know that some software is listening on that port. On the server side, Linux **netstat** or **ss** can be used to show which processes are using a particular port.

Cd00r doesn't listen on a port in a normal way. It doesn't respond to SYN packets with SYN-ACK packets. Instead, it sniffs packages that go to a range of ports, and looks for SYN packets being sent to several specified ports in order. For example, the code to trigger cd00r could be SYN packets to port 55, 74, 12, 90 then 45. Once it detected this pattern of SYN packets, it would trigger a piece of code. This is know as port knocking. Turla works in a similar way, but instead of listening for a pattern of SYN packets, it listens for special values in the SYN packets.

This method has allowed Turla to remain undiscovered on Linux for at least four years. If you fancy having a look at how Turla works, the cd00r source code is available at **www.phenoelit.org/fr/tools.html**.



Sky Hub > WAN Setu ×

192.168.0.1/sky_wan_setup.html

Apps For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

Sky Hub

Wireless | Security | Maintenance | Advanced | Support | Licensing Information | Logout

WAN Setup | Dynamic DNS | LAN IP Setup | Remote Management | UPnP

WAN Setup

Using this page you can setup several parameters related to the WAN (Wide Area Network) connection. You shouldn't change these parameters unless you have a specific reason to do so and you are an advanced user.

Specifying a Default DMZ Server allows you to set up a computer or server that is available to anyone on the Internet for services that you haven't defined. There are security issues with doing this, so only do this if you're willing to have open access to your network. If you don't assign a Default DMZ Server, your Sky Hub discards any incoming service requests which are undefined. This can be a security risk.

☐ Default DMZ Server: 0 . 0 . 0 . 0 — help

☑ Respond to Ping on Internet WAN Port

MTU Size (in bytes): 1500

Router Mode: Auto

Many routers, like this one, can be configured through an HTML interface. This potentially makes them vulnerable to cross-site request forgery attacks (CSRF).

Android Malware Viruses in your pocket

Android – the world's most popular smartphone operating system – is also a distribution of Linux and it s useful case study for how malware can affect this OS. On the desktop, Linux is used much less than Windows or Mac OS X, and has far fewer problems with malware. On phones, Linux (as Android) has a much larger market share than any of the competitors (around 85% of all smartphone users are on Android), and also has the majority of malware (around 97% of smartphone malware is on Android).

The situation isn't as simple as it may first appear though. Smartphone software is typically installed through app stores. Android is the only popular platform that allows appstores other than its own, and it's on these third-party app stores that most of the malware exists. That's not to say there isn't any malware on Google Play Store, but that it's a small proportion of the overall situation.

Don't panic!

Servers are far more public than most computers, and so are most vulnerable to attack when a weakness is discovered. The popularity of Linux on the server market means that Linux vulnerabilities are exploited heavily once they are discovered.

Desktop computers, on the other hand, are not usually reachable from the internet, and this means attackers have far fewer chances to access them. Linux distros' repositories are our greatest strength in fighting malware on the desktop. As long as you only use trusted repositories, you're unlikely to have any problem with malware. That's true now, and it will remain true for the foreseeable future.

FAQ

SNAPPY UBUNTU CORE

Just in time for Ubuntu's new phone, we're condensing a new buzzword that embraces both the handheld and the cloud.

GRAHAM MORRISON

Q Oh no! Not another Ubuntu-related catchphrase that will fail to deliver on time!

A While we agree there have been a few, perhaps, hyperbolic Ubuntu-related statements made over the last few years, Canonical is getting better at controlling its enthusiasm. The Ubuntu Phone is imminent, for example, and Snappy Ubuntu Core is already available and doing things. We think it's excellent, and it's no coincidence that it arrives at almost the same time as the long-developed Ubuntu Phone.

Q If this is something genuinely useful, why the weird name?

A The weird name is because this is actually the combination of two things. The sober half is 'Ubuntu Core'. This can be explained in Linux terms you may already be familiar with – it's a tiny Ubuntu image that creates a minimal functional userspace environment from where other packages can be installed. As a compressed boot image, it uses less that 100MB, but size isn't its most important characteristic. It's that the image includes all you need to build your own more complex installation.

Ubuntu Core has been around in various forms for years, and it's designed to be a platform for creating your own distribution, with only those packages installed you need and nothing else, providing nothing more than a platform for package management. As Canonical puts it, "it's an engine, chassis and wheels, no luxuries, just what you need for massively parallel systems."

'Snappy', on the other hand, is less descriptive, and marks a new direction for Ubuntu Core. We'd like to guess that its name is the result of a marketing/PR brainstorming session where delegates were fed a diet of terms like 'Docker', 'Cloud', 'Synergy' and 'Leverage.' For now let's describe Snappy in hugely simplistic terms – it's a package manager for the new Ubuntu Core, but we'll flesh out this answer a little more in a few questions time.

"The OpenStack User Survey from 2014 listed Ubuntu as the preferred distro for the cloud ."

Q Does that mean this initiative has something to do with that web of network services known colloquially as 'the cloud'?

A Yes it does. Ubuntu is actually a massively popular cloud operating system. The OpenStack User Survey from 2014, for example, listed Ubuntu as the most preferred distribution for the cloud, followed by RHEL and CentOS. It's also popular on Google's and Amazon's cloud platforms, as well as HP's and even Microsoft's. Unfortunately for Canonical, and despite the incredible growth of cloud-related business, the company receives very little from all those businesses spinning up instances of Ubuntu. We're guessing that Canonical's latest cloud initiatives are an attempt to harness some of that Ubuntu love and put its service provision at the heart of cloud users, perhaps with an option for providing paid-for managed subscriptions at some point.

Q Hasn't Canonical covered this with Ubuntu Server?

A Ubuntu Server has been one of our favourite spins of Ubuntu for some time. It's a full version of Ubuntu without many desktop packages pre-installed. It will even install a LAMP stack (Linux, *Apache*, *MySQL* and PHP) as part of the installer, taking you from bare metal to *WordPress*, for instance, in less than 10 minutes. Add to this the convenience of **apt-get install** for any other packages you need, and the vast amount of support available from any Ubuntu forum, and Ubuntu Server is a brilliant option for low-end boxes, public clouds and VPSs. But that's not the target demographic of Ubuntu Core.

Q What makes Ubuntu Core more suitable for installing on

virtual machines in the cloud than something like Ubuntu Server?

A It's only better in specific circumstances. Ubuntu Server would still be a great choice for your own machines, for example, but the cloud has opened up all kinds of potential uses that could never have been envisaged back in the old server days. Mostly this potential is thanks to scale, because if there's one thing that defines cloud services, it's their ability to scale quickly and transparently. This is primarily what differentiates the cloud from those old server networks, despite there being nothing particularly new in the technology that runs them. The cloud also creates new problems, such as security and automation.

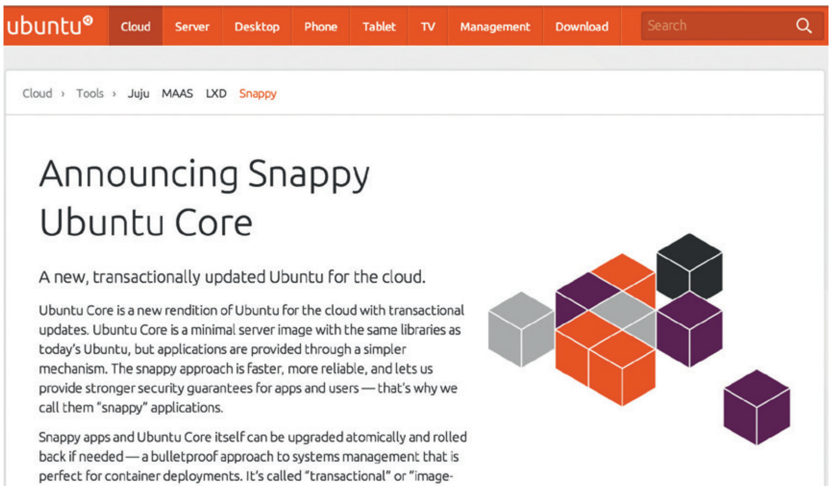
Q You mean when you're creating hundreds of new servers with a click of a button, you need some way of keeping them updated and secure?

A Exactly. Ubuntu Core is a good foundation for this because it's such a minimal distribution. As Mark Shuttleworth puts it, "It's completely extensible to all forms of container or service." But the minimalism also makes it much easier to secure and lock down. What's missing from the original Ubuntu Core is an equally scaleable and secure ability to update and install packages for these new environments, rather than through the traditional **apt-get** package management, and that's where the partnership with Snappy comes in.

Q Does that mean Snappy is the bit designed for the cloud?

A Not quite, although Canonical must have realised its potential for the cloud early on. Snappy was originally designed for its smartphone platform because Canonical needed to ensure carrier-grade secure updates and a way of separating the operating system from the applications that people will want to install. It's one thing if updating from Ubuntu 14.10 to 15.04 breaks your desktop, forcing you to re-install. It's quite another if an update breaks your Ubuntu Phone, and that's why Canonical has had to be so careful.

Snappy does this by keeping the operating system and the application files completely separate, and more importantly, both sets of files are 'read-only'. That makes it far easier for Canonical/carriers/cloud providers/administrators/you to validate the integrity of an installation. At run-time, the permissions are handled by AppArmor, the access control system that's been a part of Ubuntu for years (and is also the basis for the new phone application security). In that way, applications can be run within isolated containers where the OS is shared, but not the space where other applications may be running.



Announcing Snappy Ubuntu Core

A new, transactionally updated Ubuntu for the cloud.

Ubuntu Core is a new rendition of Ubuntu for the cloud with transactional updates. Ubuntu Core is a minimal server image with the same libraries as today's Ubuntu, but applications are provided through a simpler mechanism. The snappy approach is faster, more reliable, and lets us provide stronger security guarantees for apps and users — that's why we call them "snappy" applications.

Snappy apps and Ubuntu Core itself can be upgraded atomically and rolled back if needed — a bulletproof approach to systems management that is perfect for container deployments. It's called "transactional" or "image-

Canonical's announcement includes instructions on trying out Snappy for yourself.

Q How does Snappy manage installation if the operating system is read-only?

A To make this work, Snappy borrows a concept from 'Docker', the software abstraction container that's being used to deploy applications (collections of pre-configured packages) to the cloud. Installation and updates are applied as atomic, indivisible transactions that also allow you to roll forward and backwards through updates in the same way you might with a modern filesystem like btrfs. It accomplishes this trick by installing updates and applications as 'deltas' containing only the differences between the base installation or the previous package and the update.

Q Applications like *LibreOffice*, you mean?

A No. As we're in the cloud, we're talking about distributed networking applications. That's why the Docker framework is so important, because it opens Ubuntu Core to Docker applications immediately. Some of the most popular at the Docker hub include configurations running node.js, *WordPress* and databases, but equally, many people create their own bespoke setups to provide specific configurations and services for themselves or for their customers. Docker is becoming something of a standard for "shrink wrapping your applications and shipping them to the cloud", to paraphrase Mark Shuttleworth in his Snappy announcement. Ubuntu is popular among the Docker community, so it makes sense migrating the tested formula from phones and into the cloud, even if they're on opposite sides of the CPU scale.

Q As all of this is so cloud-based, how do us mortals get to play with the technology?

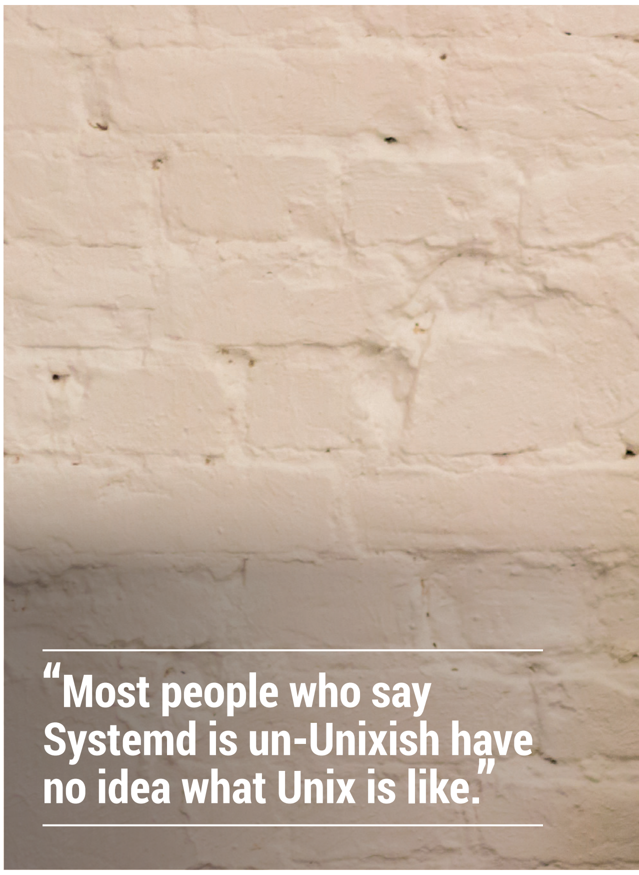
A At its most basic, you can try running Snappy on your local machine using **qemu-kvm** and a pre-made image downloadable from **ubuntu.com**. Instructions are also included for running an instance on Microsoft's Azure (one of the first platforms to declare its support for Canonical's new venture), as well as Google's Compute and Amazon's EC2. When everything is up and running, typing **sudo snappy install docker** will install the Docker package, for example, and you can search and update packages in exactly the same way you would with **apt-get**, with the addition of being able to roll back to previous versions. To try out Snappy Ubuntu Core, point your browser at: **www.ubuntu.com/cloud/tools/snappy**. LV

REBUILDING LINUX FROM THE GROUND UP

We meet **Lennart Poettering**, the lead developer of Systemd, an ambitious (and controversial) attempt to rewrite the Linux base system.

Few pieces of software in history have been so fiercely debated as Systemd. Initially a replacement for Sysvinit, the boot scripts that start up a Linux installation, Systemd has grown into a hugely powerful – and sometimes complex – replacement for the "bag of bits" that make up the Linux base system. It's growing all the time and now handles logging, device hotplugging events, networking, scheduled actions (like Cron) and much more. Almost every major Linux distribution has adopted Systemd, but there are still some unhappy campers out there, so Mike and Graham ventured to Berlin to meet Systemd's lead developer and get his view.

We also looked beyond Systemd, and Lennart's intriguing proposals for a new packaging system to make life easier for third-party application developers...


"Most people who say Systemd is un-Unixish have no idea what Unix is like."

LV Systemd has now been adopted by pretty much every major distribution, and yet whenever it gets mentioned in discussions on the web, flamewars erupt. What do you think are still the biggest misconceptions?

Lennart Poettering: There are many different misconceptions. Something you always see is the claim that Systemd is monolithic – and another is that it's not like Unix. The Unix misconception is a pretty interesting one, because most people who say Systemd is un-Unixish have no idea what Unix is actually like.

What's typical for Unix, for example, is that all the tools, the C library, the kernel, are all maintained in the same repository, right? And they're released in sync, have the same coding style, the same build infrastructure, the same release cycles – everything's the same. So you get the entire central part of the operating system like that. If people claim that, because we stick a lot of things into the Systemd repository, then it's un-Unixish, then it's absolutely the opposite. It's more Unix-ish than Linux ever was!

The Linux model is the one where you have everything split up, and have different maintainers, different coding styles, different release cycles, different maintenance statuses. Much of the Linux userspace used to be pretty badly maintained, if at all. You had completely different styles, the commands worked differently – in the most superficial level, some used **-h** for help, and others **--h**. It's not uniform.

If we put a lot of the glue in one repository, it's not all the way towards Unix, but it's half way between traditional Linux and traditional Unix. We do not put libc and the kernel in the same repository, just the basic things. So that's a misconception that I'm always bemused about, and I'm pretty sure that most people who claim that have never actually played around with Unix at all.

LV Another issue is: some people see Systemd presented as an init system replacement, but now it's doing X, Y and Z on top. You've said it's about replacing a "bag of bits" with an integrated suite of tools. When you started Systemd, was it a case of Red Hat saying to you, "we want a new init system", or...

LP: No, it was actually the opposite. Back in the day, when we started working on Systemd, many of us who worked on the lower levels of the operating system realised that Sysvinit was not going to be the future. And then I was playing around with writing my own init system, which had the funny name Babykit, and that was 10 years ago or something. And then Canonical's Scott James Remnant started working on a new init system called Upstart. He made it public, and I stopped working on Babykit.

We, at that time, thought: OK, Upstart is the future! Scott understood how init systems work – it needs to be dynamic, it needs to react to events, and it's not the static thing that Sysvinit was. So we thought that was the way of the future, but as it progressed, we realised it probably wasn't the future, because we realised that conceptually, it was the wrong design.

The way Upstart worked is that, as a programmer or admin, you write: if A happens to B, or X happens to Y, do a certain thing. But we believed that an init system should work the other way around, where you say: this is where I want to go to, and you figure out the rest. Because of that design, Upstart was very simple, but it put a lot of complexity on admins and developers,

because you actually had to write down all these rules. It wasn't the computer that figured out what to do.

We thought: if you want to solve this properly, then you need to let the computer do these things. And this had lots of different effects: for example, Upstart always maximised what happened on the system, while we think you always have to minimise what happens. And the reason for that was simply because, if you specify exactly what state you want to end up in, you can pull in all the dependencies recursively and boot to exactly that.

"We started writing Systemd, and Red Hat didn't like it at all. So I worked in my free time."

The Upstart way is always, "if this is started, then start that". If the network is up, you take that as a trigger to start NFS and things like that. It always has this effect that you start as much as possible instead of as little as possible.

So anyway, long story short, we came to the conclusion that Upstart is conceptually wrong, and it moved at glacial speeds. It also had the problem that Canonical tried very hard to stay in control of it. They made sure, with copyright assignment, that they made it really hard to contribute, but that's what Linux actually lives off. You get these drive-by patches, as I would call them, where people see that something is broken, or something could be improved. They do a Git checkout, do one change, send you it and forget about it.

LV And you never see them again!

LP: Yeah, and this is great – these are the people you want to have, because the vast majority of patches are actually of that kind. It gives you this polishing that you want. The people invested in the project all the time do the big things, and don't care so much about the polishing. So these kind of patches are what you want. But if you do these copyright assignment things, you will never get those people because they would have to sign a contract before they can send you something.

Putting it all together, we realised that Upstart wouldn't be it. So at one Linux Plumbers Conference, four years ago or so, Kay Sievers and I said that we should do something about it, after we saw at the conference how Upstart wasn't moving ahead. And then we started working on it, pulled out the old Babykit code, gave it a new name, and started proposing it.

A lot of people understood that this was the better approach. It was a lot more complex than Upstart – to make it clear, I think Upstart actually has its benefits. The source code is very, very nice, and it's very simple, but I think it's too simple. It doesn't have this engine that can figure out what the computer is supposed to be doing.

So we started writing Systemd, and Red Hat didn't like it at all. Red Hat management said: no, we're going for Upstart, don't work on that. So I said, OK, I'll work on it in my free time. Eventually Red Hat realised that the problems we solved with Systemd were relevant, and were problems that needed to be solved, and that you couldn't ignore them.

Then we convinced the Fedora Technical Committee to adopt it, and then Red Hat internal management accepted it for RHEL, and we managed to convince every committee that mattered, bit by bit. It was absolutely not that Red Hat told us to work on it – we had to convince them.

LV I don't think many people know that!

LP: This is something that people in general don't know. They assume that Red Hat is this one entity, that has one opinion and pushes one thing. It's really not like that. The people who work at Red Hat, the engineers, they come from the community – they first become famous in the community, they hack on things, do good stuff, and then Red Hat comes along and says, "Hey, do you want to work for us?"

And when you start working for Red Hat, they don't check your opinions at the door. You can be sure that if there are multiple opinions on one topic in the broader community, the very same opinions inside Red Hat exist too. Inside of Red Hat there are discussions. Red Hat has many different people, and most of them have strong opinions and convictions.

LV And much of this debate happens in public, on public mailing lists. Then you have some people saying that all this arguing

looks bad, compared to how Microsoft or Apple does things. But I bet they all have the same arguments, just as passionately.

LP: I'm absolutely sure. There was this time when the people working on *Microsoft Word* had their own compiler to build *Word* and the rest of *MS Office* with. Microsoft had the Visual Studio group, and the Office group, and they had their own individual compilers. That's just crazy of course.

So I don't think that Red Hat is different from anywhere else; except that at Red Hat, because people are working on open source, they have much greater attachment to their code. So they have even stronger opinions.

If back at the start of Systemd, you and the other developers had explicitly said: "We're going to replace a lot of the base system", do you think it would've been better received? Some people see it as an init system that's suddenly touching everything else.

LP: Initially it was an init system – it was just PID 1. We knew from the very beginning what we were getting ourselves into. We knew very well that touching something that has so much history, that is so close to what admins do all day... That changing it would be a massive problem. So we knew that people would hate us for it. We knew we'd have to fight for a long time to get it accepted.

We eventually realised that doing just the init system would never be a complete solution. Because if you do an init system but still invoke all the shell scripts and all the other things needed to bring up the system, you've only solved part of the problem. You've solved one thing but not 90% of the problem. So we slowly started doing stuff that all the other Linux distros did, and implemented that in simple C code that was fast and parallelised.

Debian had its init scripts, and Fedora had its init scripts, and they all kind of did the same thing, and did it differently, and some are better, and some are worse. We thought OK: this is bullshit, let's write this in C in a unified way, and try to pick the best features of all distributions and make a convincing argument that it's the right way.

So it initially grew. But something to realise is that there's very little in Systemd that's actually required. Systemd requires Journald, because every single service that runs on the system is connected to Journald, and we need some way to log things during early boot. So Journald is a requirement, and Udev is a requirement. But pretty much all other components are completely optional. If you don't want to use the way Systemd loads kernel modules from a static list, then you can absolutely replace it.

Or if you don't want to use some of the more modern components like Networkd, then use something else. I mean, on my laptop I even use NetworkManager, because Networkd doesn't do wireless, right? Networkd is more for containers and servers. So if you want to adopt Systemd, you can absolutely adopt the baseline, which is the three components that I mentioned. You can keep the rest of the system – however, our implementation of the individual parts is usually pretty convincing, and usually people then replace more.

"Most people at Canonical didn't even realise that they had commit access Systemd."

Some people see it as a requirement for Gnome...

LP: But it's not actually a requirement. Some people don't realise that when Gnome started making use of Logind, I actually wrote the patch for that. I ported *GDM* onto Logind. But when it did that, I was very careful to make sure it would still run on ConsoleKit. I didn't

You need a thick skin to hack in open source code sometimes, especially if half of the world seems against you.

Lennart lives in Berlin, and knows where to get great Vietnamese grub.

want to have those fights – if people want to continue running ConsoleKit, they can. Those patches made it in, but some people saw that Gnome now works with Logind, hence it must not work with ConsoleKit any more!

But that's actually not true. And to my knowledge the code is still in there – the compatibility for ConsoleKit. The Gnome team has the general problem though, that nobody's willing to maintain it. People who want to stick to the old stuff, they actually need to do some work on it. If they don't, then it will bit-rot and go away.

So anyway, we tried to do these things in the nicest possible way, but of course people generally don't acknowledge it!

LV A lot of people just think there's only Red Hat working on Systemd.
LP: Oh yeah, we're a lot of people now. Yesterday we had 26 committers, and 40 people contributing code every month or so. The committers group is quite diverse, and for us it's quite an exercise in making the diversity of the community be reflected in the diversity of the people who work on it. This is also related to how Upstart worked: Upstart was very locked-down, and Canonical always wanted to stay in control of everything. For us it was an exercise to make sure this doesn't happen. We're not the ones in power – the community is.

So of those 26 committers, there's a good chunk working for Red Hat right now, but there are people from Intel, Canonical... We had people from Canonical in the committers group, all the time during the discussion about whether they should even adopt Systemd. Most people at Canonical didn't even realise that they had commit access to these things.

There are also developers from Debian – two or three of them.

LV There should be a Systemd foundation!
LP: [Laughs] Well, we don't want to make it too formal. We have this speed, this quick pace with how we progress Systemd, and I think it can only work if we stay somewhat loose and not have strict regulations about how these things work.

But we try to make sure that it's inclusive. We have people from Arch Linux, people from all the Linux distributions, big companies that do open source. We want to make sure it stays that way.

LV Why do you think some distributions managed to adopt Systemd without any major fights, and then others like Debian had very intense debates and resignations? Is it just because it's a distro with more political processes?
LP: Arch Linux probably did it the quickest way. You know, distributions attract different kinds of people, of course. If you looked at Arch Linux, it attracted very progressive kinds of people – like power users. They're progressive and want to make the best out of their computers. So it was easy for them to adopt.

Then if you look at Gentoo, for example, they still haven't done Systemd as default. They used to be like Arch Linux is now – they used to be the young people who adopted things quickly. But the Gentoo people aged, and they became more conservative.

And Debian is probably an even more conservative bunch. Debian is a really old project, and many people from back in the old days are still active on it. So they have longer release cycles. And Fedora always defined itself as being on the bleeding edge, of course, so it was easier. Well, not that easy – some people don't realise that inside of Fedora and inside of Red Hat, there

After Avahi, PulseAudio and Systemd, we're intrigued to see what Lennart tackles next…

were lots of fights. So it's to do with the culture around the various distributions. And Slackware are the ultra conservatives!

LV Do you read the comments when Systemd is being discussed on the net? Do you despair when it all turns into hatred and flamewars?

LP: For some reason it doesn't touch me too much. I try to keep an open mind and figure out what people actually think. There's a lot of noise out there, but usually there's some core of an argument – something that we should actually be aware of. So if people are annoyed by Systemd, usually they ran into some kind of bug or something. It might not necessarily be a Systemd bug, but we need to take it seriously.

Nowadays Systemd is very polished in many ways, and the reason why it is so polished is because we actually listen to people. Sometimes people say we don't listen – we do, but we just don't always agree. If we would just stick our heads in the sand and not care at all what people wrote, Systemd would certainly not be what it is, or have found the adoption that it has.

LV You've said yourself that the flamewars could dissuade potential open source contributors from getting involved.

LP: I'm in the lucky position in that there's no pressure on me in any way. I know that a lot of people have pressures that they live under, and if you also get pressure from the internet over some things that you do in your free time, because you love it, that is very disappointing for them. So I have a luxury, and I know it, and I can only feel for people where it's not like that.

I know a lot of people who've had enough of open source, and who will not participate in the communities where things get really bad. And that's a big loss for open source.

LV Something else we wanted to talk about is your proposal for packaging. What's that about?

LP: It's really about augmenting the Linux platform with a new way to package applications. It's not about simplifying things or changing things – it's adding something to the ecosystem that we were missing so far. There are lots of people working on that in different areas.

If you look at all the operating systems that are popular these days, like Android, Mac OS, iOS, Windows Metro, they always have really strong app platforms, where they provide a sandbox, a nice way to distribute apps and ship updates. On Linux we don't have anything that's as convincing. We don't have a common way to sandbox stuff, and the way that we ship stuff is with Deb packages, or RPMs.

It's madness for third-party app developers, to develop for Linux. Like, what do they develop against – which distribution? And if they make that decision about which distributions they wanted to support, it'd be quite a few, and then there are lots of versions. You might want to support Fedora 20, 21, and 22, and then OpenSUSE and its various versions, and Ubuntu and its various versions… All of those distributions bring different libraries.

So the test matrix, the combinations of software you have to test your apps with, grows incredibly. That's not something that's digestible for third-party app developers. And it's really hard. The only way you can really deal with that is to get your stuff into the distributions. Then the distributions will do all the work for you – they'll rebuild for you, test for you and things like that.

LV But for upstream developers it's hard to get new releases into distributions quickly.

LP: Exactly – you're bound to what the distributors do, to the lifecycles of the distributions, the release cycles. You're not responsible any more for your software – you've passed it on to the distributions. Which in many cases is actually a good thing, but in general it's not what third-party developers want. If you look at how *Firefox*, for example, packages its Linux version, it's a tarball that installs in its own directory.

The classic Linux distribution model, where you get everything from the distribution nicely packaged up, vetted for security problems, with security updates – that's a fantastic thing. But I also think it leaves out all these third-party people, and if we want to grow the Linux ecosystem beyond what we already have, into something where it's actually useful for a broad number of people, where we have markets and more apps, we need to provide a way to make these apps digestible.

And by digestible I mean: we need to have good sandboxes. If you don't get your software from the distribution any more but directly from the developers of the software, then you have the problem that you can't trust the code as much. Distributions add a bit of trust, in that they look at the code before packaging it, and then you get a nice stamp on it: this is good software. And then you only have to trust the distribution, and not trust 100 different software vendors any more.

Now, if we open this up and make it typical that you install one distribution and then 100 different apps from 100 different vendors, we need to do something about this trust problem. So that's why we need sandboxing. We need to reduce the chance that badly behaving software can destroy your data.

LV OK, but how do you deal with the niggling little differences between distributions, where everything has slightly different library versions, filesystem locations?

LP: Our idea is to introduce something called runtimes. When third-party developers develop their stuff, they should be able to do so against one very fixed set of libraries, in very specific versions, compiled in a very specific way. They can test their stuff with that – and then, if they're sure that everything works, they can check it off, and when it's installed on the final machine, it knows exactly the combination of software that it runs against. Instead of some weird combination of software local to that machine, where they have real trouble making sure it works.

Third-party developers don't want to do all the support for someone who says: "Yeah, but I have this old version of libc and it doesn't work", or "I have this really custom distro I compiled myself". So we have this concept we call a runtime, which is basically just a set of libraries with very specific versions. The idea is that you can install multiple runtimes at the same time. And then, if you have apps that require different runtimes with different versions, they'll run against their specific runtime and everything's good.

"Sometimes people say we don't listen – we do, we just don't always agree."

After our interview, Lennart pointed us in the direction of the East Side Gallery for some Berlin Wall exploring fun.

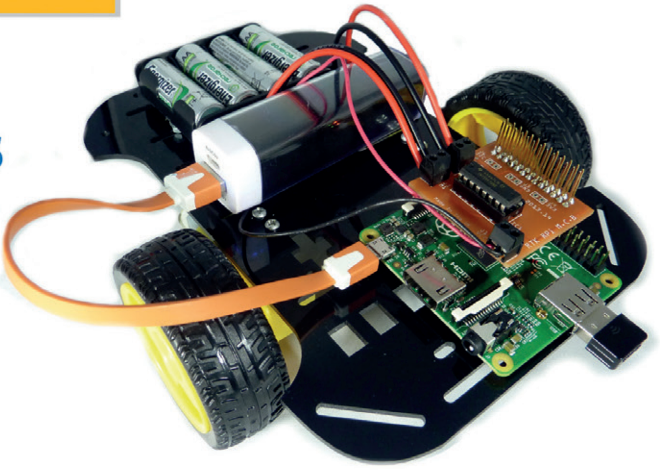
LV Didn't the Linux Standards Base try to do something similar for third-party devs?

LP: What the LSB did there was to standardise a set of libraries, but didn't define any specific versions. And classic Linux distributions only allowed installation of one set of libraries at a time – you could have one libc, and one OpenSSL. So LSB tried to make the best out of the traditional Linux model, but that's not enough.

There's a scheme that we put together that's not unlike what Android has. For example, if you develop an Android app, you do so focusing on one specific runtime, right? It's one that Google defines, and if there's something that's not in this huge runtime, then you have to ship it yourself inside of the app, and everything is good. And the phones have a couple of runtimes for the different versions, and then you pick one of the versions you want to develop against – usually the newest version, or maybe an older one. So we kind of want to adopt the same scheme, but make it more pluralistic. In Linux everything is pluralistic.

Ryanteck LTD

Raspberry Pi Robotics Kits Starting From £24.99



Available Pre-Soldered & With Raspberry Pi A+

Buy Today At http://store.ryanteck.uk
And use LINUXVOICE for 5% off your order!

5% Off Code Expires 31st of Jan 2015

BUY LINUXVOICE MUGS AND T-SHIRTS!

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

shop.linuxvoice.com

LINUX VOICE REVIEWS

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

Andrew Gregory
Imagination piqued after Pi Wars, Andrew needs to get his hands on some servos, fast.

I've just written, and then deleted, an angry rant about the EU's imposition of a new piece of tax law on its subjects. In a nutshell, to make it harder for Amazon to dodge tax by basing themselves in Luxembourg, all businesses supplying digital products now have to pay sales tax in the territory where the customer is based, rather than where the business is based. This sounds fair enough, but because of the way it's being implemented, businesses that sell online now have to collect two independently verifiable proofs that the customer lives where they say they live (otherwise we could pretend that all of our digital subscribers live in Monaco, and spend the saved tax on cigars).

Help me, Obi-Wan!

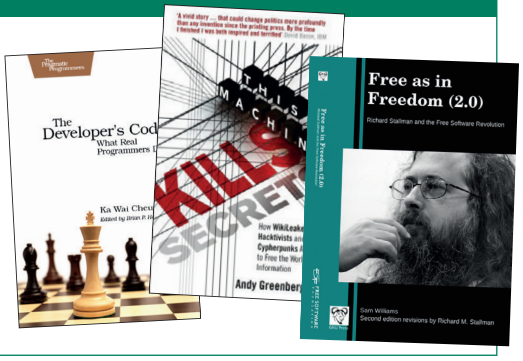
There's an obvious guilty-until-proved-innocent aspect to this, but more importantly, it means that any customer buying online from an EU business has to supply more information about themselves than if they were to buy the same product from outside the EU. It's a needless inconvenience that will inevitably drive business away from the EU. Good news though: the system accepts ODF – after years of heel-dragging, we can now interact with our government using an open format. Every cloud...

On test this issue...

BOOKS AND GROUP TEST

If you've helped a relative install Linux over the Christmas holidays, you've probably had loads of phone calls in the last few weeks asking for help fixing some vague problem or another. Well, if you install one of the remote desktop setups from this month's Group Test, you won't have to rely on vague 'It doesn't work'-style error messages – you'll be able to log in and fix the problem as easily as you would on your own machine. Linux is magic!

Also magic is the written word, and we've a great selection of brain-enhancers for you on page 56.


The Developer's Code
What Real Programmers
Ka Wai Cheung
This Machine Kills Secrets
Andy Greenberg
Free as in Freedom (2.0)
Richard Stallman and the Free Software Revolution
Sam Williams
Second edition revisions by Richard M. Stallman

USB DAC 128 Mk II from Henry Audio

Graham Morrison reviews a high-end audio output device without resorting to words like 'air', 'gravity' and 'pea soup'.

DATA

Web
www.henryaudio.com
Developer
Børge Strand-Bergesen
Price
€260.00 (£200 approx.)

A good tune is still a good tune on AM radio, or through an MP3 player, or from the front seat of a car while the kids are screaming for One Direction in the back. Music is about enjoyment, and it can be enjoyed in virtually any environment. But we're also certain that if you spend a lot of time listening to music, improvements in playback quality will increase your levels of enjoyment. It may come from moving from AM to FM radio, or mono playback to stereo, or from taking time out of a busy day to listen to some music on your own. But it can also come from improving the way your digital music is converted into audio. This is the job of a DAC, the digital-to-analogue converter, and they're everywhere. From the headphone and speaker outputs on your smartphone, to your games console or DVD player to your car or amplifier. They perform an important job, and despite DACs having been consumer products since the dawn of the compact disc there's still enormous variation in their quality and capabilities.

"There was more stereo width and clarity, and a much stronger low-frequency bass response ."

DACs come in all shapes and sizes, with some costing thousands of pounds. It's a hardware category occupied by audiophiles and enthusiasts, and more recently, hackers. This is where the USB DAC 128 Mk II belongs. It's an affordable DAC, in audiophile terms, with reportedly exceptional quality output for the price, and with strong links to both Linux and open source. The controller inside its small aluminium case is an Atmel AVR32 general-purpose MCU, running C code that's open source, and the whole project is a descendent of the Audio Widget, a DIY DAC based on similar hardware, software and specification. The manual even includes the schematics.

Børge Strand-Bergesen, the brains behind Henry Audio, has been a long-time contributor to this community, and he's an audio geek who's been playing with converters for decades. You only need to look at his recent blog post (**http://www.henryaudio.com/blog.php**) on the difficulties of filming wagon wheels in motion to understand something of his dedication to taming streams of sound.

The PC requirements for high definition audio can be quite demanding, and the specs require a dual-core PC. However, we had good results from a relatively lowly quad-core ARM Cortex-A9 at 1.0GHz and even a Raspberry Pi.

Living in a box

The box itself feels small but substantial (it's 114.4mm wide by 32.8mm high with a depth of 128mm). There are two halves to the metal surround, and the top half can easily be removed with a small hex/Allen key, giving access to the cleanly designed circuit board with its internal headers and potential for modification. There's a single dual-colour LED on the front, alongside a nicely rendered logo, and the rear panel houses two momentary switches, the audio connectors and a mini USB port.

Power comes in through the USB, so there's no external PSU to worry about, although the USB cable isn't included in the package. When connected to your PC, the USB DAC 128 Mk II appears as a standard class compliant USB audio output device, requiring no drivers nor any further configuration. There are no inputs, and outputs are available as both digital and analogue when listed from your software's audio mixer. Digital co-axial output is provided as a convenience rather than a feature, because it bypasses the DAC, but the same gold-plated RCA connectors are used for both functions.

We first connected ours to an Arch laptop and an amplifier, and yes, you need an amplifier. When we spoke to Børge and asked why the unit couldn't incorporate a headphone output, his response was that building a quality amplifier for a pair of headphones would make the unit dramatically more expensive. We know from shopping for similar products ourselves that he's got a point, but it diminishes the portability of the device when you've got to carry around two units.

As well as our Arch machine we also connected the unit to a Raspberry Pi and an ARM-based TBS 2910 Mini PC to see whether the DAC could be used as the output for a media centre.

The USB DAC 128 Mk II operates as both a class 1 and class 2 USB audio interface, and the colour of the LED indicates which mode the unit is currently using – bright green for class 1 and a more subdued orangey red for class 2. The difference in capabilities of the two modes is dramatic. Class 1 seems to be provided purely for compatibility with older Windows hardware that doesn't include class 2 drivers by default. Its maximum output resolution is 24-bit audio at 48kHz – or a little higher than CD quality. Class 2 output is capable of 32-bit audio delivered at 192kHz. The two modes are toggled manually using a slightly unintuitive button sequence on the back of the unit, but we had no problems using only class 2 from Linux when the correct mode was enabled.

This isn't an audio magazine. And we're all too aware of the mire of comments that accompany many Hi-Fi magazines/websites trying to describe something as subjective as audio quality – we still remember the gushing Hi-Fi review of an extortionately priced SATA cable for a music server. But reviewing an audio interface without giving an opinion on the sound quality would be remiss, and we do understand some of the technical challenges in creating great audio.

Quality in digital audio output is the result of absolute rock solid timing, and we'd even argue this is more important than resolution. Most USB audio interfaces, for example, rely on the PC for both the clock and the management of the data stream (known as a synchronous protocol). Any drift in timing affects how the DAC interpolates between one sample and the next, resulting in less audible clarity. (there's an analogous problem in video when a framerate becomes jittery.)

The clock and the crystal oscillators that drive the Henry Audio unit are of exceptional quality, and are fundamental to its sound. So too is the DAC chip itself, an Asahi Kasei AKM4430, and the asynchronous USB driver that Børge has spent a long time developing. This means the interface talks back to the PC rather than simply processing everything sent across the cable – an asynchronous protocol.

We are the music makers

We sat down to listen with some trepidation. Our first choice of music was something we're very familiar with, the album *Exai* by Autechre. To most, it will sound like an electronic cacophony of high-pitched squeals and noise, but it's one of the author's favourites and we'd imagine a good test for any converter. Best of all, we own the album as both vinyl and high-resolution 24 bit FLAC files, so we could test the DAC against an analogue input from the record player, the DAC on a (modest) Denon amplifier and the 128 Mk II. The difference between the amplifier's DAC and that in the USB 128 Mk II was so huge we had to check we'd not connected the source correctly. The amp was being driven by an HDMI output from the laptop, but there was no comparison with the output produced by the USB DAC. Without resorting to the sycophantic lexicon of audiophile journalists, there was undoubtedly more stereo width and clarity, and a much stronger low-frequency bass response with the 128 Mk II – a huge improvement over the Denon's built-in DAC and marginally better than the vinyl.

The schematics and the controller software are open source, and there's an active community of hackers building better power supplies and converters for the Audio Widget baselines designs.

Living in a box

Since then, we've spent a considerable amount of time listening to the DAC, and not just scary electronic music. We tested 24bit/192kHz playback with Mozart's Violin Concerto, where we found the differences more subtle, and we listened to heavily compressed pop, where the DAC added much more dynamic life to the recording. Everything we played sounded better, sometimes subtly and sometimes dramatically. And while we came into this review having absolutely no intention of spending £200 on something as prosaic as a converter, we're left convinced that it's worth the money.

If you're an audio hacker who wants a quality DAC to tinker with, we know of no better device. If you're a Linux user wanting great quality audio output, and you've got the music collection and the amplifier to match the investment, it's worth it. However, if you're still listening to music with your free smartphone headphones, you're better off investing in a better pair of headphones.

LINUX VOICE VERDICT

It may seem like a luxury, but if you listen to anything other than MP3 and you've got a decent amplifier, it's definitely worth the investment.

★★★★★

Fedora 21

This "game changer" of a release is provided in three new versions. **Mike Saunders** checks them out.

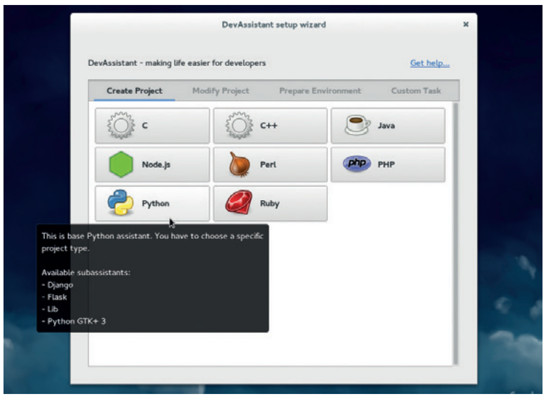
DATA

Web
www.fedoraproject.org
Developer
Red Hat and community
Licence
Free/open source licences

The Fedora Project is no stranger to reinvention. It started off as a community-supported successor to Red Hat Linux, as that particular distribution started to focus on the enterprise market, and most recently has been gaining ground as a bleeding-edge test-bed for upcoming Red Hat Enterprise Linux products.

With Fedora 21, the distribution has been split into three sub-projects: Cloud, Server and Workstation. These are flavours of the same distribution, with different focuses, but all based on the same core components: kernel 3.16, *glibc 2.20*, *Yum 3.4.3* and *Systemd 216*. The Cloud version provides a minimal environment that's designed to run Docker containers, and includes work from Project Atomic, making it easy to roll back updates should they cause problems. The Server flavour, meanwhile, includes the *Cockpit* management interface, *Rolekit* for specifying server roles, and the *FreeIPA* identity management tools.

We spent most of our time with the Workstation variant, which runs Gnome 3.14 on top of *X.org* server 1.16.1. Wayland is available, but only as a technology preview; we're still a while away from seeing it replace X by default. So without big changes like Wayland, what's in here to make it worth the upgrade from Fedora 20, or just trying the distro if you use something else? Well, the installation process has been simplified, bringing it down to a handful of clicks. After choosing your language, *Anaconda* (the Fedora installer) will try to auto-detect as much as possible, and present a recommended disk layout, so you can get the distro onto your hard drive with very little effort. After installation, the Software package manager is also worth checking out, as it has seen GUI refinements too.



This is base Python assistant. You have to choose a specific project type.

Available subassistants:
- Django
- Flask
- Lib
- Python GTK+ 3

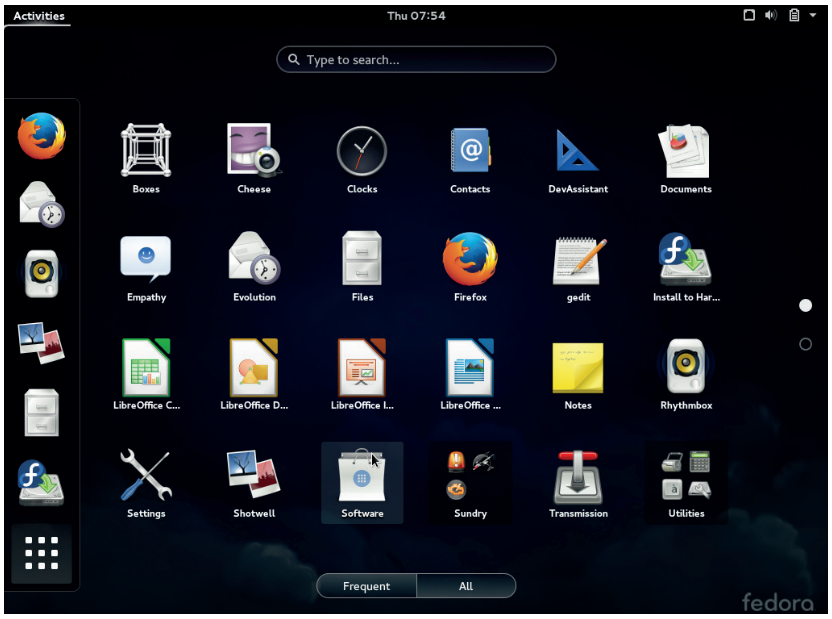
The *DevAssistant* is a great little addition for this release, making it easy to set up new coding projects.

Gnome Terminal sees the reintroduction of transparent backgrounds, after plenty of rage from users when this feature was removed in an earlier release. Much work has been put into making the distro look better on HiDPI displays (like on the MacBook Pro and some Chromebooks), although once you start running non-Gnome apps, the results are a mixed bag.

Identity crisis

But while the Workstation flavour continues Fedora's reputation as a cutting-edge desktop distro, we're left wondering about the long-term goals of the Server and Cloud flavours. Why would you run Fedora on a server? If you don't mind major upgrades every year that could break things and require re-learning new tools, Fedora Server might make a decent server OS. Or if you just want to see what's coming in future CentOS and RHEL releases, it's a great way to try out new technologies. However, its support lifecycle is a paltry 13 months – we suspect most Fedora users would rather use CentOS (CentOS 7, for instance, will receive security updates until 2024!).

Still, on the desktop it's still one of our favourite distributions, bringing new ideas to the table without being too bleeding-edge to be usable. The next few releases are likely to be more ambitious though, especially with Wayland.



Gnome 3.14 is the default desktop, but there are 'spins' with alternatives such as KDE and Xfce.

LINUX VOICE VERDICT

A solid, if not especially exciting, release from the Fedora team with lots of welcome refinements.

★★★★☆

Seafile 4

Fed up with being spied on by GCHQ, the CIA and Google, **Ben Everard** is now looking for a server to run his own cloud storage.

Cloud storage is a wonderful idea. However, it comes at a price – when you upload your data to someone else's server, you lose control of it. As soon as it leaves your machine, it's not really your data any more. There is a solution to this – run your own cloud storage.

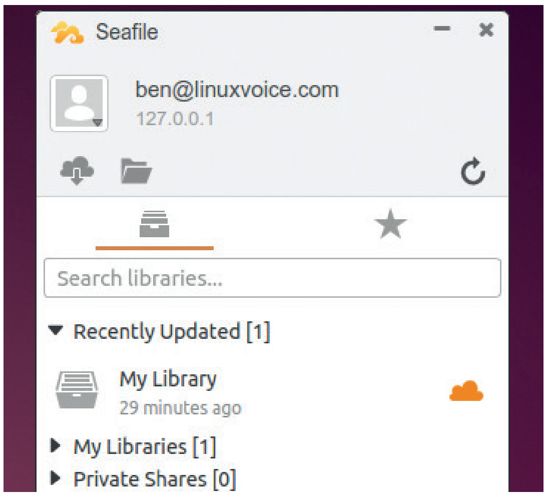
Seafile is an open source cloud storage platform. It comes in three parts: the client, the server and *Seahub*. *Seahub* is the web interface that enables you to manage users, and upload and download files without the client software installed. The client software synchronises directories on your computer with libraries stored on the server, and the server software provides all the functionality to make the client and *Seahub* work.

Running your own cloud storage can be more secure, especially as most cloud storage providers have been shown to share data with government spies. To help keep everything safe, *Seafile* provides the option to encrypt your data on the client side. This means that it's secured before it leaves your machine, so even someone with complete server access wouldn't be able to decrypt it. This is especially useful if you're using a *Seafile* server someone else is running.

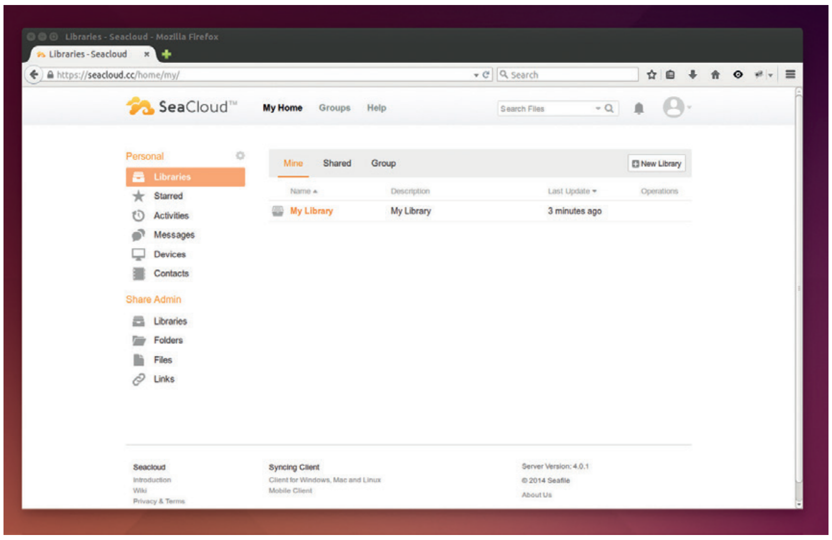
Ease of use

The client software is easy to use, and works well. You can add accounts from more than one server, and it will sync the required libraries with your machine.

The server provides all the features you would expect from cloud storage. You can share files and libraries with other users, with groups of users, and with the public (via an HTML link) easily. It keeps a history, so you can always undo any changes. There's also messaging between users, but this feels a bit like



The *Qt* app looks good, but doesn't fit in with our desktop



an afterthought, and we can't see any advantages of this is over email (every login is an email address).

The *Seahub* web interface provides you with full access to your libraries without needing to install any client software. It also enables you to preview and edit some filetypes, although these are quite limited, especially editing, which is only text, markdown, and seaf formats. The latter of these is a rich text format developed specifically for *Seafile*, so isn't widely supported by other software.

There are client packages for Ubuntu and spinoffs on the *Seafile* website. For other distros, you'll need to check your package manager, or install from source (**https://github.com/haiwen/seafile-client**).

Installing the server just requires you to download and unzip a tarball, then run a shell script that configures the environment. By default, *Seahub* runs on the *Gunicorn* server, so if you're already using port 80 for a web server, you'll need run it on a different port (it defaults to 8000). It is possible to run it using *Apache* or *Nginx*, though the setup is a bit more involved. The documentation covers everything you need to know: **http://manual.seafile.com**.

Seafile should run on just about any box including cheap VPSes. There's even a version designed specifically for the Raspberry Pi. Provided you just want file-syncing and cloud storage, *Seafile* is easy to install and run, and gives you all you need. LV

If you don't want to run your own server, you can get an account on **www.seafile.cc**. Free accounts come with 1GB, but this can be increased to 100GB for a $10 monthly fee.

DATA

Web
www.seafile.com
Developer
Haiwen
Price
Free under Apache licence or from €12 per user

LINUX VOICE VERDICT

Seafile is excellent for cloud storage, but its messaging and editing capabilities are weak.

★★★★☆

Kodi 14.0 Helix

It's new beginnings for XBMC, as **Graham Morrison** tests the first version released under its new name.

DATA

Web
kodi.tv
Developer
Team Kodi
Licence
GPLv2

Kodi is the media player/platform that used to be know as *XBMC*. It's best run on a computer connected to a television where it can manage, serve and play movies, music, photos and more from its full screen mile-wide interface. Through a comprehensive plugin repository, it can orchestrate television recordings, play YouTube videos and enable you to catch up on BBC programmes, and many other services. It's been developed for over 10 years, and this is the second major release of the last 12 months, following on from the last XBMC release in May 2014.

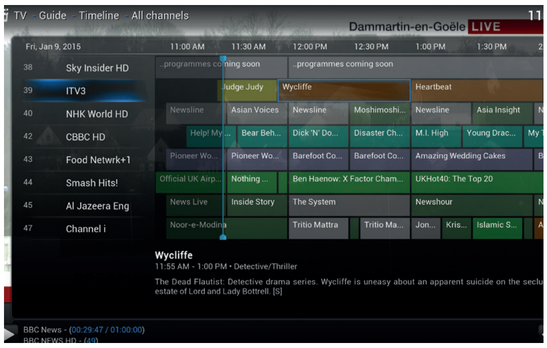
For years, the overall quality of the application has been a shining example of what open source can achieve. The user interface is fast and polished, with the application always making best use of whatever environment it finds itself in. We really don't know how the development team manages to create so many regular and significant updates while maintaining the quality.

From TV to T. Rex

Part of this release is a renaming exercise, as everything from an old installation will be moved over to the new one. That may make reverting to the previous version difficult if you want to keep your settings. We first installed the latest edition of the *TVHeadEnd* plugin to test the new PVR functionality, and we experienced almost no stability problems.

"Kodi always performs exceptionally well, and we've noticed a significant upgrade with version 14."

There seems to be a speed increase in the way *Kodi* scans your network for media shares, as well as grabbing the programme guide. In the process of writing the OpenELEC tutorial (see page 84), we've spent quite a bit of time with the latest release,



Wycliffe
11:55 AM - 1:00 PM • Detective/Thriller
The Dead Flautist: Detective drama series. Wycliffe is uneasy about an apparent suicide on the seclu
estate of Lord and Lady Bottrell. [S]

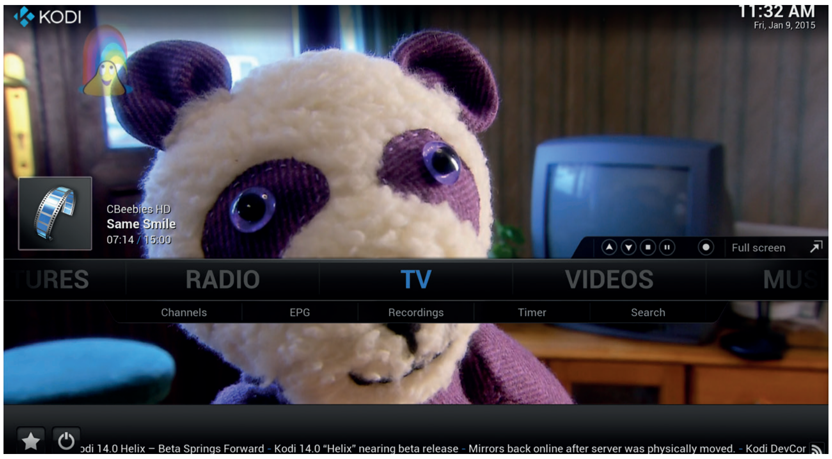
Kodi gives you a clear, intuitive interface from which to choose from the sea of rubbish clogging your telly.

installed on a PC, on a Raspberry Pi, on a quad-core ARM-based platform and a Nexus 5 Android phone. As long as the hardware can take advantage of some form of hardware acceleration, *Kodi* always performs exceptionally well, and we've noticed a significant performance upgrade with version 14, especially on those less powerful platforms. This is likely to be a result of significant library, codec and hardware acceleration upgrades for all kinds of platforms. There's cutting-edge support for the latest *FFmpeg* packages, for example, bringing compatibility with the shiny new H.265 codec, an even more efficient algorithm that promises to halve your file sizes for the same quality.

The only problem is that there's no support for hardware acceleration, which means it was only our PC (and not the smaller mcahines on which we tested we tested *Kodi*) that was able to decode a full HD movie encoded with the new codec. There's also new support for binary codecs, which may enable proprietary media playback at some point. On the plus side, decoding now defaults to being multithreaded. As usual, the more power you throw at *Kodi*, the better it will feel, and the Raspberry Pi in particular needs a lot of patience, especially if you install any of the PVR plugins.

This is one of those strong consolidation releases that shoudn't be underestimated. *Kodi* works brilliantly and looks awesome and this release only strengthens its position. Considering everything that has had to be changed to accommodate the new name, we think it's a huge success.

The *Kodi* team has had help from Intel to squash a firmware bug that caused random crashes after a period of time.



odi 14.0 Helix – Beta Springs Forward - Kodi 14.0 "Helix" nearing beta release - Mirrors back online after server was physically moved. - Kodi DevCor

LINUX VOICE VERDICT

An unrivalled media player that begs to be installed on an embedded system underneath your television.

★★★★☆

VMware Player 7

Ben Everard doesn't need real computers any more. He's moved to the cloud and runs everything virtually.

Virtualisation software gives you the power to run many virtual machines (VMs) on a single computer. Each of these machines can have a different OS, and will run completely independently from the host machine. Here at Linux Voice, we use VMs extensively for testing out different distros, and to run our web server.

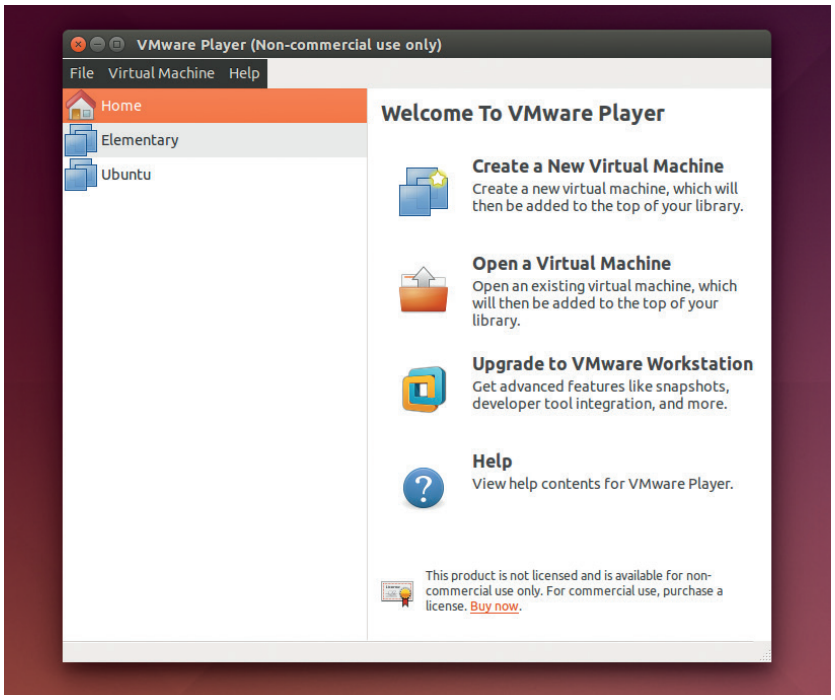
There are a few different pieces of software available for virtualisation, some open source (like *VirtualBox*) and some) are proprietary (like *VMware Player*). Since it's proprietary, you won't find *Player* in your distro's repository. Instead, you need to download and execute (as root) the bundle file from **www.vmware.com/products/player/**.

Version 7 brings support for more RAM (up to 64GB), more CPU cores (up to 16) and more video memory (up to 2GB). It also brings the advantages for Intel's latest Haswell chips. If you're already running *VMware Player 6* and don't need any of these, version 7 may not be worth the cost of an upgrade (£63).

There are two different feature sets, the professional version (which enables you to run VMs that have been restricted), and the personal version (which is only allowed for non-commercial use). When you install *Player*, you'll be asked for a licence key, but you can skip this step if you want the personal version.

The graphical interface for *Player* is simple to use, and shouldn't cause any problems even if you're not familiar with virtualisation software. You just select the install media, and follow the instructions, and you should have a working virtual machine. The simple GUI is *Player*'s strongest feature given that other virtualisation applications often have either confusing interfaces, or are command-line only.

Player gives you the ability to control most aspects of the virtual machine including the amount of memory allocated to it, the network setup, the storage available and ability to allocate more than one processing core to the VM. You can also enable USB support for versions 1 to 3 of the protocol.



Welcome To VMware Player

Create a New Virtual Machine
Create a new virtual machine, which will then be added to the top of your library.

Open a Virtual Machine
Open an existing virtual machine, which will then be added to the top of your library.

Upgrade to VMware Workstation
Get advanced features like snapshots, developer tool integration, and more.

Help
View help contents for VMware Player.

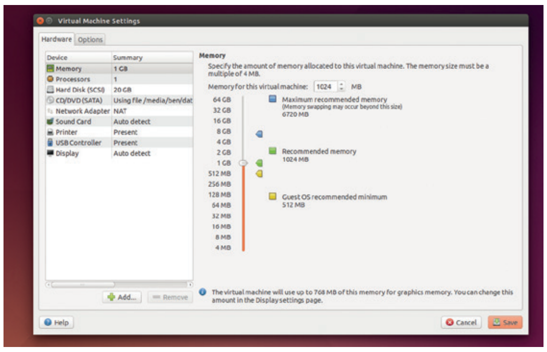
This product is not licensed and is available for non-commercial use only. For commercial use, purchase a license. Buy now.

The clutter-free interface presents the user with only the options they need.

Performance

We tested *Player* against the most similar open source software (*VirtualBox*) in a speed test. The challenge was simple: Which could boot ElementaryOS the fastest? *Player* took 29 seconds from starting the VM to the desktop being fully loaded, but *VirtualBox* managed the task in 25 seconds. That's a performance difference of 16% in favour of the open source software.

If you're looking to move up to enterprise-level virtualisation solutions, *VMware*'s software has some significant advantages (but comes at a significant cost). *Player* is the first rung on the ladder up to this. However, unless you're planning on using it alongside the more powerful specialised VMware tools, *VMware Player* doesn't offer any significant advantages over open source solutions (and is missing some useful features like snapshots). As proprietary software, it doesn't link in with distro-specific tools such as the package manager. LV



Device	Summary
Memory	1 GB
Processors	1
Hard Disk (SCSI)	20 GB
CD/DVD (SATA)	Using file /media/ben/dat
Network Adapter	NAT
Sound Card	Auto detect
Printer	Present
USB Controller	Present
Display	Auto detect

Memory

Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB.

Memory for this virtual machine: 1024 MB

Maximum recommended memory
(Memory swapping may occur beyond this size)
6720 MB

Recommended memory
1024 MB

Guest OS recommended minimum
512 MB

The virtual machine will use up to 768 MB of this memory for graphics memory. You can change this amount in the Display settings page.

VMware Player's settings window includes useful hints for beginners to help them choose sensible settings for their virtual machines.

DATA

Web
www.vmware.com
Developer
VMware
Price
Free for non-commercial use, or £120

LINUX VOICE VERDICT

VMware Player is let down by not having snapshots, and is poor value for money compared to other options.

★★☆☆☆

The developer's code

Ben Everard delves into a programmer's book that's not about programming.

The Developer's Code is a series of essays on the subject of commercial software development. This isn't a book on the subtleties of the art of computer programming; it's all about how to work well in a commercial setting.

The essays are grouped into eight categories (Metaphor, Motivation, Productivity, Complexity, Teaching, Clients, Code and Pride). Each essay is about a skill that isn't directly related to programming, but that programmers need in order to work effectively, such as how to manage teams well, and how to speak to clients. The short essays are easy to read and completely self-contained, so it's easy to read this book in pieces if you don't have time to get through the whole thing in one go.

If your main interest in programming is in open source or solo projects, this book has very little for you. However, if you're serious about progressing in this industry, *The Developer's Code* is the best such book we've come across.

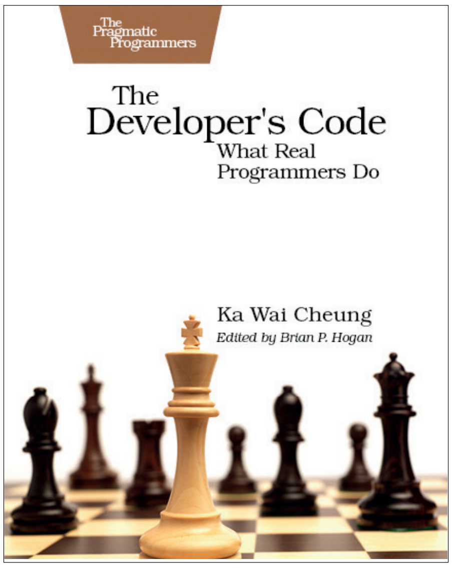
The Developer's Code doesn't include any special new techniques or hidden secrets that haven't been published before. Instead, it presents useful information in an easy-to-digest fashion that will help you use your programming skills more effectively. If you feel as if you're stuck in a Dilbert strip, it could help you out.

LINUX VOICE VERDICT

Author Ka Wai Cheung
Publisher Pragmatic Bookshelf
ISBN 978-1934356791
Price £19.50

A useful book to help professional programmers advance up the corporate ladder.

★★★★☆



As well a physical book, *The Developer's Code* comes as a DRM-free eBook (in a variety of formats) at **https://pragprog.com**.

This Machine Kills Secrets

Dave Rebner is so secretive he won't even tell us his real name.

Cypherpunks – people who believe that sufficiently sophisticated cryptography can change the world for the better – have profoundly affected the last decade. Perhaps the most famous of these people is Julian Assange, but he wasn't the first, and he isn't alone. In *This Machine Kills Secrets*, Andy Greenberg charts the movement from its birth on a mailing list in the early days of the internet to the schisms that rocked Wikileaks following Assange's house arrest. It looks at the differing personalities that often clashed, and the different technology they built to free the world's information.

Greenberg has spoken with many of the key people in this saga, and this book includes first-hand testimony from just about every era the movement covers. Many books focus on the figureheads that are the public face of the movement, but *This Machine Kills Secrets* goes deeper and includes testimony from the geeks that made it all possible. It's well researched, well written, and an excellent read.

This Machine Kills Secrets isn't just a book about how leaking has happened in the past, it's a book about why it has happened, and why it will continue to happen. It's about the motives and passions that have led to the technology leakers use today.

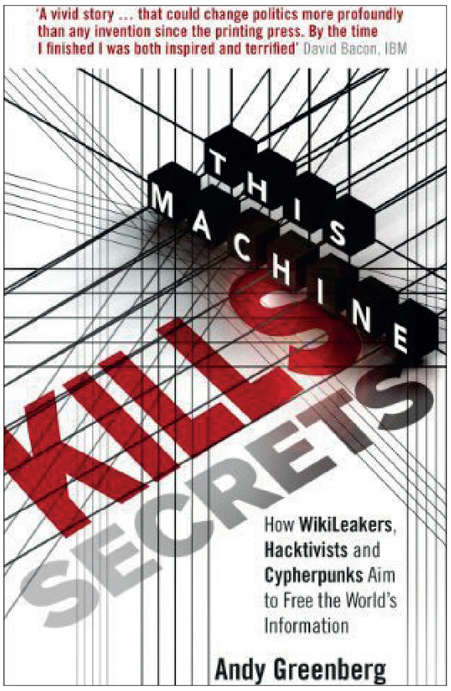
The book finishes with Assange still under house arrest before he fled to the Ecuadorian Embassy, and before Ed Snowden leaked the NSA documents. This doesn't detract from the book, but it does mean that if you're looking for a modern history of leaking, you'll need to supplement this book with another covering the more recent leaks, such as *No Place To Hide* by Glenn Greenwald.

LINUX VOICE VERDICT

Author Andy Greenberg
Publisher Virgin Books
ISBN 978-0753540510
Price £12.99

The information wars are upon us, and this book tells the history of the resistance.

★★★★★


'A vivid story ... that could change politics more profoundly than any invention since the printing press. By the time I finished I was both inspired and terrified' David Bacon, IBM
THIS MACHINE KILLS SECRETS
How WikiLeakers, Hacktivists and Cypherpunks Aim to Free the World's Information
Andy Greenberg

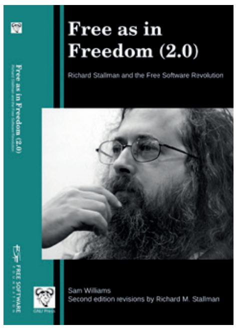
Woodie Guthrie's guitar may have been a machine for killing fascists, but all of our computers can help kill secrets.

Free as in Freedom (2.0)

Mike Saunders pokes his nose into this biography of the mighty RMS.

This book is a few years old now. So why are we covering it here? First: it's available as a free download (in PDF format) from **http://shop.fsf.org/product/free-as-in-freedom-2**. Second: you can get a copy of it signed by Richard Stallman (RMS) himself. And third: it's simply a great read!

The book explores Stallman's career as a programmer and activist, looking at his early work and how his positions on software freedom developed. It's fascinating to read how RMS was merely a humble hacker at the MIT AI Lab in the 1970s and 1980s, with no grand plans for the future, but slowly becoming aware of the increasing commercialisation of software. There used to be a tradition of sharing and openness in the computing world, but RMS saw these principles being eroded, and the straw that broke the camel's back was when he couldn't get access to the lab printer's source code in order to fix some bugs. At least, not without signing an NDA, anyway. The book is a fascinating insight into the culture of the time, and what made RMS the fighter he is today.


Print copies, signed by RMS, are available for $50.

LINUX VOICE VERDICT

Author Sam Williams and Richard Stallman
Publisher Free Software Foundation
ISBN 9780 9831 59216
Price Free (PDF), $20 (book), $50 (signed)

Essential reading for anyone who wants to understand how Free Software came about.

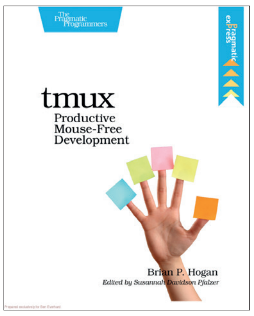
★★★★☆

Tmux: Productive mouse-free development

Does anyone need a mouse? **Ben Everard**'s is now redundant and free.

Tmux, the terminal multiplexer, is a tool for managing many terminals at once. Its most popular features are the ability to run many terminals in separate panes in a single window, and to detach from a session, but leave the terminals running. It's entirely keyboard-driven, so you can control everything without reaching for the carpal tunnel-forming rodent by your side.

Like many powerful keyboard-driven tools, *Tmux* can require a little effort to learn. It doesn't have an especially steep learning curve, and you should be able to get started with it quite quickly. However, if you want to get the most out of it, you'll probably need a little help, and *Tmux: Productive* mouse-driven development is possibly the best way to get started. At 66 pages, it's concise, but at the same time, its six chapters (Learning the basics, Configuring Tmux, Scripting customised tmux environments, Working with text and buffers, Pair programming with Tmux and Workflows) cover everything you need to know. It also lets you know why you might need particular features, not just how to use them.


Once you've mastered *Tmux*, you'll never go back to using he mouse.

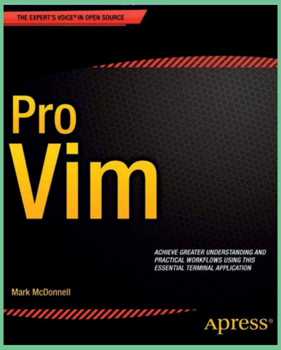
LINUX VOICE VERDICT

Author Brian P Hogan
Publisher Pragmatic Bookshelf
ISBN 978-1934356968
Price £10.99

Tmux is more powerful than most people realise, and this book will help you unlock it.

★★★★☆

ALSO RELEASED...


Master *Vim* and throw your mouse away.

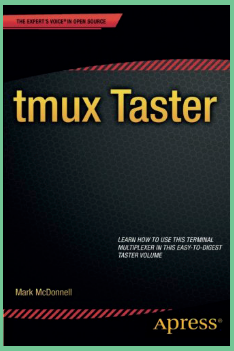
Pro Vim

Learning a powerful text editor can change your life. You might find *Vim* horrendously terse and strange, but once you get your head round it and master the basics, you're well on the path to editing enlightenment. This 424-page tome assists you on the journey.

Why some products sell, and some fail.

Badass: Making Users Awesome

Got a great idea for a product or service, but no marketing budget? This guide, from Kathy Sierra of Head First fame, shows you how to turn your ideas into a success. The key? Making your users feel badass – as in, spiffing, tip-top, in our British English vernacular.


Tmux + *Vim* = heaven. In our humble opinion.

Tmux Taster

This mini guide (96 pages) takes you through the basics of *Tmux*, an extremely handy terminal multiplexer. Wassat, you say? As reviewed immediately to the left? That's right. It's such a useful piece of software that books are springing up like mushrooms in the dew.

LINUX VOICE

GROUP TEST

REMOTE DESKTOP CLIENTS

Tired of hopping from one computer to another, **Mayank Sharma** tests options that allow him to control all his computers remotely.

On Test

Remmina

URL http://freerdp.github.io/Remmina
VERSION 1.1.1
LICENCE GNU GPL
Can this do-it-all software ward off the competition?

KRDC

URL www.kde.org/applications/internet/krdc
VERSION 4.14.1
LICENCE GNU GPL
Does KDE's default app do enough?

Vinagre

URL https://wiki.gnome.org/Apps/Vinagre
VERSION 3.12.2
LICENCE GNU GPL
...does Gnome's?

TightVNC

URL www.tightvnc.com
VERSION 2.7.10
LICENCE GNU GPL
Is the once popular Java app keeping up with the times?

NoMachine NX

URL www.nomachine.com
VERSION 4.3.30
LICENCE Freeware
Is it the best NX client out there?

TeamViewer

URL www.teamviewer.com
VERSION 10.0
License: Freeware
Does it offer more than just convenience?

Remote desktop clients

When you think of remote access the first thing that comes to mind is SSH. System admins have been using it since time immemorial to mount remote directories, back up remote servers, spring-clean remote databases, and even run remote GUI apps. You probably use SSH to interact with your Raspberry Pi anchored behind the TV.

However, there are times when you need to remotely access the complete desktop session rather than just a single app. Perhaps you want to handhold the person on the other end through installing and using a complex piece of graphical software, or want to tweak settings on a Windows machine from the comfort of your Linux distro. That's where remote desktop software comes in handy. Using these nifty little applications you can remotely access and operate a computer from all sorts of devices.

There are various protocols that are designed to interact with a remote desktop. For this group test we've set up the *Vino* VNC server on a Linux Mint machine and a *TightVNC* server on a Raspberry Pi and on a Windows 8.1 box. Many of the clients on test support multiple protocols. The exceptions are the two proprietary clients, which we'll connect to with their own servers.

A good remote desktop client should be responsive, and we'll rate it higher than a client that does a wonderful job of replicating the remote desktop in true colour but takes ages to register clicks and key presses. We'll also keep an eye out for related features such as the ability to encrypt connections and transfer files and audio along with the remote desktop.

The clients and servers are all running inside our network connected via Wi-Fi. While for maximum performance you'd want them to be connected via Gigabit LAN cables, it rules out the all-important convenience factor for most readers.

"There are times when you need to access the whole desktop, not just one app."

Protocol soup

VNC or Virtual Network Computing is one of the most popular mechanisms for accessing a remote desktop. At its heart is the RFB (Remote Framebuffer) protocol, which works at the framebuffer level and is therefore supported by all platforms. One big advantage of the protocol is that you can connect to a VNC server with a client from a different vendor. Then there's Microsoft's proprietary Remote Desktop Protocol (RDP). While the RDP server is only available for the Windows platform, there are clients for Windows, Linux, Mac OS X, Android, iOS and other platforms. Besides these, several proprietary remote desktop solutions have their own proprietary protocols.

Getting started with desktop sharing

The basics behind this essential technique.

A remote desktop sharing session involves a server and a client. The server component is installed on the remote machine that you want to access and the client component is installed on the local machine, or even on a mobile device such as a tablet.

In a typical desktop sharing session, the remote computer (also known as the host, as it's hosting the session) enables a user to view the contents of the host computer's desktop. The remote machine can host this connection on a local network or even over the internet. Furthermore, the host computer can also hand over control of the keyboard and mouse to the other party. In this case, all keystrokes and mouse clicks on the client are registered on the server as if they were actually performed on the remote machine.

You'll also have to poke holes in the firewall on the remote host machine to make sure it allows the client to connect. Different remote desktop servers work on different ports. For example, by default the VNC server listens on port 5900 for connections and on port 5800 for download requests.

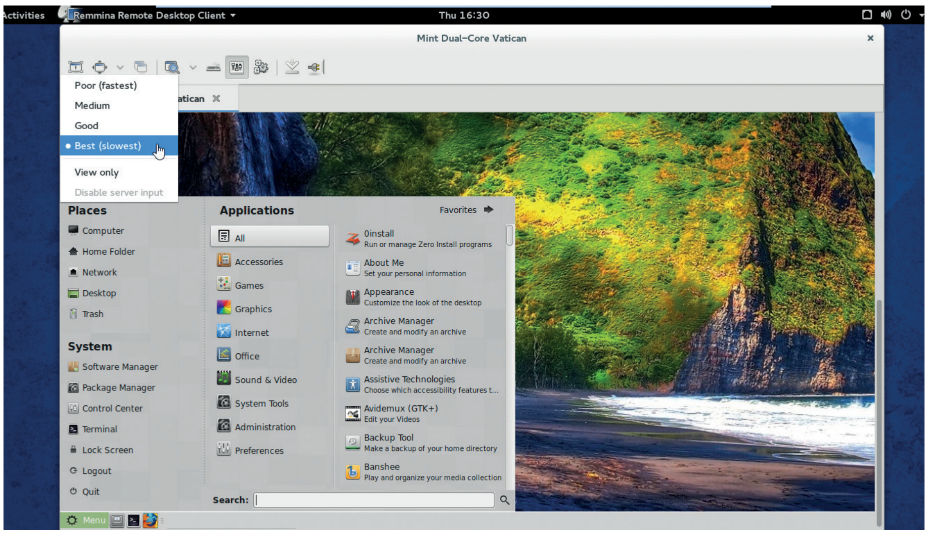
If you use a router, you must configure it to forward connections if you want to connect to a remote desktop over the internet. Remember that in order to establish a remote connection, both the host and the client have to use the same protocol.

Remmina

Maximum performance.

One of the best things about *Remmina* is that the app supports a variety of protocols including VNC, RDP, NX, SSH and more. It has a simple interface, and maintains a list of profiles, and you can organise connections in separate groups. Before you begin, you'll have to create a profile and define the parameters before you can connect to a remote server. At the very least, you'll have to select a protocol from a drop-down list and enter the IP address of the server. Optionally you can define other parameters as well that vary depending on the protocol being used. For example, for VNC connections, you can optionally choose the colour depth and quality of the connection. You also get checkboxes to toggle some quick settings like starting a simple View Only session, disable encryption, and more. If you're connecting to a NX or a RDP server, you can also specify a resolution for the remote desktop.

You can use the app in window mode as well as full-screen mode. Remmina has a tabbed interface that enables you manage multiple remote desktop sessions from a single window. When connected you get a bunch of buttons for common tasks such as switching to full-screen mode, or to the scaled mode in case the remote desktop doesn't fit. You also get a button to change the quality of the connection. Unlike other open source apps, Remmina changes the colour depth of the remote desktop of the fly, which is a definite plus. There's also a button that sends all the keyboard commands to the remote connection. There's a hidden toolbar at the top of the screen which gets you all this control in the full-screen mode.



Remmina is a *GTK* app and will bring along a lot of baggage when installed on a KDE desktop.

Remmina houses default remote connection settings under its Preferences window. Here you can tweak some auto save settings for the connections, define default connection resolution and custom hotkeys. RDP users gets a bunch of additional options to help trim down the size of the remote desktop stream, such as the ability to turn off the wallpaper, menu animations, cursor shadow and more.

Stable performer

To test its responsiveness, we tried playing a *Snake*-like game on the remote desktop. When connected to the puny little Raspberry Pi, the game was playable but the keystrokes were delayed by a block or two. Changing to a lower quality level didn't have any noticeable impact on this delay.

Conversely, when connected to a dual-core Mint box with full-HD resolution, the game was playable even at the best quality setting. However, video playback wasn't watchable at any quality setting – at the lowest quality level the video was less jerky but the colours were all wrong; at the other end of the quality setting the colours were perfect but the the video was skipping frames. Also *Remmina* doesn't transfer audio and lacks the ability to transfer files.

We could easily scroll through lightweight PDFs at best quality, while PDFs with lots of images were best scrolled through at lower quality levels and were readable at the lowest setting. *Remmina* is available in the official repositories of most popular distros.

VERDICT

Impresses with its list of supported protocols, features and performance.

★★★★☆

Krdc

Krude but effective.

Krdc is KDE's default remote desktop client and supports the VNC and RDP protocols. The app does a nice job of handling connections, with the main interface showing a history of connections with the recently accessed servers at the top. You can also arrange the list by the number of visits to a server. You can even bookmark connections you want to use more often.

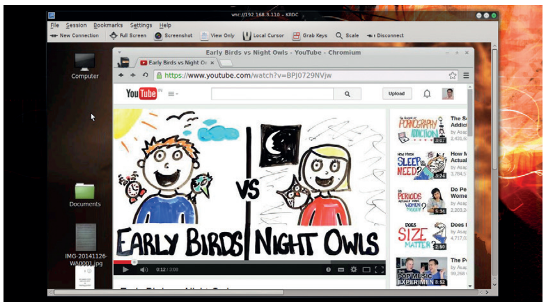
Although the main interface might seem overwhelming to a new user, with a handful of menus and buttons, it's fairly simple to operate. To establish a connection, you only need to select a protocol and enter the IP address of the remote machine you wish to connect to.

This brings up the host configuration box, from where you have to select a quality setting. The default is Medium, which is claimed to be suitable for DSL, cable, and fast internet connections. There's also high quality for LAN and low quality for slower connections. And that's it. Depending on the three settings, *Krdc* works out the other details for the connection. The app's set of choices is rather limited, but you do get the option to manually specify a resolution for the session.

Krdc lists all connected remote computers in different tabs. From within a connection, you get buttons to switch to full screen, scale the remote display to fit the local resolution, take a screenshot of the remote display, change the session into a view-only mode, and send all keyboard inputs to the remote computer.

We connected to another computer on the local LAN using the default medium setting. Videos played flawlessly albeit without sound, and our test PDFs were readable and scrolled nicely as well. However, performance degraded sharply when we selected the High quality option which is suggested for LAN computers. Inversely, the Low quality wasn't of much use as the colours and fonts rendered poorly.

"Krdc's main interface might seem overwhelming, but it's fairly simple to operate."


Early Birds vs Night Owls - YouTube - Chromium
EARLY BIRDS VS NIGHT OWLS

Krdc doesn't offer the option to route the connection through a SSH channel.

VERDICT
A useful client for KDE users who use VNC occasionally.
★★★☆☆

Vinagre

Not nearly as versatile as vinegar.

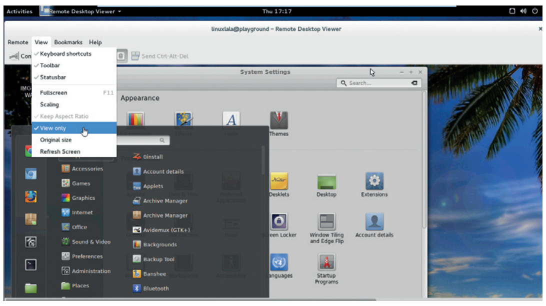
Vinagre, also known simply as Remote Desktop Viewer, is Gnome's default client for viewing remote desktops and supports the VNC, RDP, SPICE and SSH protocols. It has a minimal interface that's very much like *Remmina*. However, there aren't nearly as many advanced options that are available behind *Remmina*'s simple GUI. To connect all you need to do is pick a protocol from the pull-down list and enter its IP address. There's also a very helpful Find button next to the host address field that hunts for active servers on the local network.

Also much like *Remmina*, you get optional checkboxes for starting a full-screen session, a view-only or a scaled window. You also have the ability to select a colour depth from 24-bit true colour to three-bit ultra-low colour. You can also enable JPEG compression if you have the resources to bear the processing overhead.

Another useful option is the ability to tunnel the VNC connection through an SSH server. To establish a secure session make sure you run the SSH server on the remote server that's also the VNC host.

Also remember that before switching to the full-screen mode, you should enable the keyboard shortcuts option (under the View menu) and then use the F11 key to switch between fullscreen and window mode. During an active remote desktop session, you can stop sending keyboard and mouse to the remote desktop and effectively turn it into a view-only session. The interface also includes a button to send the famous three-finger salute (Ctrl+Alt+Del) to the remote desktop.

Performance wise, the app is pretty mediocre. The *Snake* game was playable at all quality levels and while videos played without jerks on the 16 bit High Colour setting they weren't really watchable because of the lack of colours. If you went any higher, the videos became jerky and started skipping frames. Also, Vinagre doesn't send audio, and we had to disconnect to change the quality setting.

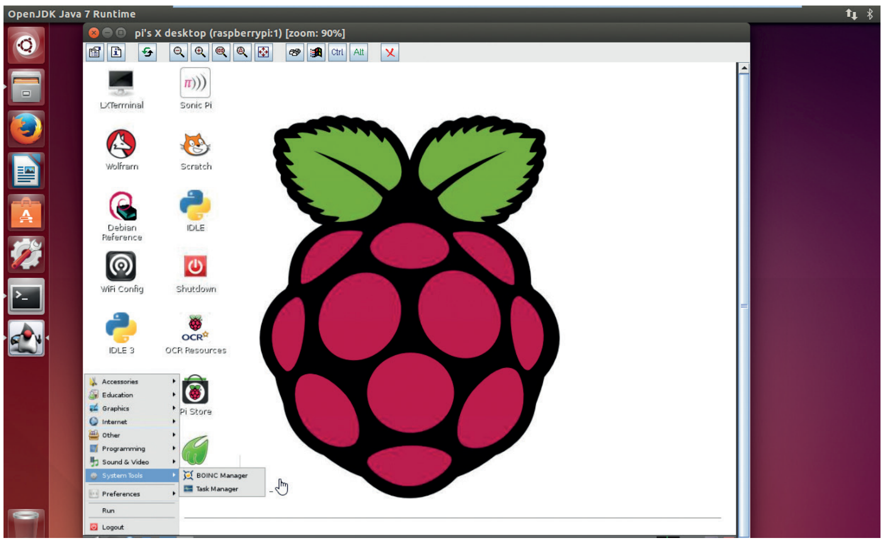


The Reverse Connection option simplifies the process of accessing a host behind a firewall.

VERDICT
Vinagre is to Gnome what *Krdc* is to KDE – a usable default for occasional use.
★★★☆☆

TightVNC

Never let go of your computers.



The project has recently released an Android client optimised for mobile internet connections.

TightVNC is one of the oldest VNC client that's still in development and is the progenitor of many popular VNC clients with different goals. The *TightVNC* project doesn't just produce a client. They also release a VNC server, which is what we use on the Raspberry Pi.

The project uses its own enhanced version of VNC's RFB protocol. The project has added extensions to the RFB protocol to improve performance over low bandwidth connections. *TightVNC* gets its name from the fact that it encodes the VNC stream more tightly by using a combination of the JPEG and Zlib compression mechanisms. However, this compression shouldn't impose any performance penalties on modern processors. In fact, the official Raspberry Pi documentation asks users to run the *TightVNC* server to set up a VNC on the Raspberry Pi.

One of the best things about *TightVNC* is that it is still compatible with other implementations of VNC. However to use its tight encoding and gain full advantage of its enhancements you must use *TightVNC* at both end of the connection. The *TightVNC* client for Linux is written in Java that doesn't need to be installed – just double click on the **.jar** file to launch the client (assuming you have installed the JRE). It works fine with the OpenJDK JRE. We tested the *TightVNC* client by connecting it to the *Vino* server as well as its own *TightVNC* server. When using *Vino*, we had to turn off encryption on the server before *TightVNC* would connect to the server. It also correctly autodetected colour depth. Video playback without the audio was watchable and the Snake game was playable without any issues.

Some fiddling required

When using *TightVNC* server, we had to alter the default config file to show the server's MATE window manager. We had to do a similar modification to view Ubuntu's Unity desktop as well. This connection uses the Tight protocol, although we didn't notice any remarkable improvements. One major difference is that the *TightVNC* server shares a new desktop, while *Vino* shares the same desktop that's currently on the remote desktop.

In addition to the lack of audio from the remote desktop, there's also no means to chat with the user on the other end. Furthermore, the file sharing facility is only supported on the Windows platform. You wouldn't be able to transfer files even if you use the *TightVNC* Linux client to connect to the server on Windows.

VERDICT

A decent performer that covers all your multi-platform multi-device bases.

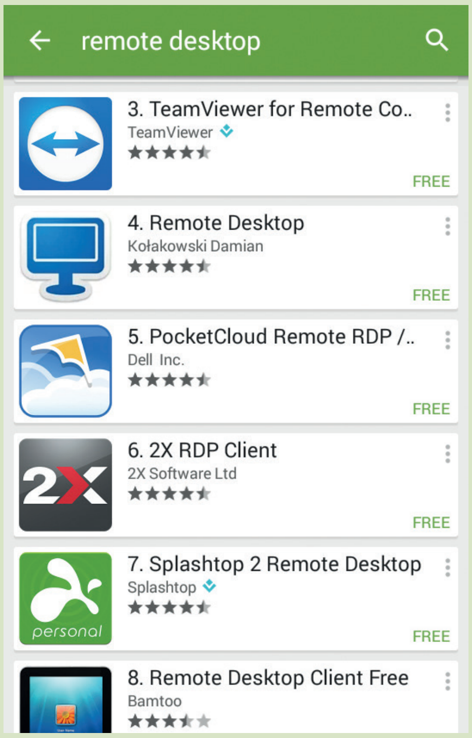
★★★★☆

Other options

There are several other remote desktop clients and solutions that you can use. While we have only covered the most popular and actively developed VNC clients, there are several others. There's *RealVNC*, which is often touted as the official VNC client. You can use it for free for a limited number of connections after registering on its website. Other VNC clients include *TigerVNC*, *TurboVNC* and *x11vnc*. If you're using Microsoft's RDP server, you can connect to it using the *Rdesktop* client.

There's also the open source *Neatx* server based on *NoMachine*'s NX technology, and Google's *Chrome Remote Desktop* tool which is currently in beta for Linux. If you aren't averse to proprietary solutions, there are several paid and freeware solutions such as *Bomgar* and *Mikogo*.

One solution that uses remote desktop sharing technology is *iTALC*. It is an open source solution for remotely accessing, controlling and managing classroom computers and uses the RFB protocol. Also, if you just wish to access a single app and care more about security than zippyness then you can just enable X11 forwarding over SSH.



remote desktop

3. TeamViewer for Remote Co..
TeamViewer
★★★★★
FREE

4. Remote Desktop
Kołakowski Damian
★★★★★
FREE

5. PocketCloud Remote RDP /..
Dell Inc.
★★★★★
FREE

6. 2X RDP Client
2X Software Ltd
★★★★★
FREE

7. Splashtop 2 Remote Desktop
Splashtop
★★★★★
FREE

8. Remote Desktop Client Free
Bamtoo
★★★☆☆

Google Play has several clients for accessing remote desktops on your Android device.

NoMachine NX vs TeamViewer

Cross-platform proprietary freeware at their best.

NoMachine NX uses the NX protocol that tunnels a remote X session across an SSH encrypted channel. The protocol also encodes and compresses data to minimise the bandwidth required. This allows it to do some cool things such as pipe audio from the remote server to the local client.

The tool can automatically pick up any NX servers that are accepting connections running on the LAN. You can also define a new connection by specifying its IP address and the login credentials that you specified while setting up the server. *NoMachine* will then detect the remote resolution and offer to change it to match the local resolution. By default, it'll forward audio to the local client and mute it on the server, but you get the option to unmute it on the server as well.

Once connected you can access all its features from the Session menu, which is accessed from the page peel in the top-right corner of the window. The menu gives you access to some useful features, such as the ability to access a device such as a disk, or printer, stream the mic input to the remote server, and record the remote desktop in a WebM video.

NoMachine lets you export the contents of a local disk to the remote machine or import the remote disk into the local desktop. You can also copy files by dragging them to and from the remote desktop. You can mount a remote disk either as public, which mounts it in **/media** or private which mounts it on the user's desktop. Similarly, you can also manage local and remote printers and USB devices including removable disks, scanners, web cams and more. Performance-wise *NoMachine NX* is phenomenal. Video playback, games and PDFs look and work as if you are operating them on the local computer. There's no noticeable lag and the images and video are very crisp.

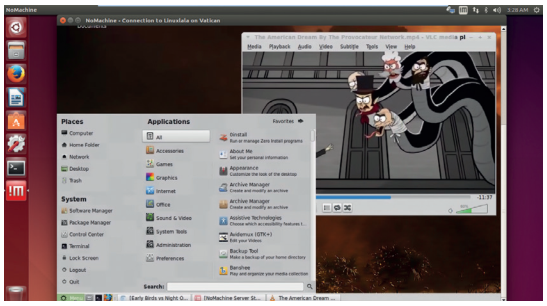
One for the team

TeamViewer is perhaps one of the most recognisable names in the remote desktop domain and the app is used by several major enterprises. However, its Linux client isn't nearly as spectacular. In fact, *TeamViewer*'s Linux client still runs with the help of *Wine*, like its initial version several years ago.

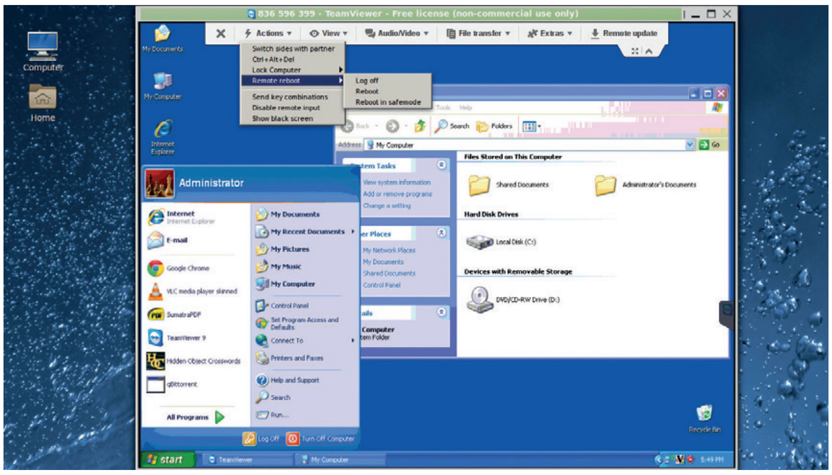
The client offers more features than you get with the usual open source remote desktop clients, and *TeamViewer* uses its own proprietary protocol that connects clients through a central server. One obvious advantage of this scheme is that you can connect to a remote desktop from anywhere, even those behind firewalls, without messing with routers or setting up port forwarding.

TeamViewer is one of the most convenient apps to set up and use. Just launch the client and enter the unique numeric code displayed on the machine you wish to connect to. If you're accessing your own remote computer, you can set up a password and log in unattended. If you are handholding another *TeamViewer* user, all you need from them is their unique code and the randomly generated password that'll be valid only for the current session.

However, the quality of the connection is very poor. You can use it to either render a nice desktop at a slow frame rate or an unreadable desktop at a usable speed. For what it's worth, *TeamViewer* does offer a few extra features such as a text and video chat client, the ability to transfer files as well as a VoIP service. The client can also host group meetings. Some features though, such as the ability to invite other users into a session, require you to sign into a *TeamViewer* account. Like *NoMachine*, you can also record a session in *TeamViewer*'s own **.tvs** format. But we couldn't get the option to convert it into AVI format even after following the manual.


NoMachine places an icon in the system tray that gives you access to tools such as the whiteboard, which can be used for scribbling instant messages.

"Video playback, games and PDFs look and work as if they are on the local computer."



Both *NoMachine* and *TeamViewer* can be installed on any Linux distribution, and have precompiled binaries for RPM- and Deb-based distros.

VERDICT

NOMACHINE NX The best remote desktop solution for the pragmatic user. ★★★★★

TEAMVIEWER It might be a big name on Windows but on Linux it's just a bit meh. ★★☆☆☆

OUR VERDICT

Remote desktop clients

KDE's *Krdc* and Gnome's *Vinagre* are good defaults for their respective desktops and both don't support NX connections. One big turn off with *Krdc* is the limited flexibility with the quality settings which really just left us with only one usable quality option. *Vinagre* didn't fare much better with its mediocre performance.

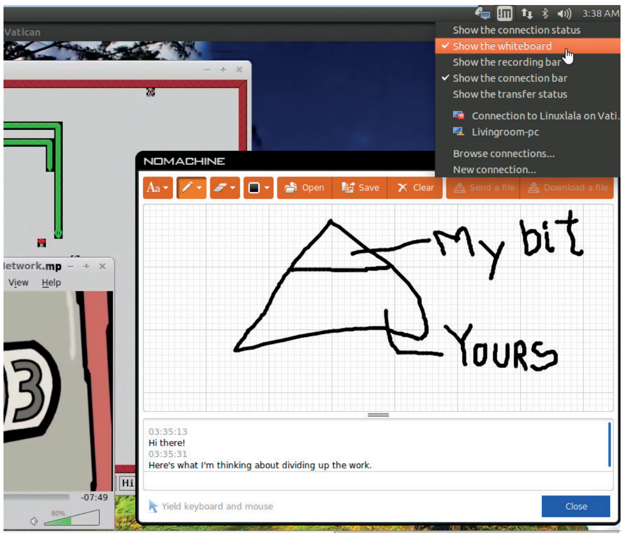
TightVNC is the only Java app on test. The app performed well even when used with other VNC servers. One big advantage of the app is that it's cross-platform and even has mobile clients. It's the recommended server for the Raspberry Pi, and if you're using it on the Pi you should use the client to take advantage of the protocol's tight compression.

However, for maximum coverage there's no beating *Remmina*. The client supports the widest range of protocols and will connect to all kinds of remote desktop servers. The app scores well in the performance department as well and gives you the flexibility to change the quality settings of the connection on the fly. All things considered, *Remmina* is a wonderful remote desktop client and would serve you well if you can live without extra features such as the ability to transfer files

That leaves us with the two proprietary clients. *TeamViewer* turned out to be a disappointment. Its ease of use doesn't outweigh its poor performance and we can't recommend it for any use case. *NoMachine NX* surprised us too, pleasantly, that is. We haven't yet awarded a Linux Voice Group Test to a proprietary app, but *NoMachine NX* is way ahead of everything out there. The open source clients do their bit too and depending on your use case, might be the perfect tool for many of you. However they aren't as comprehensive as *NoMachine NX*, which will work for all types of use cases. You can use it to simply access your remote desktop or any peripherals attached to it or use it to collaborate with another remote user over the internet without much fuss.

"It's proprietary, but NoMachine NX is way ahead of everything out there."

	VNC	RDP	NX	File Transfer	Audio Support
Remmina	Y	Y	Y	N	N
Krdc	Y	Y	N	N	N
Vinagre	Y	Y	N	N	N
TightVNC	Y	N	N	N	N
NoMachine NX	N	N	Y	Y	Y
TeamViewer	N	N	N	Y	N



03:35:13
Hi there!
03:35:31
Here's what I'm thinking about dividing up the work.

At least the NX protocol is open.

1st NoMachine NX 4.3.30

Licence Freeware **Version** 4.3.30

www.nomachine.com
Proprietary software done right. The quality is so good, we have to recommend it even though it isn't free software.

2nd Remmina 1.1.1

Licence GNU GPL **Version** 1.1.1

http://freerdp.github.io/Remmina
Supports the widest range of protocols and performs well.

3rd TightVNC 2.7.10

Licence GNU GPL **Version** 2.7.10

www.tightvnc.com
Best coupled with its own server that's tuned to make best use of limited resources.

4th Krdc 4.14.1

Licence GNU GPL **Version** 4.14.1

www.kde.org/applications/internet/krdc
Good default client for simple use.

5th Vinagre 3.12.2

Licence GNU GPL **Version** 3.12.2

https://wiki.gnome.org/Apps/Vinagre
Another good default for occasional use.

6th TeamViewer 10

Licence Freeware **Version** 10.0

www.teamviewer.com
Overshadowed by every client in terms of performance.

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 26 FEBUARY

HACK THE WEB

ETHICAL HACKING

Security 101: how to hack vulnerable servers with Linux (and how to protect your own from the scumbags who lurk on the internet).

EVEN MORE AWESOME!

Raspberry Pi?
Something big's coming out of East Anglia, and we don't mean Black Shuck. All will be revealed relatively soon.

Ubuntu phones
Father Christmas was kind – he brought us some shiny new hardware to play with, and it's running Ubuntu's smartphone operating system.

Keep it secret...
... keep it safe, with the ultimate paranoiac guide to hiding encrypted partitions on your hard drive. Now go find government secrets...

LINUX VOICE IS BROUGHT TO YOU BY

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

Editorial consultant Nick Veitch
nick@linuxvoice.com

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

Printed in the UK by
Acorn Web Offset Ltd

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

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

Copyright Linux is a trademark of Linus Torvalds, and is used with permission.
Nothing in this magazine may be reproduced without permission of the editor, until October 2015 when all content (including our images) is re-licensed CC-BY-SA.
©Linux Voice Ltd 2014
ISSN 2054-3778

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

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

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

The Internet Protocol

The Internet Protocol is at the heart of – well – the internet. But what exactly does it do?

What would you consider to be the most important inventions of the last 50 years? Genetic engineering? Post-It Notes? The Teletubbies? How about the Internet Protocol? It underpins the entire internet (obviously) and has found its way into cars, fridges, televisions, smoke alarms, in fact the entire "Internet of Things".

Let's start with the big picture, and talk about protocol stacks. Consider the very common case of a web server sending a web page to a browser. The server and the browser communicate with a protocol called HTTP (Hyper Text Transfer Protocol). So the web server builds an HTTP response packet, which consists of the content of the web page it's sending, with an HTTP header stuck on the front. This header contains the information that the HTTP layer needs to do its job. It's in this header, for example, that you'll find the HTTP status code such as 200 (OK) or 404 (file not found).

Enter the TCP layer

Having assembled the packet, the browser hands it down to the transport layer, TCP (see Figure 1). The task of this layer is to "guarantee" delivery of the packet to the correct program (in this case, the web browser) on the destination machine by providing the illusion of a permanent "circuit" connecting the server and the client. This layer adds its own, rather complicated, header to help it do its job. The TCP layer doesn't know anything about the data it's carrying. For example, it doesn't distinguish the HTTP header from the rest of the packet. As far as the TCP layer is concerned, the whole thing is just the "payload" it's being asked to deliver.

The TCP layer hands the packet down to the IP layer, which is responsible for routing packets across an interconnected set of networks (an "internet") to the correct machine. The IP layer adds its own header, and again, it regards the whole of the packet handed to it from the layer above simply as its payload.

There's at least one more layer below that before the packet actually hits the wire. The detail here depends on what medium is being used to actually transmit the packets; assuming that it's some form of Ethernet, the IP datagram will get encapsulated inside an Ethernet frame, with its own header and its own destination address, as I'll discuss. (Though for a tongue-in-cheek alternative, see RFC1149: A Standard for the Transmission of IP Datagrams on Avian Carriers.)

When the packet reaches its destination (where your browser is running) it proceeds back up the protocol stack, each layer discarding its header before passing its payload up to the layer above. Finally the original HTTP packet is handed up to the browser which (after removing the header) renders the page for you.

Each layer thinks of itself as talking directly to its peer layer – the one at the same level in the stack – at the other end. The application layer talks to the application layer, the TCP layer talks to the TCP layer, and so on. In reality, of course, the data flows down and up the protocol stacks.

IP addressing

Back in issues 6 and 7 I discussed the TCP and UDP protocols in some detail, with emphasis on the "sockets" API that provides access to these protocols from our code. I want to focus on the IP layer this month. Typically, programmers do not interact directly with this layer, although it is possible to create a "raw" socket that lets you craft your own transport layer header. Program like **ping**, and some of the weirder forms of **nmap scan**, use this technique. But we are not really going to look at IP through a programmer's eyes.

To begin at the beginning, every connection from a computer to an internet

IPv6

You probably don't need to be told that we're running out of IPv4 addresses. RIPE (the organisation that allocates these things in Europe) started allocating addresses from its last /8 block two years ago (a /8 block is 2^24 addresses, roughly 16 million, which sounds a lot but is actually less than 0.5% of the IPv4 address space).

The number of addresses available in IPv6, with its 128-bit addresses, is too big to get a proper handle on. I just used two Post-It notes working out that you could allocate the equivalent of an entire IPv4 address space for every square millimetre of the earth's surface – in fact, you could do it 100 million times over. Although IPv6 is on its way, it's slow in arriving. You've been able to build IPv6-only intranets with Linux for years. The latest infographic from RIPE claims that globally, more than 20,000 websites, 240 network operators, and 10 home router vendors now offer IPv6 products and services.

Nonetheless, I think we're still some way away from having full end-to-end IPv6 connectivity from the average home user to the average website. I keep thinking I'll call my ISP and ask them… but they'll just tell me to reboot my router and see if that fixes the problem…

is allocated an IP address, which is 32 bits long and is written in a format called "dotted decimal notation" – you split the address into four lots of 8 bits (called an octet), write each octet's value down as a decimal number (giving a value between 0 and 255) then stick dots in between. So you end up with something like 104.28.7.18. This address is logically split into two parts – a network ID and a host ID. The network ID is the piece that's used for routing (getting packets to the right network); the host ID only comes into play once a packet has reached the right network, when it's used for the final stage of delivery to the destination machine.

What's a subnet mask?

The division between the network piece and the host piece is specified by the "subnet mask", which is also usually written in dotted decimal notation. For example, a subnet mask of 255.255.255.0 converted to binary gives us 24 ones followed by 8 zeros, meaning that the top 24 bits of the IP address are network ID, and the remaining 8 bits are the host ID. There's a more compact way of representing this. We might say that a machine is on the network 192.168.1.0/24, meaning that the top 24 bits of this (the 192.168.1 piece) specify the network and the remaining 8 bits select the host.

Figure 3 shows a typical small internet. Machines A, B, C and D are connected to the upper network 192.168.0/24; machines P, Q, and R are connected to the lower network 192.168.1/24. Additionally, machine S is connected to both networks (it has two network cards) and can route packets between them. Finally, machine D has a connection to the outside world.

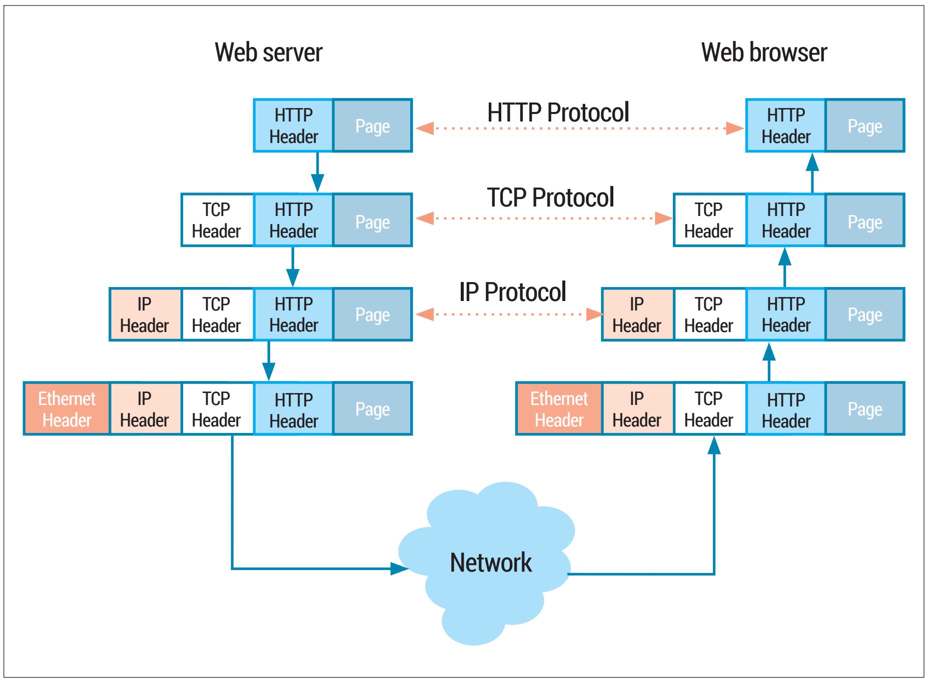


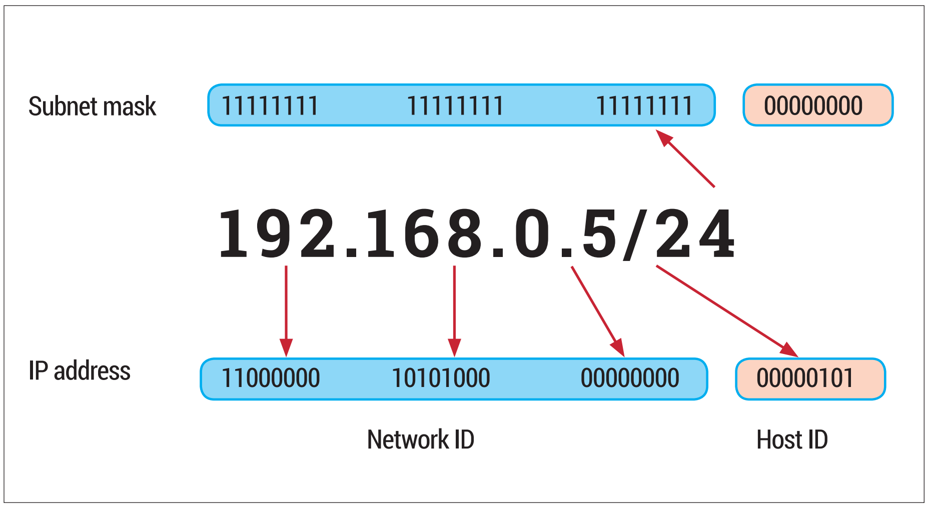
Figure 1: As a packet passes down through the layers of a protocol stack, each layer's header forms part of the payload of the layer below.

In the early days of the internet every single machine that used TCP/IP had a globally unique IP address assigned to it. We could establish direct end-to-end connectivity between any two machines. But the internet grew way beyond expectations and we started running out of addresses. So in 1996, the Internet Assigned Numbers Authority designated three "private" address blocks as follows:

10.0.0.0–10.255.255.255 (10/8 prefix)
172.16.0.0–172.31.255.255 (172.16/12 prefix)
192.168.0.0–192.168.255.255 (192.168/16 prefix)

The idea was that machines that only needed to communicate within their own private "intranet" could use IP addresses from these private blocks and didn't need to apply for an address allocation from a central registry. More than anything else, this strategy has staved off the exhaustion of IPv4 addresses, as countless corporate networks around the world re-use these private address blocks.

In our diagram, there is only one globally unique IP address – that's the 176.13.4.92 address of the outward-facing connection of machine D.



An IP address is split into a network ID and a host ID. The "CIDR" notation shown here specifies the boundary between the two pieces

The routing routine

So, what does the IP layer do, exactly? Well, it has the job of delivering a packet to a specified destination IP address. To figure out how to send IP packets on their way, each machine maintains a routing table. Machines on "stub" networks, like machine P in the diagram, only need to know two things – which network they're connected to (192.168.1/24 in this case) and where to send packets destined for other networks (192.168.1.254 in this example); this is usually called the default gateway.

If we examine the routing table of machine P, we'll see something like this:

```
$ route -n
Kernel IP routing table
Destination   Gateway       Genmask       Flags Iface
0.0.0.0       192.168.1.254 0.0.0.0       UG    eth0
192.168.1.0   0.0.0.0       255.255.255.0 U     eth0
```

I'll explain all this in a minute, but first let's

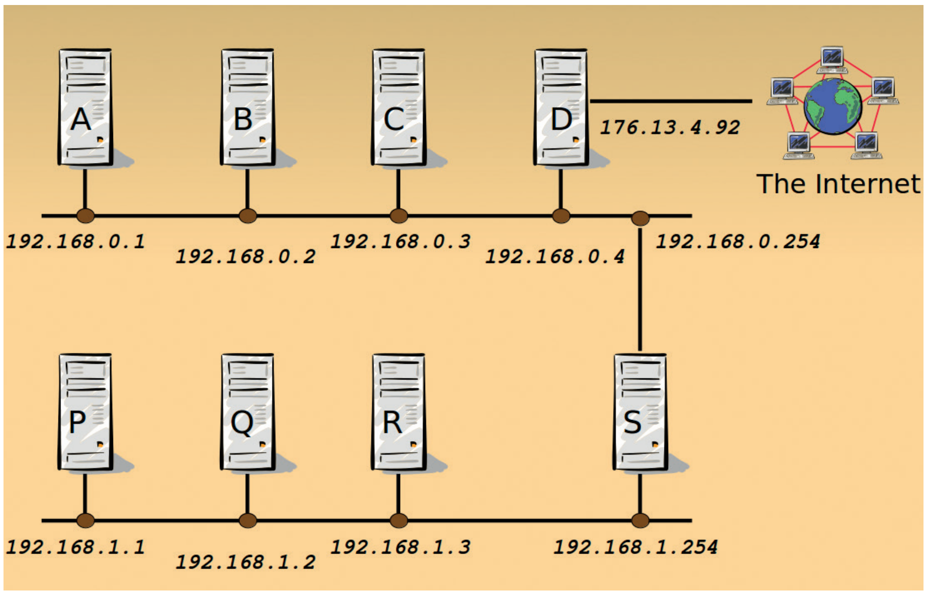


Figure 3: The internet in miniature – two networks connected by a gateway.

look at the routing table for machine A, which has an extra entry because it needs a route onto the lower network, so it might look like this:

```
$ route -n
Kernel IP routing table
Destination   Gateway       Genmask        Flags Iface
0.0.0.0       192.168.0.4   0.0.0.0        UG    eth0
192.168.1.0   192.168.0.254 255.255.255.0  U     eth0
192.168.0.0   0.0.0.0       255.255.255.0  U     eth0
```

Here's how it works. The IP layer works through the entries in the routing table in turn. For each one, it takes the packet's destination IP address, bit-wise ANDs it with the value in the Genmask column and compares it to the value in the Destination column. If they match this is considered as a potential route. If more than one entry in the routing table matches, the most specific route – the one with the longest Genmask – wins.

So, taking the three entries in turn: the first entry always matches, because any destination IP address AND-ed with 0.0.0.0 is going to give 0.0.0.0. So this route will be used if there isn't a more specific match – it says that 192.168.0.4 is our default gateway. The second entry defines the route onto the lower network; basically it says "to reach the 192.168.1/24 network, go via 192.168.0.254". The third entry has no gateway defined; it says that traffic to the 192.168.0/24 network doesn't need to go via a gateway because that's the network we're actually connected to. In all three cases, packets will go out via network interface eth0. That's a bit of a no-brainer because it's the only one we've got. Let's take a look at the routing table on machine S:

```
$ route -n
Kernel IP routing table
Destination   Gateway       Genmask        Flags Iface
0.0.0.0       192.168.0.4   0.0.0.0        UG    eth0
192.168.0.0   0.0.0.0       255.255.255.0  U     eth0
192.168.1.0   0.0.0.0       255.255.255.0  U     eth1
```

A careful examination of this (the last two lines) shows that the machine has direct connections to two networks, 192.168.0/24 (via its "upper" network connection eth0), and 192.168.1/24 (via its lower connection eth1).

To get a feel for how IP routing and packet delivery works, let's consider three routing scenarios in turn:

1 Machine A to machine C.
2 Machine A to machine Q.
3 Machine A to a machine somewhere in the outside world.

Machine A to machine C

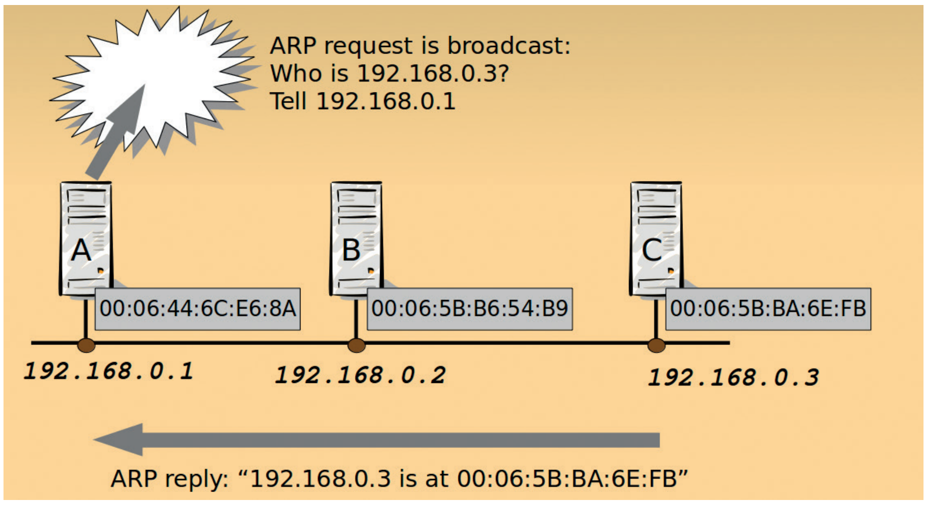
This is the easy case, because the destination address of the packet, 192.168.0.3, is on the same network as machine A, as determined by the third entry in machine A's routing table. But we're not quite home and dry, because the packet needs to be encapsulated into an Ethernet frame for transmission, and we need to know the Ethernet address of machine C. Ethernet addresses are 48 bits long and are written down as a group of 6 pairs of hexadecimal digits, separated by colons, for example 00:06:5B:BA:6E:FB. Ultimately, it's this address that's used to get the packet to the right machine.

Keep in mind though, that it's pointless addressing a packet to an Ethernet address that isn't on your network -- these addresses are not used for routing. The Address Resolution Protocol (ARP) is used to discover the Ethernet address. Essentially, machine A broadcasts an ARP request onto its local network that says "Who is 192.168.0.3? Please tell 192.168.0.1". All the machines pick up and ponder this request but only machine C, recognising its own IP address, responds with the reply: "192.168.0.3 is at 00:06:5B:BA:6E:FB". Finally, machine A is able to build an Ethernet frame and send it out on the wire in the reasonable expectation that it will reach machine C.

Broadcasting an ARP request every single time you want to send an IP datagram is clearly not smart, so machine A will keep the result for a while (60 seconds by default) in its ARP cache. You can examine this cache with the **arp** command:

```
$ arp -a
? (10.0.2.2) at 52:54:00:12:35:02 [ether] on eth0
```

You can also manually add and delete ARP



A broadcast ARP request is used to find the Ethernet address of a directly connected machine whose IP address is known.

cache entries with this command, though there shouldn't be any need to.

Machine A to machine Q

Our second scenario, machine A sending to machine Q, is a little more complicated. The destination IP address for machine Q is 192.168.1.2. From the second entry in machine A's routing table, it discovers that to reach this network it needs to send the packet to 192.168.0.254, the upper network connection of machine S. So it will check its ARP cache for an entry for this IP address, and use the associated Ethernet address if it finds one, or broadcast an ARP request if it doesn't. Note that machine A has absolutely no idea what will happen to the packet after it reaches machine S.

The focus of attention now turns to machine S, the gateway. Tasked with delivering the packet to 192.168.1.2, it discovers from its routing table that one of its network interfaces (eth1) is directly connected to that network. So it will broadcast an ARP request on eth1 to get an Ethernet address for machine Q, and finally send the packet to its destination.

Machine A to the outside world

In our final routing scenario, machine A wants to send a packet to a machine out in the internet -- perhaps to Linux Voice's web server at 104.28.6.18. Machine A quickly discovers that its only hope is to go via its default gateway (machine D at 192.168.0.4). Now we haven't looked at machine D's routing table, but it will in turn discover that the packet needs to go out on the 176.13.4.92 interface to its own default gateway -- a machine operated by the site's Internet service provider.

But there's a problem. Sending the packet out with a destination address of 104.28.6.18 and a source address of 192.168.0.1 will work fine, but getting a reply back is a different matter: 192.168.0.1 is a private address; you can't route packets to it across the internet.

So here's what happens. Machine S picks an unused TCP port on its outward-facing interface. Suppose it picks port 13348. It then re-writes the SOURCE IP address and port number on the packet to be 176.13.4.92 and 13348, and sends the packet on to Linux Voice's web server. This server thinks the request originated at machine S and sends the reply back there; that is, to 176.13.4.92 port 13348. Machine S, meanwhile, has remembered the IP address and port number that this request originally came from – ie, machine A. So it now re-writes the DESTINATION IP address and port number of the reply packet and sends it back to machine A.

"NAT is fundamental to how machines on private internal networks interact with the outside world."

This trick is known as NAT (Network Address Translation) and is fundamental to how machines on private internal networks can interact with the outside world. If you browse the web from home, your broadband router does this on every single packet you send. Note that machine A has no idea that NAT is taking place – as far as it's concerned, it's sending the packet to machine S simply because it's the default gateway to the outside world.

NAT is, in a sense, extending the IP address space by using the port number as part of the address. This form of NAT is sometimes called IP masquerading, because it hides the internal structure of our network from the outside world. It only works when a network connection is initiated from a machine within the local intranet. A web browser running somewhere "out there" cannot connect to a web server running on our intranet. In this sense, NAT offers a kind of firewall, protecting our systems from external attack.

So... if you get the impression that all this routing stuff can get complicated... well, you're right. But keep in mind that the operations I've described occur thousands of times on maybe a dozen machines, just for a single visit to a website. Long live IP!

So long and thanks for all the fish

I've decided to stop writing for Linux Voice, as part of a process that I am told is called "retirement". So this will be my last piece. I'd like to thank the Editorial Team for offering me the opportunity over the last months, and wish them every success as they take the magazine through into its second year. Thanks to all of you who have read what I've written – that's what makes it all worthwhile. Oh, and the cheques of course.

Command of the month: ip

The **ip** command is the main administrative tool for things down at the IP layer. It's intended to replace commands like **ifconfig**, **route** and **arp**. As such, it's a bit of a jack-of-all-trades, with an extensive command syntax. Commands are basically of the form:

```
# ip object action
```

where the objects you can perform actions on include addresses, network interfaces (**ip** calls them links), **arp** cache entries, and routes. The actions you can perform depend on the object you're operating on, but typically you can show, add or delete them. Here are a few examples:

To show all addresses assigned to all interfaces (roughly analogous to the old **ifconfig -a**):

```
$ ip address show
```

To list just the IPv6 addresses assigned to eth0:

```
$ ip -family inet6 address show dev eth0
```

To show the routing table (similar to **route -n**):

```
$ ip route show
```

To add the static route from machine A to the bottom network in our example:

```
$ sudo ip route add 192.168.1.0/24 via
192.168.0.254 dev eth0
```

...and to delete it again:

```
$ sudo ip route del 192.168.1.0/24
```

The **help** option of the command makes it, to some extent, self-documenting. For example:

```
$ ip help
```

will give you a list of the object you can operate on, and drilling down a level further:

```
$ ip route help
```

will show you the actions you can perform on a route.

I get the impression that the **ip** command hasn't gained quite the level of adoption that it perhaps deserves, a result (I suspect) of its extensive command syntax, and the inertia of the sysadmin community

FOSSpicks

Sparkling gems and new releases from the world of Free and Open Source Software

Hunting snarks is for amateurs – **Ben Everard** spends his time in the long grass, stalking the hottest, free-est Linux software around.

Web-based database management

PHPMyAdmin

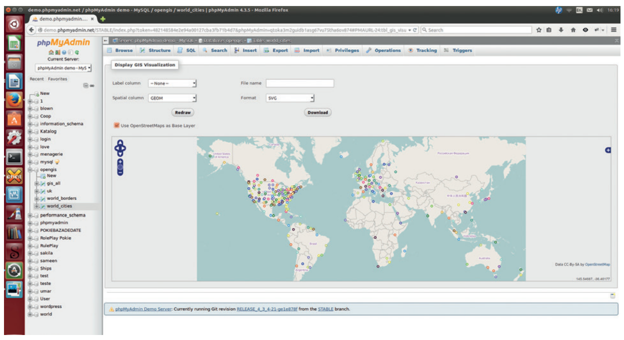
PHPMyAdmin may sound like a tool for administering PHP, but it's not. It's a front-end for *MySQL* and *MariaDB* written in PHP. From creating databases and tables, to backups, to finding particular pieces of data in the tables, *PHPMyAdmin* really can perform just about everything you need to do on a database, but for anything that's not directly supported, there's an SQL interface on the web page.

Unsurprisingly given the name, *PHPMyAdmin* runs on top of a LAMP stack, so if you've already got this installed, then getting *PHPMyAdmin* is just a case of downloading it and unzipping it somewhere in the webroot.

Configuring *PHPMyAdmin* can be a little more awkward. This is either done by hand, or by moving **config.inc.php** into a config folder and using the web-based script. Full details are on the project's website at **http://docs.phpmyadmin.net/en/latest/setup.html**. Alternatively, most distros have a package for *PHPMyAdmin*, though this might not always be the latest version.

One of the big advantages of *PHPMyAdmin* is that it makes it easy for non-experts to manage databases. Backing up and querying are probably the most basic tasks, and these are easily performed provided you know a bit about databases. The search tool works as an SQL query builder, so it helps you learn SQL as you use it. In fact, the whole *PHPMyAdmin* interface is closely tied to SQL, so it's easy to transfer your skills both ways. If you learn in *PHPMyAdmin*, it can be easy to pick up SQL, and if you already know SQL, you should find it easy to get started with *PHPMyAdmin*.

"PHPMyAdmin really can perform just about everything you need to do on a database."

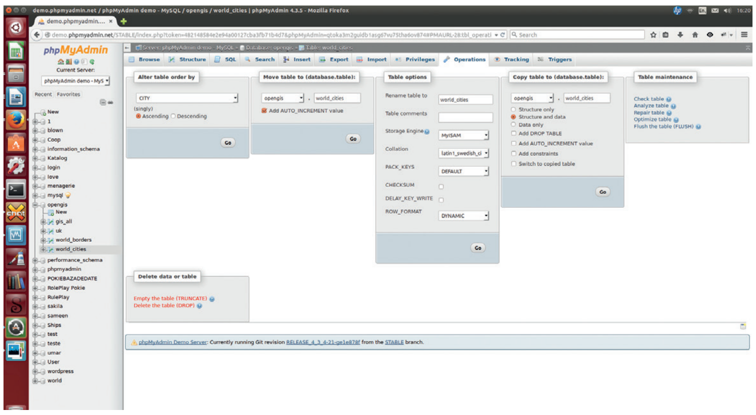


The cities of the world displayed in *PHPMyAdmin*'s GIS data view on top of an OpenStreetMap outline.

As well as general database tools, there's a range of tools to help you visualise data including GIS (geographical) data map overlays, various chart-drawing tools, and image viewers.

There are also plenty of features for advanced users. The profiling options can help you optimise the performance of queries, and just about everything can be tweaked so it works the way you want it. There's even an advisor that tries to highlight potential performance problems and solutions. The server monitoring tools can then help you tell how effective any optimisations have been.

If you're building your database from scratch, there's also a relational designer tool to help you create or amend a schema, and see how the keys are set up. We can't recommend this tool highly enough. You can try out a live demo of *PHPMyAdmin* without installing at **http://demo.phpmyadmin.net**.



PHPMyAdmin isn't the best-looking HTML interface, but it is themeable if you prefer a different colour scheme.

PROJECT WEBSITE
www.phpmyadmin.net

Online collaborative text editor

Etherpad

Etherpad is an online notepad for real-time collaboration. That means you can work with people on a single text document and see what each other is doing as you're doing it. Originally, *Etherpad* was closed source, and Google bought the company producing the software with the aim of including it in the ill-fated Google Wave. Now *Etherpad* is open source, and is hosted in thousands of places both public and private.

Etherpad is based around what it calls 'pads', which is just another name for text documents. These can be quickly created, and shared between users using just the URL.

Slightly confusingly, there were once two pieces of software, one called *Etherpad* and another called *Etherpad-lite*. The original *Etherpad* is now defunct, and *Etherpad-lite* is commonly known as *Etherpad* (though you may still see the -lite suffix in some filenames).

The main focus of the app is on collaboration rather than word processing, so the stylistic options are quite limited. You can use bold, italic, lists, alignments and a few other basic functions, but not much. In some ways, you can think of it a bit like a stripped down, open-source version of Google Docs, or perhaps a real-time wiki.

It's this collaboration focus that makes it an invaluable tool for many open source projects.

You can view a time-slider of how the document has developed, and the contributions by different people show up in different colours. There's also an in-built chat function to help you communicate with the other people editing the document.

Collaboration tool

If you need more features, there are plenty of plugins to provide everything from spellchecking to printing to turning the pad into a collaborative development environment. You can see a full list of available options at **https://github.com/ether/etherpad-lite/wiki/Plugin,-a-list**.

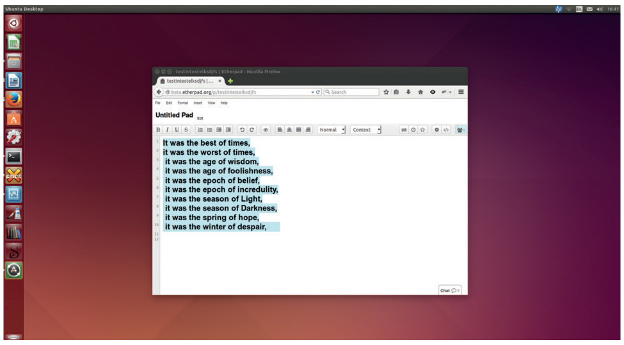
"The main focus of Etherpad is on collaboration rather than word processing."

JavaScript is used both for the interface and the back-end (via Node.js).Keeping everything in one language is useful for development, but does mean that it won't run on a normal LAMP stack. If you don't want to install Node, you can always use one of the many public *Etherpad* instances that are available, such as Wikimedia's (**https://etherpad.wikimedia.org/**), or Mozilla's (**https://etherpad.mozilla.org**). There's even a public pad hosted as a *Tor* hidden service at **https://5jp7xtmox6jyoqd5.onion/**. You'll find a list of public pads on the *Etherpad* site at **https://github.com/ether/etherpad-lite/wiki/Sites-that-run-Etherpad-Lite**. Each site has different terms of use, and some delete pads after a certain amount of time, so always take a look at the terms and conditions before hosting important work anywhere.

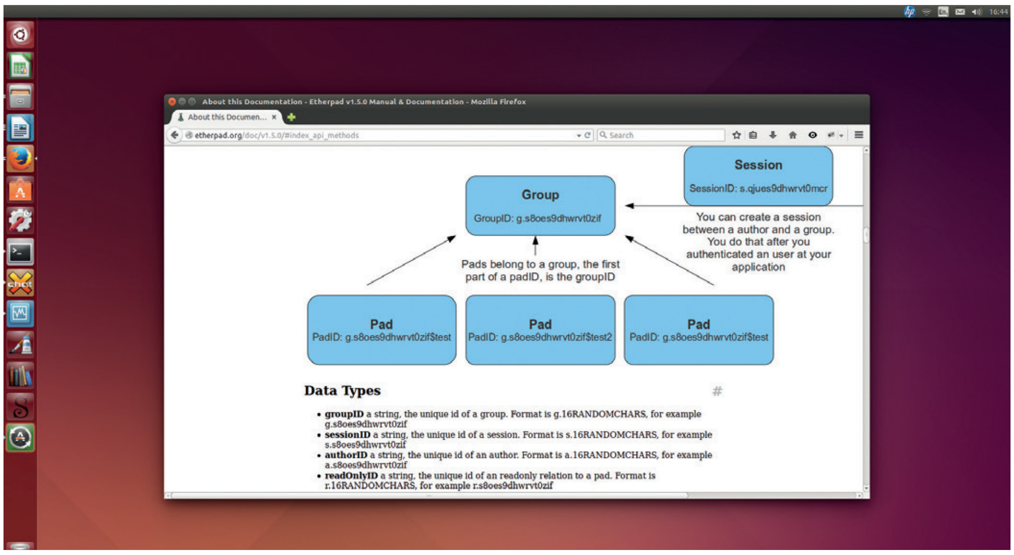
If you already run a site based on a popular CMS such as *WordPress* or *Drupal*, you should be able to find a plugin to integrate *Etherpad* into your website. There's also an HTTP API so you can interact with it from almost any software.

Along with the usual bugfixes and UI improvements, the big improvements in the latest version (1.5 aka Turkey Slayer) are full import and export support (including to word processor formats such as Microsoft's DOC and the open source ODF), and support for sharding to help scaling to large numbers of users.

The new export functions alone are enough for 1.5 to be a worthy upgrade for anyone already running *Etherpad*, and make it even more attractive to new users.


It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness,
it was the epoch of belief,
it was the epoch of incredulity,
it was the season of Light,
it was the season of Darkness,
it was the spring of hope,
it was the winter of despair,
testintestelksdjfs | Etherpad - Mozilla Firefox
beta.etherpad.org/p/testintestelksdjfs
File Edit Format Insert View Help
Untitled Pad
Normal
Context
Chat

The minimalist interface forces you to focus on the content, not the application.


About this Documentation - Etherpad v1.5.0 Manual & Documentation - Mozilla Firefox
etherpad.org/doc/v1.5.0/#index_api_methods
Session
SessionID: s.qjues9dhwrvt0mcr
Group
GroupID: g.s8oes9dhwrvt0zif
You can create a session between a author and a group. You do that after you authenticated an user at your application
Pads belong to a group, the first part of a padID, is the groupID
Pad
PadID: g.s8oes9dhwrvt0zif$test
Pad
PadID: g.s8oes9dhwrvt0zif$test2
Pad
PadID: g.s8oes9dhwrvt0zif$test

Data Types

- **groupID** a string, the unique id of a group. Format is g.16RANDOMCHARS, for example g.s8oes9dhwrvt0zif
- **sessionID** a string, the unique id of a session. Format is s.16RANDOMCHARS, for example s.s8oes9dhwrvt0zif
- **authorID** a string, the unique id of an author. Format is a.16RANDOMCHARS, for example a.s8oes9dhwrvt0zif
- **readOnlyID** a string, the unique id of an readonly relation to a pad. Format is r.16RANDOMCHARS, for example r.s8oes9dhwrvt0zif

The *Etherpad* API is well documented, so interacting with it should be straightforward.

PROJECT WEBSITE
http://etherpad.org

Shell script paralleliser

Gnu Parallel 20141122

Modern computers have many CPU cores, but quite a lot of command line utilities were designed back when most machines had only a single core. This means that, by default, they don't fully utilise your hardware. There is a solution: *Gnu Parallel*. This isn't a panacea that makes single-threaded programs multi-threaded, but it load-balances command line programs across multiple CPU cores.

Imagine, for example, you have a directory with many gzipped files that you want to unzip. The simplest way of decompressing them is with a simple **for** loop:

```
for file in *.gz; do gunzip $file; done
```

However, this would run the entire operation on a single CPU core, which could be quite slow if you have a lot of zipped files. Instead of running them on a single thread, you could launch new threads for each unzipping. This would run across many CPU cores, but wouldn't do so very intelligently:

```
for file in *.gz; do gunzip $file & done
```

A more efficient option is to use *Gnu Parallel*. This intelligently spreads the load over all available CPUs, and should run faster than either of the previous examples. Unzipping the files is done with:

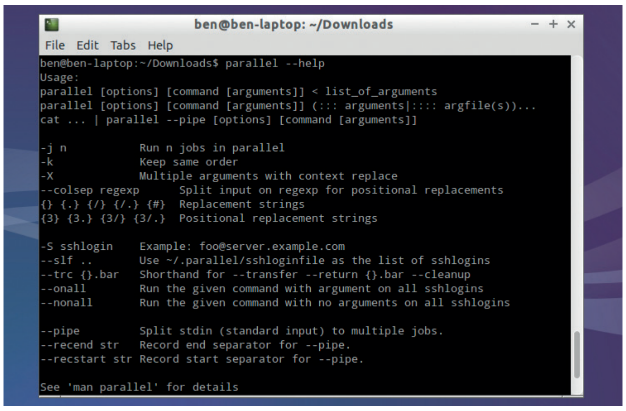
```
parallel gunzip ::: *.gz
```

Unleach the power!

This is the simplest form of the command, and is an easy substitute for a **for** loop on files when there's quite a bit of processing to do. The three colons simply split the command to run from the selector of the files.

Gnu Parallel has far more power than this, and there's a thorough tutorial on the Gnu website at **www.gnu.org/software/parallel/parallel_tutorial.html**

Perhaps one of the most powerful features is the ability to split-run across multiple machines. This may be the simplest method of utilising the processing power of a cluster of machines – perhaps a bunch of EC2 machines you've just split up, or an office full of PCs that are unused overnight.



```
ben@ben-laptop:~/Downloads$ parallel --help
Usage:
parallel [options] [command [arguments]] < list_of_arguments
parallel [options] [command [arguments]] (::: arguments|:::: argfile(s))...
cat ... | parallel --pipe [options] [command [arguments]]

-j n            Run n jobs in parallel
-k              Keep same order
-X              Multiple arguments with context replace
--colsep regexp      Split input on regexp for positional replacements
{} {.} {/} {/.} {#}  Replacement strings
{3} {3.} {3/} {3/.}  Positional replacement strings

-S sshlogin     Example: foo@server.example.com
--slf ..        Use ~/.parallel/sshloginfile as the list of sshlogins
--trc {}.bar    Shorthand for --transfer --return {}.bar --cleanup
--onall         Run the given command with argument on all sshlogins
--nonall        Run the given command with no arguments on all sshlogins

--pipe          Split stdin (standard input) to multiple jobs.
--recend str    Record end separator for --pipe.
--recstart str Record start separator for --pipe.

See 'man parallel' for details
```

If you have a multi-core CPU, your commands should take full advantage of it.

PROJECT WEBSITE
www.gnu.org/software/parallel

Source code management

Git 2.2

Source code management may not be an exciting subject to most people, but it can be a real hassle when it goes badly, like when you can't locate a change that broke something, or get into a mess when two people have changed the same file. *Git* not only solves all the normal problems of source code management wonderfully, it was also designed from the ground up to work for open source projects (specifically the Linux kernel).

The most powerful feature of *Git* is the forking and merging options. Forking in *Git* doesn't mean the same thing as forking software in general. It creates a new copy of the code that can be worked on independently, and the idea is that it will eventually be merged back into the main branch.

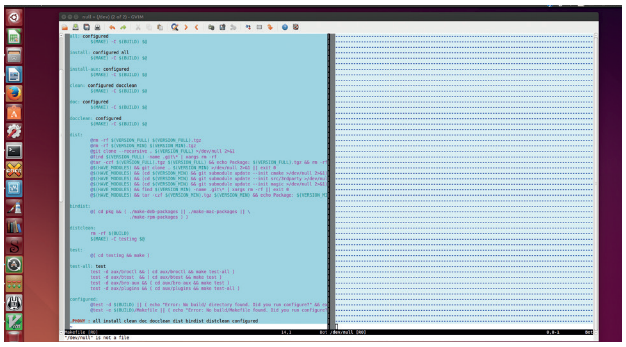
This enables different people to develop on different trees simultaneously, then combine their changes with minimal fuss, which is perfect when developers are working on different features. Understanding how these branches work is key to using *Git* effectively.

While *Git* is amazingly useful, it's not always the easiest software to get started with. GitHub (a website that hosts *Git* repositories) have put together a web-based tutorial to help you get started. You'll find an easy introduction to the world of *Git* at **https://try.github.io/levels/1/challenges/1.** By the time you've completed challenge 25, you should be well versed in source code management.

As well as the command line tools, there are graphical clients, and web-based interfaces.

However, unless you've got a good understanding of how the software works, you'll struggle to get the most out of them.

Despite being a relatively recent option when compared to the likes of *CVS* and *Subversion*, *Git* is already the most popular source code management tool in the open source world, and is becoming increasingly popular in the commercial world as well.



There are graphical clients, such as *Git-cola* shown here, for anyone who wants the power of *Git* but prefers not to use the command line.

PROJECT WEBSITE
http://git-scm.com

Music streaming software

mps-youtube

YouTube hosts millions of videos that anyone with a web browser can play, and a large number of these videos include music. This essentially makes it a massive, free (zero cost) music library. The only downside to this is that the web interface isn't ideal for music playing. *Mps-youtube* is a terminal application for searching videos and playing music without getting bogged down with the graphics.

You can install it with a simple:

```
sudo pip install mps-youtube
```

Then start it with **mpsyt**. You'll be dropped into a command prompt that you can search. For example:

```
search big buck bunny
```

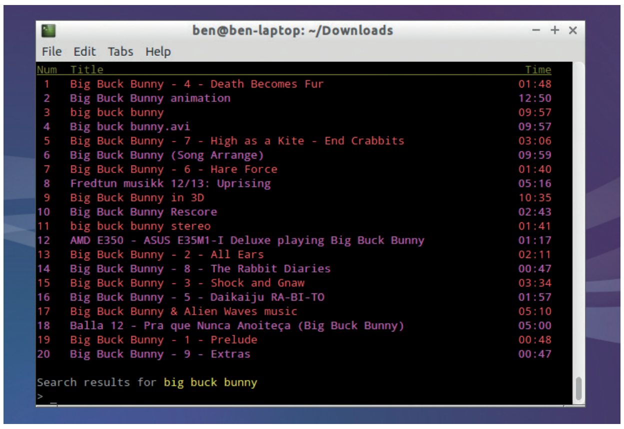
will bring back all the audio from the free (as in speech) film *Big Buck Bunny* made by the Blender Foundation. This uses YouTube's search feature – which is powered by Google – so it does a good job of finding the most popular response to queries, even if they're incomplete or not spelled perfectly.

To play a song, just enter the number of the result and hit Enter. This will stream the song straight from Google's website. Playlists and other more advanced features are supported, and altogether it's quite a powerful music player. You can add items to a playlist with:

```
add <number>
```

The **vp** command displays the current playlist. There are other options to save playlists to your local machine, and open ones that you created earlier. The help system is comprehensive, so you shouldn't have much difficulty working out how to use it. Entering help tips at the command prompt is the best way of discovering all the features.

YouTube isn't known for its high audio fidelity, so this isn't a tool for replacing a high-end stereo, but a nice way of finding new songs.



```
ben@ben-laptop: ~/Downloads
File  Edit  Tabs  Help
Num  Title                                                         Time
 1   Big Buck Bunny - 4 - Death Becomes Fur                       01:48
 2   Big Buck Bunny animation                                     12:50
 3   big buck bunny                                               09:57
 4   Big buck bunny.avi                                           09:57
 5   Big Buck Bunny - 7 - High as a Kite - End Crabbits           03:06
 6   Big Buck Bunny (Song Arrange)                                09:59
 7   Big Buck Bunny - 6 - Hare Force                              01:40
 8   Fredtun musikk 12/13: Uprising                               05:16
 9   Big Buck Bunny in 3D                                         10:35
10   Big Buck Bunny Rescore                                       02:43
11   big buck bunny stereo                                        01:41
12   AMD E350 - ASUS E35M1-I Deluxe playing Big Buck Bunny        01:17
13   Big Buck Bunny - 2 - All Ears                                02:11
14   Big Buck Bunny - 8 - The Rabbit Diaries                      00:47
15   Big Buck Bunny - 3 - Shock and Gnaw                          03:34
16   Big Buck Bunny - 5 - Daikaiju RA-BI-TO                       01:57
17   Big Buck Bunny & Alien Waves music                           05:10
18   Balla 12 - Pra que Nunca Anoiteça (Big Buck Bunny)           05:00
19   Big Buck Bunny - 1 - Prelude                                 00:48
20   Big Buck Bunny - 9 - Extras                                  00:47

Search results for big buck bunny
> _
```

It's just good old YouTube, but without all that annoying video that just gets in the way of the audio.

"Mps-youtube is a terminal app for searching videos and playing music."

PROJECT WEBSITE
https://github.com/np1/mps-youtube

Graphical maths tool

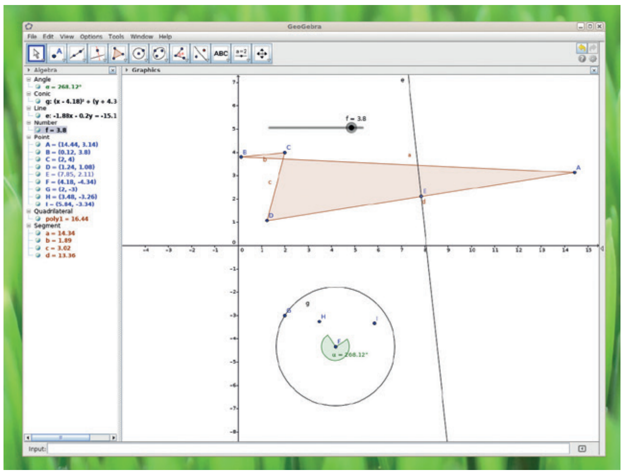
GeoGebra

Back when I were a lad you could buy a Mars bar for 25p and still have change for the bus ride home, and all this were fields, and when it came to maths class, you drew equations using graphical calculators.

Now, mobile computers – from laptops to smart phones – are seemingly everywhere, and it might be time to replace these ageing calculators. There are many plotting tools available, but *GeoGebra* stands out because it's designed for interactive exploration. That means that it's not just good for displaying data, but learning about the mathematical properties of the graphs. You can think of it less like a plotting tool and more like a mathematical play pen.

Objects are dragged-and-dropped onto the canvas, then you can manipulate the properties either by entering absolute values, or linking them through a simple language. In some ways, you could view *GeoGebra* as a mathematical graphical programming language rather than a charting tool. That said, it's almost entirely mouse-driven, so you don't get bogged down in code.

GeoGebra is structured to help teachers, and this means you can create work sheets for export to other users. There's a website – **tube.geogebra.org** – that hosts them as HTML pages so viewers only need a web browser and internet access to use them. You don't have to be a student to find useful things on the website though. Drivers can find a mathematical examination parallel parking (**http://tube.geogebra.org/student/m3022**), or take a light hearted look through a kaleidoscope (**http://tube.geogebra.org/student/m27651**). True to the earlier graphical calculators, there are even some games available, like billiards (**http://tube.geogebra.org/student/m167309**).



As well as mathematics, you can also use *GeoGebra* for idle time-wasting, as shown here.

PROJECT WEBSITE
www.geogebra.org

Low-latency voice chat

Mumble

Mumble is a bit like IRC for voice. It's built on a client-server model, so once a server's running, many people can connect to this and share their audio. Like IRC, it features channels so a single server can host many conversations. The *Mumble* project includes a client and a server (known as *Murmer*), so has everything you need to host your own chat sessions – provided, that is, you have a server to host it on. It's fairly low-resource, so should run on a low-spec VPS, though you may need more power if you're planning on hosting a lot of users.

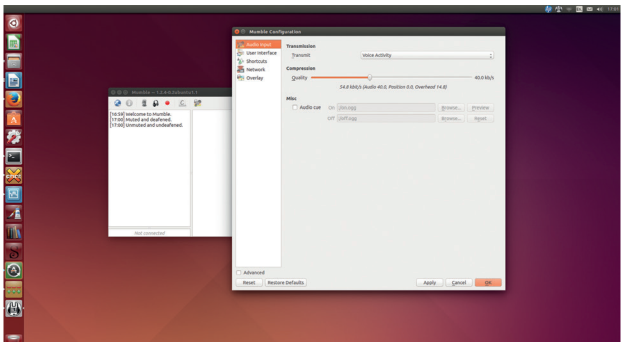
Mumble is built for low-latency, so chatting works well, and quality is good, though this can depend on the available bandwidth. You don't need to worry about who's listening in as all communications are secure by default.

It was originally designed for gamers and does have some special features to support this (such as overlays and positional audio). However, it's also useful in other areas, for example, many podcasts use it for recording.

Perhaps the most powerful part of *Mumble* isn't in the software itself, but in the ecosystem that's built around it. There's everything from web interfaces to command line clients and bots for almost every available task. There are even other servers for the *Mumble* protocol that are optimised in different ways. For a more complete list of *Mumble*-related software, check out the project wiki at **http://wiki.mumble.info/wiki/3rd_Party_Applications.**

There are clients for all major platforms, so you don't need to limit your chats to just your FOSS-loving friends. This includes iOS and Android so you can also stay chatting on the go.

You can try out *Mumble* without running your own server by finding a public server at **www.mumble.com/serverlist**.



Mumble – 1.2.4-0.2ubuntu1.1

[16:59] Welcome to Mumble.
[17:00] Muted and deafened.
[17:00] Unmuted and undeafened.

Not connected

Mumble Configuration

Audio Input / User Interface / Shortcuts / Network / Overlay

Transmission
Transmit: Voice Activity

Compression
Quality: 40.0 kb/s
54.8 kbit/s (Audio 40.0, Position 0.0, Overhead 14.8)

Misc
Audio cue On :/on.ogg Off :/off.ogg

The Mumble configurations mean you can tune the setup to your particular internet connection to maximise quality and minimise latency.

"Mumble has everything you need to host your own chat sessions."

PROJECT WEBSITE
www.mumble.com

Android photo enhancer

Effects Pro

These days, photos need to have a filter, or it's just not cool enough. We're all for jazzing up pictures, but too often it's done through proprietary web apps. Sure, *Gimp* has some great options, but if you've only got your phone with you, that's not much help.

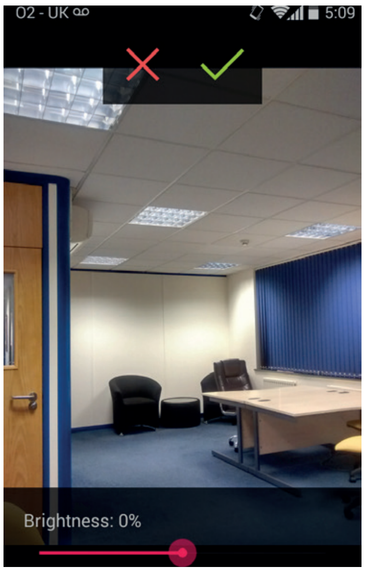
In steps *Effects Pro*. It's a simple Android app that lets you manipulate images (either new photos or ones stored on your phone). You don't need to know much about image processing – just select the effect you wish to add, adjust the amount, and *Effects Pro* does everything else.

Version 1.1 did seem to have a few stability issues on our test device (a Moto G), so we'd recommend taking the photo first, saving it, and then opening it in *Effects Pro*. This way you'll still have a copy of the image should there be a problem. Performance was also a little slower than we would like, but hopefully these minor issues will be ironed out in future releases.

There are 19 effects available, ranging from colour changes to classic photo manipulations like sepia and vignette. It's designed to be simple on a touchscreen, so doesn't have anywhere near the range of options as powerful desktop software (like *Gimp*), but with a few clicks, you can liven up most pictures.

You can get the source code from **https://github.com/yaa110/Effects-Pro** if you want to tinker with it, or the compiled software is in the F-Droid repository, so you can get it through that app store (**https://f-droid.org/repository/browse/?fdid=org.appsroid.fxpro**).

Effects Pro is simple to use, and open source: two qualities that are rare in Android apps.



This isn't the same as *Photo Effects Pro* on the Google Play Store, so if you want the free software app, make sure you get it through the F-Droid store.

PROJECT WEBSITE
https://github.com/yaa110/Effects-Pro

FOSSPICKS Brain Relaxers

Terminal-based robot deathmatch

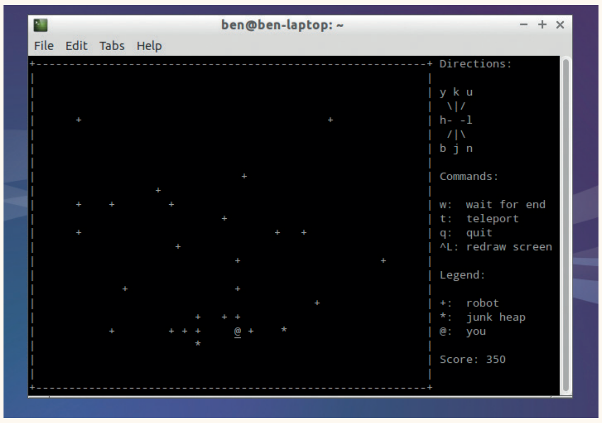
BSD Robots

What *Minesweeper* is to Windows, *Robots* is to BSD. It's the classic game perfect for wasting hours of otherwise productive time (or to run while you're waiting for something to compile). The game play is simple: you're an @ symbol surrounded by robots. Each turn you move, then the robots move. The robots don't have any collision prevention, so they're prone to colliding with each other. The aim of the game is to make all the robots collide with each other so there are none left. It gets easier, because dead robots leave scrap metal that other robots can then crash into.

If you get completely stuck and can't move anywhere (this can happen quite a lot as you're starting out), you can teleport. This moves you to another place on the screen, but it can land you in the path of a robot, so there's a chance any teleport could be fatal. Some clones have the option to safe teleport, but this corruption of the game's ideals is anathema to true *BSD Robots* aficionados.

There are other versions of the game where the protagonist is escaping zombies or Daleks, but the mechanics are the same.

As with all classic games, there are plenty of clones that are more graphically impressive – *Gnome Robots*, for instance – but for us, the classic black and white terminal version can't be beaten. If you want the full retro experience, you could run it through *Cool Old Term* to simulate a CRT display.



```
Directions:

y k u
 \|/
h- -l
 /|\
b j n

Commands:

w:  wait for end
t:  teleport
q:  quit
^L: redraw screen

Legend:

+:  robot
*:  junk heap
@:  you

Score: 350
```

You can teleport to a random place if you're really stuck, but beware, this could take you directly into the path of a robot.

GNU Robots is a completely different game, so make sure you get the right one (it's often in a package called **bsd-games**). Alternatively, you can play it in a web browser without installing anything at **http://ctho.org/games/robots**.

PROJECT WEBSITE
none

Dungeon game

Angband

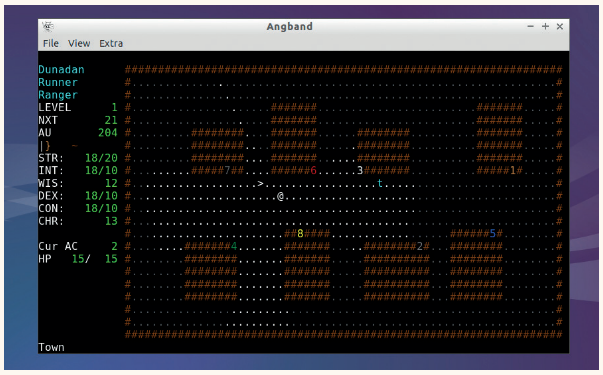
In issue 5, we posed the question: is *Nethack* the greatest game of all time? A lot of you said yes. Among the dissenters, the most popular alternative put forward wasn't some AAA game with fancy graphics, but another text-based dungeon crawler: *Angband*.

The aim of *Angband* is to delve deeper and deeper into a world of dungeons until you reach level 100 where you come face to face with Morgoth. Tolkien fans will recognise that name, and *Angband* is set in Middle Earth, though a knowledge of this isn't necessary to enjoy the game.

Levels are randomly generated, so no two games are the same. This means that you have to constantly adapt, even if you've played the level before. Carefully managing your inventory (so you're ready for any eventuality) is key to progressing in the game.

Like *Nethack*, *Angband* is turn-based, so you have time to think through your decisions. Make no mistake though, *Angband* is hard. There's a very real chance that even if you keep playing for 20 years, you still won't complete it (if you don't believe us, check out the forums on the project website). However, the counterpoint to this difficulty is that each little victory (like descending to another level) brings a huge amount of satisfaction.

So, the question remains, which is better: *Nethack* or *Angband*? Both are mind-numbingly difficult and can drive you mad at times and make you ecstatic at others. The terminal-nature of *Nethack* makes it a bit easier to run on a server, but *Angband* gets the edge from this author because the level generator seems just a little more brilliantly malevolent.



```
Dunadan
Runner
Ranger
LEVEL        1
NXT         21
AU         204
|}   ~
STR:    18/20
INT:    18/10
WIS:       12
DEX:    18/10
CON:    18/10
CHR:       13

Cur AC       2
HP    15/   15

Town
```

You can't see round corners in *Angband*, so there's always a chance of a monster waiting to attack.

PROJECT WEBSITE
http://rephial.org

flossUK

FORTHCOMING EVENTS

un-conference - *London - Saturday 7th February 2015*

London Hackspace, 447 Hackney Road, London E2 9DY

Our 5th un-conference - on-line booking opening soon.
see: http://www.flossuk.org/Events/Unconference2015

DEVOPS Spring 2015 - *24th, 25th & 26th March 2015*

The Hilton York, 1 Towers Street, York YO1 9WD

Tuesday 24th March - Workshops • Wednesday 25th & Thursday 26th - Conference

This event is the UK's only conference aimed specifically at systems and network administrators. It always attracts a large number of professionals from sites of all shapes and sizes. As well as technical talks, the conference provides a friendly environment for delegates to meet, learn, and enjoy lively debate on a host of talks.

Bookings will open in December, for more information see: http://www.flossuk.org/Events/Spring2015

OpenTech 2015 - *Saturday 13th June, ULU, University of London Union*

OpenTech will be 10 years old in 2015.

It will be back in 2015, for the usual mix of technology, experience and everything else.

For now, there's a date for your diary, and the call for talks will open soon. If there's something you'd like to hear about at OpenTech next year, we'd love to hear from you and we'll see what we can do.

OpenTech is only possible because of wonderful and generous sponsors in the past.... If you or your company is interested in Sponsoring please get in touch - opentech@opentech.org.uk

The event's predecessors were low cost, one-day conferences about technologies that anyone can have a go at, from 'Open Source' style ways of working to repurposing everyday electronics hardware.

http://www.opentech.org.uk

Dynamic Languages Conference - *Saturday 20th June 2015 Manchester*

for more information see: http://www.dynamiclanguages.co.uk

If you are not a member and want to be kept informed of future events please subscribe to '*announce@ukuug.org*' see: *http://lists.ukuug.org/mailman/listinfo/announce*

UKUUG - Silver Sponsor members include: SUSE

Why not join today? Annual Individual membership is just £42.00 inc. VAT. See: *http://www.flossuk.org/join*
As a member you can attend all our events at the specially discounted member rates and receive our quarterly Newsletter!

UKUUG Ltd. t/a FLOSS UK, The Manor House, Buntingford, Herts SG9 9AB
Tel: 01763 273475 Email: office@ukuug.org

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 get sudo powers to apt-get purge winter, but can't remember Aslan's password.

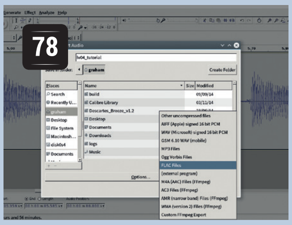
Issue 12 marks one year since we began this crazy experiment called Linux Voice. It all started with a few simple ideas: that a magazine should support the community it serves, that good publishing still has a place in a world of clickbait, and that a profitable business can share its intellectual property in a socially responsible way.

We still believe in these ideas, and we think that the past 12 months have vindicated them. Our strong subscription growth is testament to that, so Linux Voice is here to stay, and we're not going to compromise these values.

What can you expect from the next 12 months of Linux Voice? First of all, more of the same. Another 12 issues in the shops, and 12 issues released free under a Creative Commons licence nine months after they were published. Another 26 (or so) podcasts. We hope to sponsor more events (in 2014 we sponsored OggCamp and Pi Wars). If you've got any ideas for this, let us know. We'll be donating 50% of a first year's profit to organisations chosen by readers shortly.

We're also hoping to expand into other media. We can't reveal too many details yet, but in 2015 we hope to spread the word of FOSS even further. Stay tuned!
ben@linuxvoice.com

In this issue...


78

Audacity
We're looking for help with the audio version of Linux Voice. **Graham Morrison** shows you how to join this community effort.

80

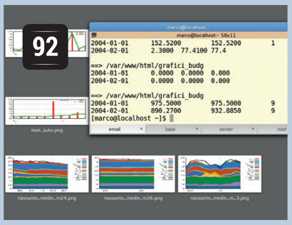
Passwords
Are your passwords secure enough? **Les Pounder** shows you how to check their strength using a Python script.

84

OpenElec
Graham Morrison installs this home theatre PC distro and converts an old computer into a state-of-the-art media centre.

88

KMail
After managing to tame this magnificent beast, **Graham Morrison** reveals the secrets to getting the most out of *KMail*.


92

Regex
If you need to search through test files, then you need regular expressions. **Marco Fioretti** shows how to get started.

96

Olde Unix
Linux stands on the shoulders of the OSes that came before it. **Juliet Kemp** takes a look at how we arrived at the modern OS.

PROGRAMMING

Arduino
100 These microcontroller boards have opened up the world of bare-metal computing to a whole new audience. With an Arduino, you can easily control almost any hardware with just a few lines of C++. In this tutorial, we look at controlling an OLED display, but the I2C protocol can be used for lots more hardware.

RESTful APIs
104 If you need to get data from an internet source, then there's a good chance you'll need to use a RESTful API. In this tutorial we look at what they are, see why they work the way they do and show you how to grab the data you need using either a normal web browser or a single line of Python code.

ASM
106 In part one of this series, we introduce the idea of programming using just the instructions that the CPU uses rather than high-level statements, and look at how assembly language works. After all, true geeks speak to the computer in its own language – compilers and interpreters are just for newbies.

LINUX VOICE TUTORIAL

VOICE RECORDINGS WITH AUDACITY

GRAHM MORRISON

Learn some new skills, get your work onto the internet and help with our crowdsourced audio editions.

WHY DO THIS?
- Help Linux Voice reach a bigger audience.
- Learn one of the basic audio applications for Linux.
- Take the first step into the world of podcasting.

When we released our first two issues under the Creative Commons CC BY-SA licences, we wanted to include audio versions of as much content as we could. Not only, we thought, would this help people who often don't have the time to sit down and read, it would help all sorts of other people who may find reading difficult. With the help of the Linux Voice community, we were able to pool around five hours of various recordings for each issue and make them available for free.

We need as much help as possible for issue three onwards, so we thought we'd cover the brief details of how you can make these recordings yourself, and if you enjoy the process and wanted to help, you can help us with our future recordings. Keep an eye on LinuxVoice.com for details.

Step by step: Record with Audacity

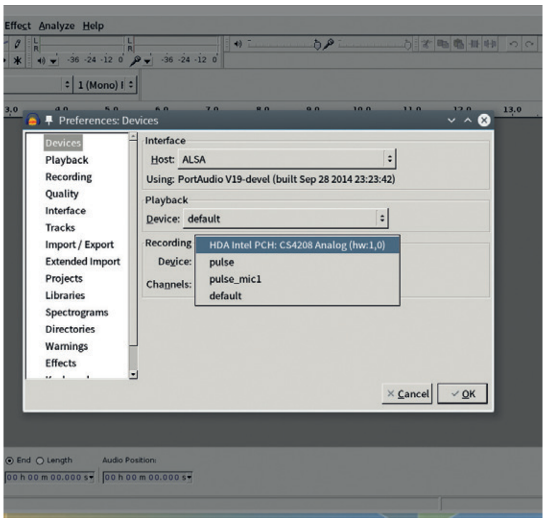
1 Install and configure Audacity

The best tool for recording and editing audio is *Audacity*. It's part of every distribution's package repertoire, it's relatively simple to use and provides all the functionality you need to make even complex audio recordings. Making any kind of recording is better with an external microphone, but it's not unfeasible with whatever's built into your system (we've had a few good recordings made with a laptop's internal mic).

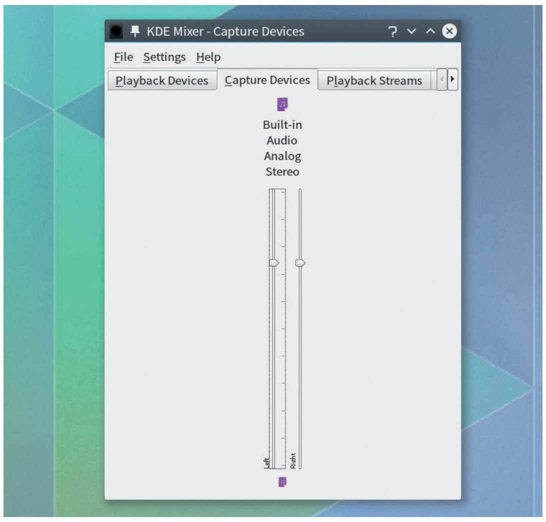
More than choice of microphone, it's more important that your recording environment is as quiet and as dead as possible. By 'dead' we mean that sound is dampened by carpet or curtains, rather than bouncing off flagstone flooring and tiling. It also helps if your computer is quiet. With all that set, open up *Audacity* and make sure your microphone is the configured input for any recording. If you're running *PulseAudio*, it's better to select the device name rather than choose one of *PulseAudio*'s inputs.

2 Get the right level

After reducing the amount of background noise, the next most important thing to get right is the input level of your recording, and your desktop audio mixer input should be ramped up. But this really means how loud you should speak and how close the microphone you should be, rather than dealing with amplification before a recording is made (which is what you might adjust if you're running a mic through a mixer). This is important because every recording has a noise floor – the level at which noise is introduced from both the environment and from the recording hardware itself, and this noise floor is the same regardless of how loud your recording is. This means that if your recording is too quiet and you boost the recording to make it louder, you'll also boost the level of the noise floor. If a recording is louder to begin with, the proportional boost to noise is smaller, making a better quality recording.



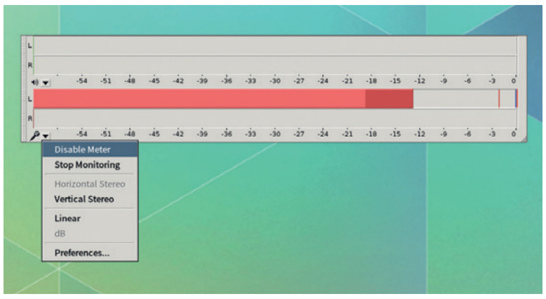
Interface
Host: ALSA
Using: PortAudio V19-devel (built Sep 28 2014 23:23:42)
Playback
Device: default
Recording
Device: HDA Intel PCH: CS4208 Analog (hw:1,0)
Channels:
pulse
pulse_mic1
default



3 Monitoring your input signal

You can monitor the input level within *Audacity* by selecting Start Monitoring from the small drop-down menu next to the microphone icon near the bar graph in the main window. When enabled, a red bar will start bouncing across the widget. If you prefer vertical bars rather than horizontal, you can drag and re-size the whole widget out of *Audacity*'s main window. You'll probably see only a single bar when there's space for two and that's because microphone inputs, including those on your computer, are nearly all mono, as opposed to stereo.

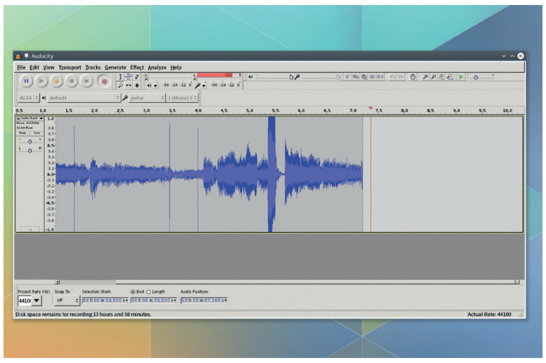
We make our recordings in mono too, as they take half the space and processing, but they can still be used to create stereo files if necessary. Make sure that your speaking voice/recording uses the full range of the monitoring bar without hitting the maximum, which for historical reasons is labelled 0(dB) . Too loud a signal is worse than too quiet a signal, because it adds ear-crunching distortion.



4 Your first recording

If everything looks good, make sure you've got a glass of water at hand and press R on the keyboard or the red button in the toolbar to start the recording. The input monitor will still bounce and a waveform showing the amplitude of your recording over time will be drawn to the right of the audio track. Glancing at this is a good way of checking everything is going OK and you should now try narrating an article.

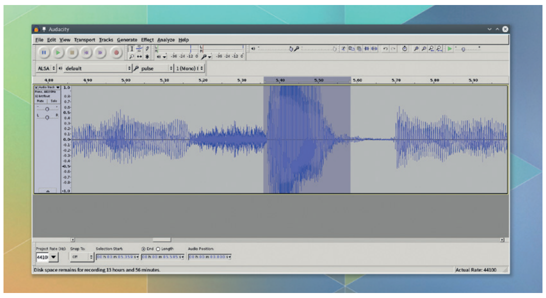
It will typically take a few tries to get something right, and also to find a pace that works for you. You'll likely start quicker and slow down as you get more used to the process. But remember you can always go back and edit what you've already recorded, or start again if you mess up too badly, so don't worry about small glitches or mistakes as you go along. A four page tutorial typically takes us around 15–20 minutes, depending on your reading speed.



5 Edit the audio

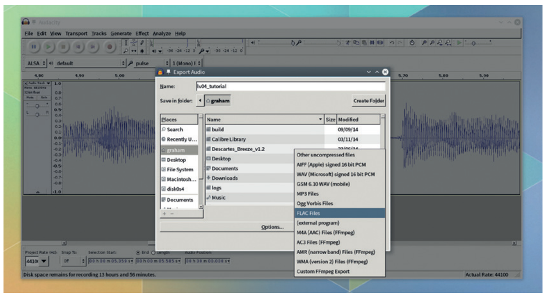
If you've made any mistakes, you can easily edit them out. They key to this is listening to the playback and seeing which bumps in the waveform correspond to the bits you want to cut. Selecting these areas and using the Edit menu or the usual shortcuts will remove them, merging the two sides of the recording bordering on your selection. You can also copy and paste areas from other parts of the recording, which is useful if you want re-record bits you've messed up at the end. Pasted audio will not overwrite audio by default, but be inserted at the cursor.

The only other processing that might be useful is to normalise the recording. This will make sure the waveform fills all the available headroom without changing the dynamics of the recording itself. To do this, select the entire recording and choose 'Normalise' from the 'Effects' menu. You can leave the settings in the window that appears at their default values.



6 Export the audio

When you're happy with the recording, it's time to share it. For issue 2, we've asked for files to be provided in the FLAC format, where possible. This is not just because FLAC is the lossless equivalent of gzip for audio, but because it doesn't process the audio in any way. This is so we can encode the files using our own scripts to ensure that all files are processed in the same way. If space or bandwidth is an issue, it's no problem providing files as MP3, OGG or even Opus either, if that's going to make it easier to put on our server. You can't use *Audacity*'s 'Save' dialog to create the audio file; instead you have to use 'Export Audio' from the File menu. Use the drop-down menu beneath the file list to choose FLAC, and you don't need to adjust any other options. You'll also be asked to enter some metadata for the recording, but we'll replace this with our own scripts. All that's then left is to save the file and send us the link! Thanks!



LINUX VOICE TUTORIAL

PYTHON 3: BUILD A PASSWORD CHECKER

LES POUNDER

Use programming logic, variables and functions to check the strength of your passwords.

WHY DO THIS?

- We will learn how to accept user input via a graphical user interface and then check the input against an algorithm to ensure that the password input meets a criteria.

TOOLS REQUIRED

- Python 3 installed on your machine.
- EasyGUI installed on your machine.

DISCLAIMER

This project should not be used to check any "real world" passwords for example online banking or shopping.

Keeping our data private is a major concern, and one way to keep our data safe is via a good password. But being human we rarely choose a secure password; instead use insecure data such as the name of our pet, our date of birth or other such details. Data such as this is not really suitable in a password, and we are encouraged to create harder to guess passwords with a mix of both upper and lower case characters along with numbers.

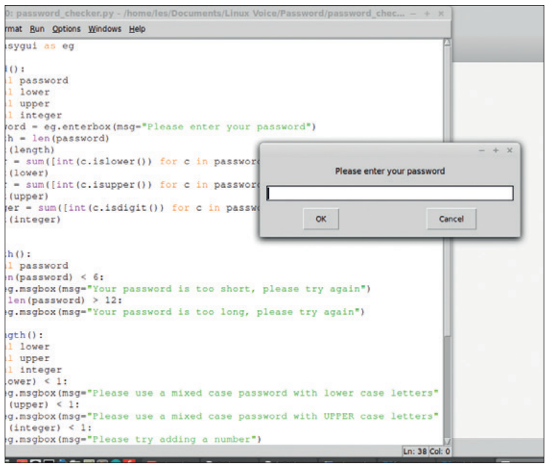
In this project we will be using the popular programming language Python, and we will be writing the code using the Python 3 syntax, which also enables this project to be run on Python 2 systems. This project can be created using any computer, including a Raspberry Pi.

For this project you will need

- **Idle 3** A Python editor for Python 3
- **pip3** A Python package manager
- **EasyGUI** A GUI library for Python

For this tutorial we're using Linux Mint 17, which is based on Ubuntu, and so we will install software using the **apt** package manager. The software is also available for other Linux distributions.

To install *Idle 3* open a terminal and type in the following.



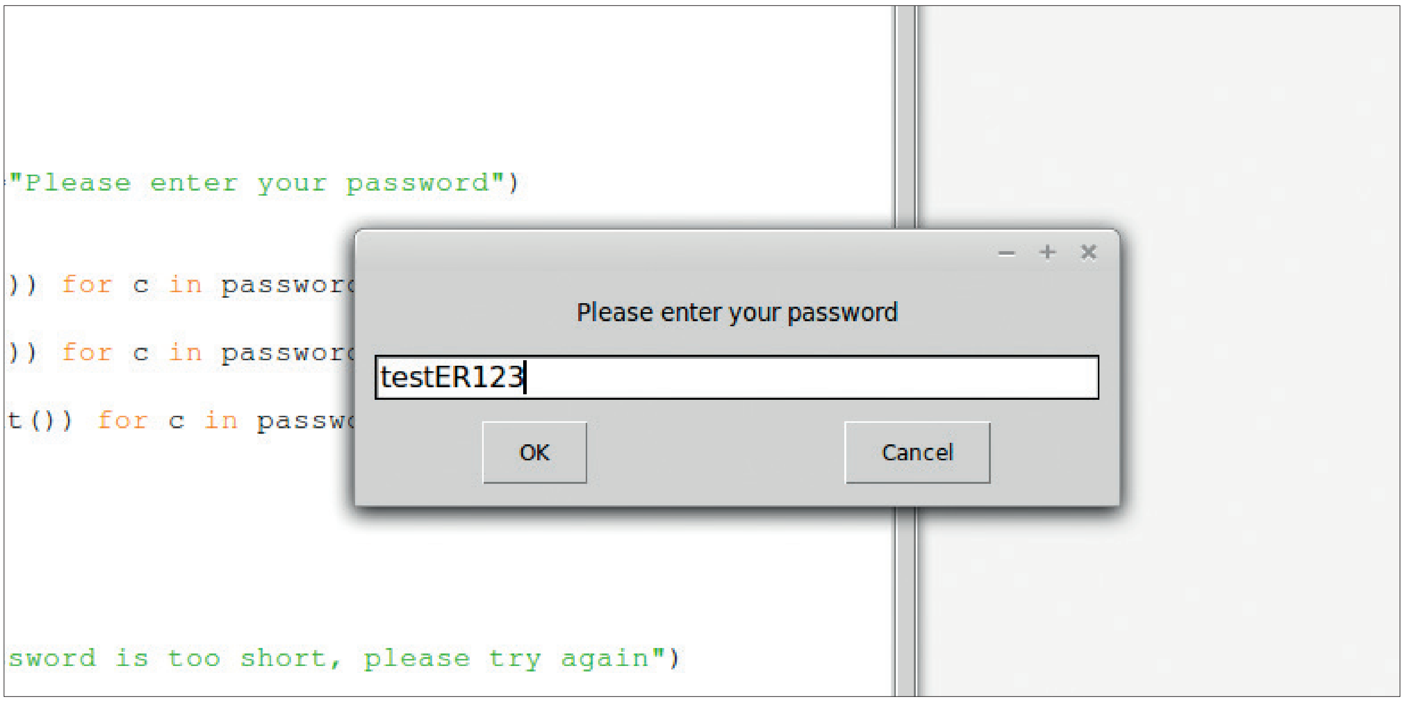
```
0: password_checker.py - /home/les/Documents/Linux Voice/Password/password_chec...
rmat  Run  Options  Windows  Help
sygui as eg

l():
l password
l lower
l upper
l integer
ord = eg.enterbox(msg="Please enter your password")
h = len(password)
(length)
 = sum([int(c.islower()) for c in passwor
(lower)
 = sum([int(c.isupper()) for c in passwor
(upper)
er = sum([int(c.isdigit()) for c in passw
(integer)


h():
l password
n(password) < 6:
g.msgbox(msg="Your password is too short, please try again")
len(password) > 12:
g.msgbox(msg="Your password is too long, please try again")

gth():
l lower
l upper
l integer
ower) < 1:
g.msgbox(msg="Please use a mixed case password with lower case letters"
(upper) < 1:
g.msgbox(msg="Please use a mixed case password with UPPER case letters"
(integer) < 1:
g.msgbox(msg="Please try adding a number")
```

The password checker in action. Our simple application has but one job to do: to keep our passwords strong.

```
sudo apt-get install idle3
```

New to Python 3 is the bundling of the **pip** Python package manager, which works in a similar manner to other Linux package managers. This should be installed by default for your distribution, but if this is not the case open a terminal and type in the following.

```
sudo apt-get install python3-pip
```



```
"Please enter your password")

)) for c in passwor
)) for c in passwor
t()) for c in passw

sword is too short, please try again")
```

Using *EasyGUI* we can easily create a graphical user interface to capture the user's password.

With **pip** for Python 3 installed now we use it to install *EasyGUI*, in a terminal type in the following

```
sudo pip3 install easygui
```

After a few moments *EasyGUI* will be installed and ready for use in our project.

The project

In this project we use Python 3 to write a program that captures the user's password and checks it against a series of conditions. These conditions are:

- The password must have six or more characters.
- The password must have less than or equal to 12 characters.
- There must be at least one integer in the password.
- There must be at least one upper case character.
- There must be at least one lower case character.

We start the project by importing additional modules to expand the abilities of our Python code.

```
import easygui as eg
```

In this example we import the **easygui** module and rename it as **eg**, which makes it a lot easier to work with. With the imports completed we now create a series of functions that handle the input and testing of the password.

Firstly we create the function and name it **pword()**:

```
def pword():
```

Next we create four global variables that are used by our function and other functions later in the project. These global will contain the data for our project.

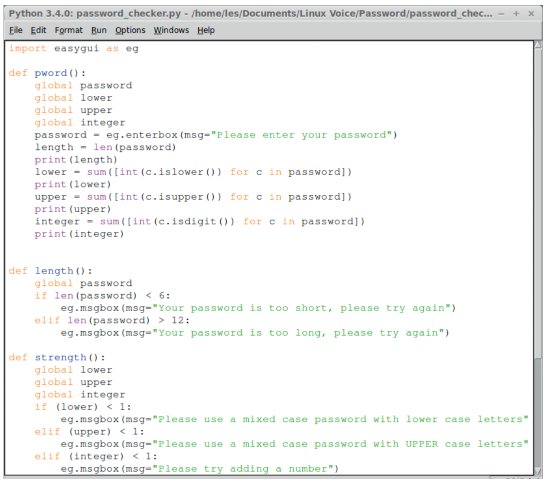
```
  global password
  global lower
  global upper
  global integer
```

With the variables created we now use one of the *EasyGUI* functions to create a dialog box that will enable us to enter our password and then save it as the global variable **password**.

```
  password = eg.enterbox(msg="Please enter your password")
```

With the password captured and stored as a variable we can now do a lot of checks.

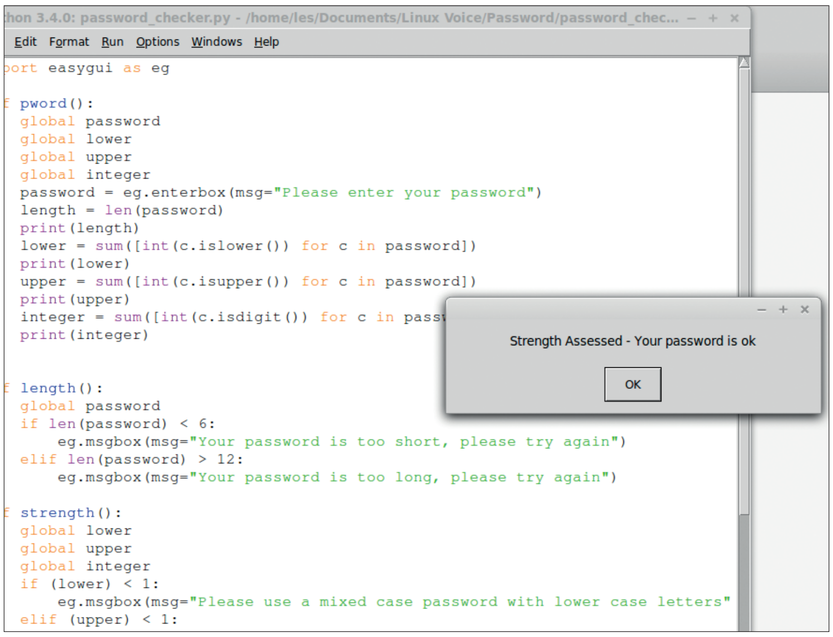
```
  length = len(password)
```



Python 3.4.0: password_checker.py - /home/les/Documents/Linux Voice/Password/password_chec...

```
import easygui as eg

def pword():
    global password
    global lower
    global upper
    global integer
    password = eg.enterbox(msg="Please enter your password")
    length = len(password)
    print(length)
    lower = sum([int(c.islower()) for c in password])
    print(lower)
    upper = sum([int(c.isupper()) for c in password])
    print(upper)
    integer = sum([int(c.isdigit()) for c in password])
    print(integer)


def length():
    global password
    if len(password) < 6:
        eg.msgbox(msg="Your password is too short, please try again")
    elif len(password) > 12:
        eg.msgbox(msg="Your password is too long, please try again")

def strength():
    global lower
    global upper
    global integer
    if (lower) < 1:
        eg.msgbox(msg="Please use a mixed case password with lower case letters"
    elif (upper) < 1:
        eg.msgbox(msg="Please use a mixed case password with UPPER case letters"
    elif (integer) < 1:
        eg.msgbox(msg="Please try adding a number")
```

There isn't a great deal of code to this project but it is a great meaty project to reinforce your learning and brush up your skills.



hon 3.4.0: password_checker.py - /home/les/Documents/Linux Voice/Password/password_chec...

```
port easygui as eg

f pword():
  global password
  global lower
  global upper
  global integer
  password = eg.enterbox(msg="Please enter your password")
  length = len(password)
  print(length)
  lower = sum([int(c.islower()) for c in password])
  print(lower)
  upper = sum([int(c.isupper()) for c in password])
  print(upper)
  integer = sum([int(c.isdigit()) for c in pass
  print(integer)


f length():
  global password
  if len(password) < 6:
      eg.msgbox(msg="Your password is too short, please try again")
  elif len(password) > 12:
      eg.msgbox(msg="Your password is too long, please try again")

f strength():
  global lower
  global upper
  global integer
  if (lower) < 1:
      eg.msgbox(msg="Please use a mixed case password with lower case letters"
  elif (upper) < 1:
```

Strength Assessed - Your password is ok

If your password meets the criteria then it will be evaluated as a strong password, an essential part of keeping you safe.

```
  print(length)
```

Our first check is get the length of the password, which is done using the **len()** function and giving it the name of the variable that we would like to check. The output of this is stored as the variable length, which we then print to the Python shell. We do not need to print the output to the shell, but it can really help to debug a project quickly.

"Our first check is to get the length of the password, with the len() function."

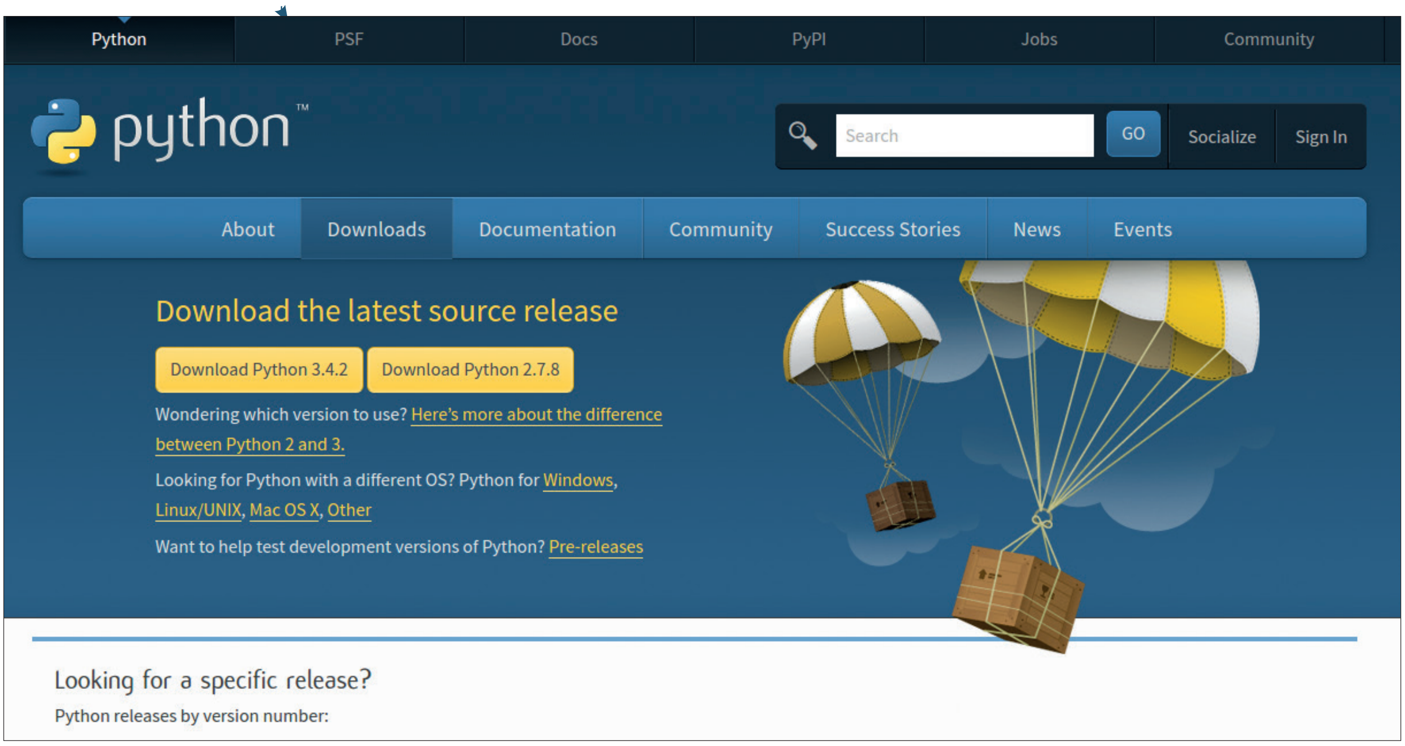
Now that we know how long the password is we need to find out what the password is made of, which will generally be a mix of lower- and upper-case letters along with numbers. We create three variables: **lower**, **upper** and **integer** and they handle lower-case and upper-case characters, and **integers** relates to any numbers in the password. For each of these variables we do the following.

```
Count every lower-case letter in the password and save it as a variable called lower.
Count every upper-case letter in the password and save it as a variable called upper.
Count every integer in the password and save it as a variable called integer.
```

In Python code it looks like this – after each line we also print the value as a debug measure.

```
  lower = sum([int(c.islower()) for c in password])
  print(lower)
  upper = sum([int(c.isupper()) for c in password])
  print(upper)
  integer = sum([int(c.isdigit()) for c in password])
  print(integer)
```

If you are familiar with spreadsheets you will see that the line is not too dissimilar to the syntax needed for *LibreOffice* or *Excel*. The term **c** refers to each character in the password, and we can instruct Python to look for characters that match **islower** **isupper** or **isdigit**. We wrap the value returned in a



Download the latest source release

Wondering which version to use? Here's more about the difference between Python 2 and 3.

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases

Looking for a specific release?

Python releases by version number:

Getting the latest version of Python 3 is really simple – you can download it via your distribution's package manager or download the latest package from the Python website.

helper function that converts the data type returned into an integer value **int(c.islower())**. This is not strictly necessary as the value returned is already an integer, but it helps to sanitise the data just in case anyone tries to intentionally break the project. Data sanitisation is best practice and a great skill to learn, as it is used a lot when working with content on the web and in databases.

Our next function has one purpose: to check the length of the password against a strict set of criteria. Its length must be greater than or equal to six characters but it must also be less than or equal to 12 characters in length. A good password should be of a reasonable length, but a long password does not equal a secure password.

In the code snippet following we first define a new function called **length()** and then instruct Python that we wish to use the global variable **password** that we created in our first function.

```
def length():
  global password
```

Next we create an **if**..**elif** conditional statement that will first check to see if the length of the password is less than six characters. If this condition is true we will use EasyGUI's **msgbox** function to generate a pop-up dialog box that advises the user that their password is too short and that they should try again.

```
  if len(password) < 6:
    eg.msgbox(msg="Your password is too short, please try again")
```

Similar to the previous condition we tested, we now use **else if**, which is shortened to **elif** in Python to run another test on the password, which in this case is to see if the password is longer than 12 characters. If this is the case, a dialog box pops up to advise the user to try again.

```
  elif len(password) > 12:
    eg.msgbox(msg="Your password is too long, please try again")
```

Our last function runs a series of tests against the password. These tests are there to assess that the password meets our criteria that we defined earlier in the project.

To start we name our function **strength()** and instruct Python that we wish to use the global variables that we created earlier.

```
def strength():
  global lower
  global upper
  global integer
```

We now reuse an **if..elif** conditional statement that we used in our **length()** function to run three conditional tests.

Firstly we compare the value stored in the variable lower to see if it is lower than 1, which means there are no lower-case characters in the password. If this is correct then we use the *EasyGUI* **msgbox** function to create a dialog box that informs the user that there are no lower-case characters in their password.

```
  if (lower) < 1:
    eg.msgbox(msg="Please use a mixed case password with lower case letters")
```

After the lower-case character test is complete we

Project code

You can find the complete code for this project at our GitHub repository **https://github.com/lesp/LinuxVoice_Issue12_Password_Checker**. For those of you unfamiliar with *Git* you can download the complete package as a Zip file from **https://github.com/lesp/LinuxVoice_Issue12_Password_Checker/archive/master.zip**

Python 3

Python has come a long way since its début in 1989, when it started as the personal project of a chap called Guido van Rossum, who needed a project to keep him busy over the Christmas period. In the passing years Python has improved with each release, and since around 2001 Python 2 has been the default version. In recent years, however, there has been a strong move towards leaving the Python 2 series and moving on to version 3, which has been supported by such organisations as the Raspberry Pi Foundation.

But why should we move to Python 3? Well the most obvious reason is that the Python core developers are no longer working on any code or projects for Python 2. In fact they are so focused on Python 3 that the Python team have created an "un-release schedule for Python 2.8", or in other words, Python 2.7 series is the last of that series. There will be bugfixes and updates for Python 2 for a few years yet, but now is the time to migrate your projects to Python 3.

There are a few changes to Python 3 syntax for example, in Python 2, print is a statement, and will pick up whatever is inside the quotation marks, like so:

```
print "Hello World"
```

In Python 3, print is now a function that comes with a number of arguments to make it a lot more functional, including options to control how content is separated, ended and error control.

```
print("Hello World")
```

User input

In Python 2 **raw_input** is used to capture the keyboard input via the Python shell.

```
raw_input("What is your name?")
```

In Python 3 this has been renamed as **input** but it performs the same actions as **raw_input**.

```
input("What is your name?")
```

Working with lists

In Python 2, working with a list called "names" we pass it two strings, in this case the name of my dog and I. To check the contents we print the list to the shell. Now using del we delete the contents of the list and then use print to check that the list has been deleted.

```
names = ["Les", "Dexter"]
print(names)
del(names)
print(names)
```

In Python 3 lists have been refined with functions that handle printing the contents of the list, in this example **names.copy()** prints the contents of the list. To clear the list we use the **names.clear()** function.

```
names = ["Les", "Dexter"]
names.copy()
names.clear()
names.copy()
```

These are but a few changes made to Python 3. Head over to **https://docs.python.org/3** for a full list of all the changes and additions to Python.

now run the same test looking for upper-case characters.

```
elif (upper) < 1:
    eg.msgbox(msg="Please use a mixed case password with UPPER case letters")
```

Our last test is exactly the same, but this time we are looking for integers in our password.

```
elif (integer) < 1:
    eg.msgbox(msg="Please try adding a number")
```

To close the **if..elif** conditional statement we use an **else** condition which requires no condition to be used. For example, if all of the previous statements evaluate as false then our **else** condition must be true.

```
else:
    eg.msgbox(msg="Strength Assessed - Your password is ok")
```

With our last function completed our attention now turns to threading the functions together into a sequence.

Firstly we instruct Python that we wish to run the following code in an infinite loop, which in Python is **while True**:

```
while True:
```

Inside our infinite loop we call the three functions that we created earlier.

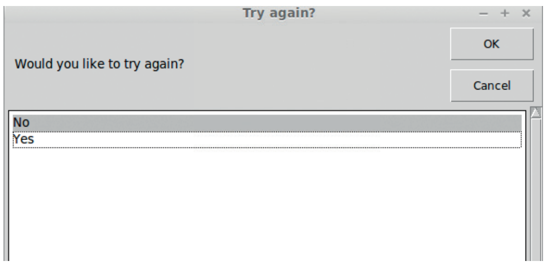
```
pword()
length()
strength()
```

With the functions called and their contents executed, the next part of the loop is a question to the user, asking if they would like to run the program again, and their answer to this question is captured using *EasyGUI*'s **choicebox** dialog. Their choices are limited to Yes and No via the choices argument. The answer is then stored inside a variable called answer.

```
answer = eg.choicebox(title="Try again?", msg="Would you like to try again?", choices=("Yes","No"))
```

In our last section of code we use an **if** statement to check the value of the variable answer. If it does not match "Yes", which in Python can be written as **!=**, then the loop is broken via break and the program ends.

```
if answer != "Yes":
    break
```



Would you like to try again?

No
Yes

The program is designed to loop infinitely enabling many passwords to be checked.

What have we learnt?

In this project we have used Python 3 to develop a simple program which evaluates a password.

> **"These tests are there to assess that our password meets the criteria that we defined earlier."**

- We provided a graphical user interface to capture the password.
- We created a function to evaluate the contents.
- We tested using programming logic to ascertain the number of lower- and upper-case characters along with the number of integers present.
- We used conditional logic to compare the actual results against what we expected to find and where they were different we advised the user as such.

Les Pounder divides his time between tinkering with hardware and travelling the United Kingdom training teachers in the new IT curriculum.

LINUXVOICE TUTORIAL

GET A LINUX-BASED MEDIA CENTRE WITH OPENELEC

GRAHAM MORRISON

Now that version 5.0 is here, there's never been a better time to install this award-winning Kodi-based media player.

WHY DO THIS?

- Install, configure and use the best Linux-based movie, music and photo viewer you can connect to your television.

TOOLS REQUIRED

- A low-powered PC, a Raspberry Pi or an ARM-based system such as the Matrix TBS 2910. You'll also need a 1GB USB stick and some storage or media files accessible from a network. The system also works best when connected to a television.

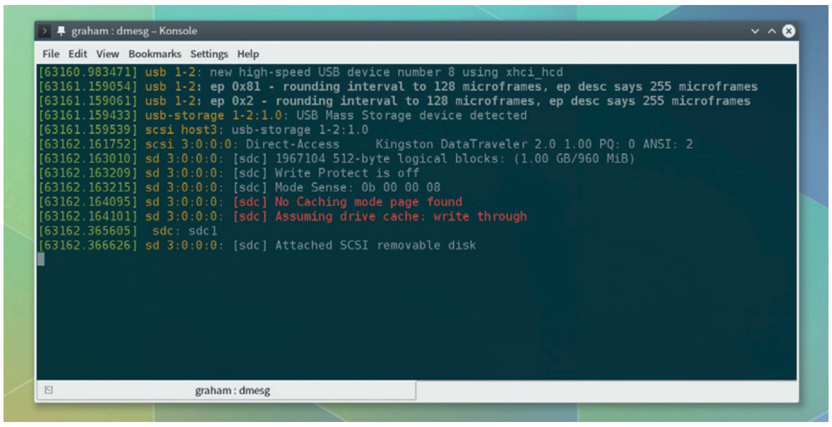
There are many distributions designed to deliver the ultimate multimedia experience to your living room. Even devices like Google's Chromecast are trying to muscle in on the action. But our favourite is a tiny yet brilliantly constructed distribution called OpenELEC, and version 5.0 has just been released, giving us the perfect excuse to write a guide on getting the most out of it.

Unlike the vast majority of distributions, including those designed for media playback, OpenELEC has one purpose and one purpose only – it's a distribution designed to run a single application, *Kodi*. *Kodi* is the television-recording, music-playing, video-managing, plugin-hosting media monster that used to be called *XBMC*. And the best thing about OpenELEC is that it's built from scratch to cut everything out that isn't absolutely necessary for running *Kodi*. That means it's not a good choice if you want to install other things alongside *Kodi*, but it's perfect if you want to get the best out of whatever diminutive hardware you want to connect to your television and amplifier. It has support for Nvidia, AMD, Intel and Broadcom hardware decoders typically found in embedded chipsets, for example, and runs brilliantly on low-powered Atom, Fusion systems and the Raspberry Pi, as well as just about any PC from the last 10 years.

1 CREATE THE INSTALLATION MEDIUM

Just because OpenELEC runs well on all these platforms doesn't mean you can't install it on a normal PC, and the install instructions for both are very similar. To prove this, we're going to include instructions for installing them both. After installation is out of the way though, OpenELEC behaves almost identically on whichever platform you choose, whether that's a PC or a Raspberry Pi. To get started, go to the download page at **http://openelec.tv**. If you're going to install on a PC, download the tar version of the generic build – 64-bit for a modern system, or 32-bit for older systems. 32-bit is being deprecated for the next major release of OpenELEC, which is worth considering if you build your installation with other hardware. We'd recommend grabbing the equivalent tar version if you're installing onto a Raspberry Pi too. At the time of writing, this file is called **OpenELEC-RPi.arm-5.0.0.tar**.

As OpenELEC typically installs via a USB stick, you'll need to make sure you run the installer against the correct device.



```
[63160.983471] usb 1-2: new high-speed USB device number 8 using xhci_hcd
[63161.159054] usb 1-2: ep 0x81 - rounding interval to 128 microframes, ep desc says 255 microframes
[63161.159061] usb 1-2: ep 0x2 - rounding interval to 128 microframes, ep desc says 255 microframes
[63161.159433] usb-storage 1-2:1.0: USB Mass Storage device detected
[63161.159539] scsi host3: usb-storage 1-2:1.0
[63162.161752] scsi 3:0:0:0: Direct-Access     Kingston DataTraveler 2.0 1.00 PQ: 0 ANSI: 2
[63162.163010] sd 3:0:0:0: [sdc] 1967104 512-byte logical blocks: (1.00 GB/960 MiB)
[63162.163209] sd 3:0:0:0: [sdc] Write Protect is off
[63162.163215] sd 3:0:0:0: [sdc] Mode Sense: 0b 00 00 08
[63162.164095] sd 3:0:0:0: [sdc] No Caching mode page found
[63162.164101] sd 3:0:0:0: [sdc] Assuming drive cache: write through
[63162.365605]  sdc: sdc1
[63162.366626] sd 3:0:0:0: [sdc] Attached SCSI removable disk
```

OpenELEC downloads look different to those for typical distributions, as they take the form of a tar file or as a device image (**.img**). This is because the installation medium is assumed to be an external USB stick or SD Card, rather than the antiquated ISO of optical media. This is only viable because OpenELEC is so small – 146MB for the generic PC version, and 105MB for the Raspberry Pi images. That means you only need a small USB thumb drive or card (for the Pi) to fit the installer. We opted for the tar file because they're easier to install onto your USB stick or SD Card, but you'll need to extract the contents of the tar archive first, either from your desktop or by typing **tar xvf filename** on the command line. The tar file itself contains a few scripts, one of which will format and install all the necessary files automatically.

Running this installer against an external device will require us to run the gauntlet of making sure you're copying files to the correct device identifier and not your PC's hard drive. If you end up copying files to the hard drive by mistake, you will lose data, so it's important to get this part correct. We'd recommend opening up the command line and typing **dmesg -w**. This will open the last few entries from your system log and automatically display further messages as they appear. If you now insert your USB thumb drive or SD card, you'll see the system update to accommodate the new device, hopefully outputting something similar to the following in the system log:

```
[63162.365605]  sdc: sdc1
```

This is telling us that the new device is called **sdc** (along with a single partition on this device called

sdc1), and we're going to use this device as the destination for the OpenELEC installer. Back on the command line, press Ctrl+C to escape from **dmesg** and switch to the untarred OpenELEC folder. Be warned, the following command will erase whatever is on **sdX** and replace it with either the OpenELEC installer for PCs, or the full installation for the Raspberry Pi, so you must make sure you replace **sdX** with the correct device name from the output of **dmesg**. Here are the commands for either the PC installer or the Raspberry Pi:

```
# FOR GENERIC PCS
sudo ./create_installstick /dev/sdX
# FOR RASPBERRY PI
sudo ./create_sdcard /dev/sdX
```

As we're using **sudo**, you'll be first asked for your user password. (Switch to your root account if your distro doesn't use **sudo**.) The installer should finish within seconds; if you get an error complaining that there's no **mkfs.vfat** tool installed, you'll need to install your distribution's equivalent to the **dosfstools** package first and try again. PC users will now need to navigate a brief installation step, while Raspberry Pi users move directly to the OpenELEC startup wizard.

2 RUNNING THE INSTALLER ON A PC

Now that the installer has itself been installed onto your USB stick you now need to boot your OpenELEC machine off this. Most will do this automatically, but you may need to use your BIOS boot menu to choose the USB device, or enable USB booting from the BIOS. The sign of success is the MAIN MENU screen, from which you get to select between five options. Choose option 1 – Quick Install of OpenELEC.

The next step will ask you for a drive to install to. Be warned – the installer is going to overwrite the drive and remove whatever might have been there before. You're next asked whether SSH should be enabled from the start, and we'd recommend saying 'Yes' before you're finally warned twice more than all data on your chosen storage medium is going to be overwritten. The installer only takes a few moments to copy over 100MB of files before you're returned to the previous menu, with no indication of whether the process has been a success. Select 'Reboot' to find out. With a bit of luck, you'll soon be presented with the OpenELEC Welcome screen and startup wizard.

The startup wizard is as simple as clicking past a few questions. If you're using a wireless connection, you'll need to configure the access point settings, and Raspberry Pi users might want to enable SSH and/or Samba from the fourth step. This will take a few moments as the additional packages are installed. After that, you'll be dropped to the default XBMC/*Kodi* side-scrolling interface, from where you can now start using your new media player.

If you've used *Kodi*/*XBMC* before, the experience from OpenELEC is identical. But OpenELEC also includes its own configuration panel which is integrated into *Kodi*'s own settings menu. You can get to this by going to the 'System' menu option and cursoring down to the 'OpenELEC' submenu option. Most importantly, and a huge advantage for OpenELEC users, is that you can update both *Kodi*and the OpenELEC distribution from the first settings page. This option is set to manual by default, but you can switch this to automatic so that updates are installed without you needing to check manually.

3 REMOTE CONTROL

For installation and testing, we're assuming you've got a keyboard connected to your media device. This is a good idea, because it's much easier to enter IP addresses and network share details when you can type them on a QWERTY keyboard. But a keyboard isn't going to be too family-friendly, which means at some point you'll need to disconnect the keyboard and configure a remote control. If you've got an Android phone or tablet, the easiest option is to use the official app. Although it's still called *XBMC Remote* and hasn't been updated for a while, it's free and works well. Before it will work with your *XBMC* installation, you need to make sure 'Allow control of Kodi via HTTP' is enabled in the Webserver page of the configuration panel, then add the IP address of your OpenELEC installation to the *XBMC* app.

You can get the IP address by going to the OpenELEC Specific configuration panel and selecting the 'Connections' page. The address will be listed on the right, and you'll need to use the menu in the Android app to add the new client. You'll also need to make sure your router gives the same IP address to

With a generic USB infrared receiver, remote controls such as those for *Windows Media Centre* will work perfectly, or you could use the free Android app.

your media centre each time it boots, otherwise your remote or any other service you add to the setup, won't find the installation. Now when you go back to the remote app, you'll be able to navigate around *Kodi* using your phone just as you would with a typical infrared remote.

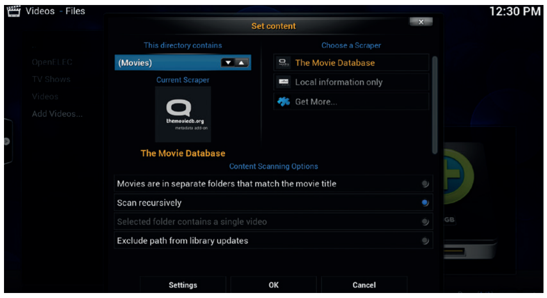
The final option we'd recommend is the best, and that's to configure a real remote. For this to work, you'll need a cheap infrared receiver (you could even make your own using the GPIO pins of a Raspberry Pi). The software that accepts keypresses and translates them into codes that Kodi can understand is called *Lirc*. OpenELEC installs and pre-configures *Lirc* to work with a few popular remotes out of the box, and that's what we'd recommend doing to get started as quickly as possible, using a Logitech Harmony remote, for instance, or a *Microsoft Windows Media Centre* remote, both of which will work without further configuration.

LV PRO TIP
If you want an even easier Raspberry Pi installation method, OpenELEC can also be installed from *NOOBS*.

4 PLAYING MEDIA

The quickest and easiest way to celebrate your installation success is to watch a film, listen to some music or browse some photos, and the easiest way to do any of these is by putting your files onto a USB stick and connecting this to your new media centre. *Kodi* should offer you a choice of 'Mounted Removable Hard Drive' which you can access by cursoring across to Video playback, to choose one media format, and selecting files. This opens *Kodi*'s file requester. The first time you run this you'll be reminded that important sources are hidden on the right of the display. When this window is closed, you should see your USB device listed, and selecting this will display the list of media files *Kodi* has detected. Playback is as simple as selecting one and pressing Enter. Use the keyboard shortcuts to control playback and return to the main GUI.

But for most devices, playing media off the network is more useful than accessing files on local storage, and *Kodi* is brilliant at doing this. Instead of selecting the storage device from the file requester, select the 'Add' button (whether you're wanting to add videos or music). This will open the 'Add Source' requester. Pressing 'Browse' will open a share window listing all the protocols *Kodi* currently speaks, including direct connections to some hardware, such as HDHomerun devices and network shares. To add a SAMBA folder being shared on your network, for example, choose the 'Windows Share (SMB)' option. The window will update to display any detected share networks, from where you'll be able to add the folder you're looking for. Finally, add a username and password if you've protected the share on the network and make sure you select 'Remember For This Path' if you don't want to go through the process again. To add the share, tab over to the 'OK' button on the right, give the share a name and select OK again. A final window called 'Set



This directory contains: (Movies)

Current Scraper: The Movie Database

Choose a Scraper: The Movie Database / Local information only / Get More...

Content Scanning Options:

- Movies are in separate folders that match the movie title
- Scan recursively ●
- Selected folder contains a single video
- Exclude path from library updates

***Kodi* can scan local and remote collections of files, but you'll need to add each source manually.**

Content' will appear, and if you select what kind of content your folder contains, *Kodi* will download covers and other details for your media.

Media visualisation

Now when you go to your media, you'll be able to go through the 'Library' option and select the share you've just added. To be able to see this content, you need to switch from the file list view to one of the image views. With movies, for instance, press the left cursor to open the view options and press Enter with the 'View' option selected. This will switch between the various viewing modes. 'Thumbnail' will show movie covers, for example, whereas 'Poster Wrap' lines up every film cover on a long horizontal line, a little like the Netflix user interface.

Other modes include online film ratings and even fan art, so it's worth experimenting with whatever works best. If you don't like having to navigate through the title or genre list before getting to your content, you can disable this option from the Library page of *Kodi*'s Settings menu, which we'd recommend if you've only got a relatively small library. The same facilities are offered for your music collection too.

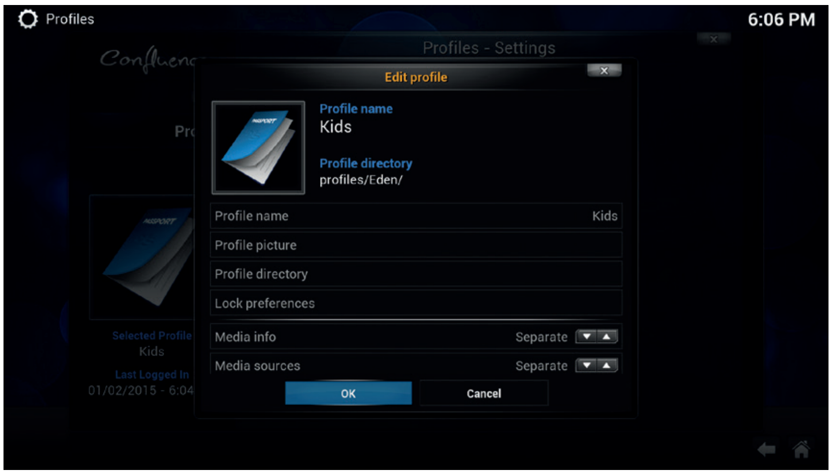
5 PARENTAL CONTROL

Often, younger members of your family are going to use your media centre and you don't necessarily want them having complete access to your media library. While there are no specific parental/content controls over who can access your media, or restricting media to certain age ranges, there are a few features that will restrict access in much the same way. The most useful of these is Profiles. Profiles are created by going to the System > Profiles submenu. A Master user is created by default, and you simply select 'Add Profile' to create a new one. The two important options are to set 'Media Info' and 'Media Sources' to

Separate. This will force you to add separate media sources for the new profile through the Music and Video menus. To switch to a new profile, go down to the power button icon at the bottom of the display and select the 'logout' option. This will re-load *Kodi* with the new access permissions.

Place content for this profile in a subdirectory of the main content folder and add this as a source. The profile's user won't be able to access the other files, yet the master user will still be able to if the 'Recursively Scan' option was enabled when you added the source. After doing this, go back to the profiles editor and change the 'Media Info' and 'Media Sources' options to 'Separate (locked)' which means the profile user can't add their own content or sources.

Of course, the profile user is still able to edit their profile themselves, and the solution to this is to create a master lock. A master lock can be used to block access to everything, using either a button combination of a pin to unlock access to the master account so that changes can be made. This can be done from your new user's profile page by selecting the 'Lock Preferences' option. If you don't want to block content but want to block the ability to change settings, you can even do the same with the master account profile. Selecting this option opens another



Profile name
Kids
Profile directory
profiles/Eden/

Profile name	Kids
Profile picture	
Profile directory	
Lock preferences	
Media info	Separate
Media sources	Separate

There's no access control based on age suitability of content, but you can create profiles with access only to specific media content

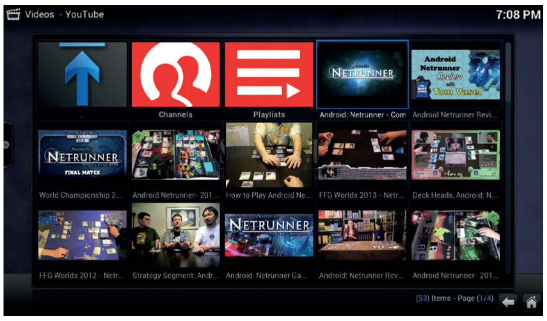
window that first lists what kind of block you'd like to use followed by the various elements of *Kodi* you can block. As a minimum, you might want to block programs & scripts, the file manager 'All' settings and the add-on manager. Locking the movies, photos and music windows locks access to the content itself, other than the titles that are listed as recently added. This could be useful if, rather than create different folders for your media, you started playback of one film and locked access afterwards.

6 PLUGINS

We've kept OpenELEC and *Kodi*'s best feature until last, and that's the way you can extend its capabilities with plugins. For photos, videos and music, you can access content-specific plugins from the 'Addons' submenu that appears when you select the content type. But each usually accesses the same plugin list, as does the Addons menu that opens from the Settings page. To install one, just select and press Enter. The addon will download and install. Some plugins not connected may need to be configured, and this is accomplished by selecting the plugin again and filling out whatever settings are required.

Adding one of these, such as the YouTube addon from Videos, will normally create a new menu entry to the 'Add-Ons' menu, but this does depend on what each plugin is doing. Selecting a plugin and pressing the I key will open the configuration panel for an add-on even if nothing was required when it installed. With YouTube, for example, you can set the quality of playback or the cache size.

Despite the long list of plugins that can be installed with no further configuration, there are plenty of others that can be installed with a little manual intervention. They're normally not included for fear of instability or because they may have some geographical restrictions, and the iPlayer plugin is a good example of this. It enables you to access all of the BBC's streaming content from your OpenELEC media centre but it will need to be downloaded and installed manually. Fortunately, this is almost as easily accomplished as installing bundled plugins. Grab the latest release of the plugin as a Zip file (see **kodi.wiki** for the link), place this file onto a USB stick and connect this to your OpenELEC box. To install from the Zip file, go to the 'Get More' add-ons link, and rather than scrolling through the list of available plugins, select the '..' symbol that represents going up a folder, first to take you to all add-ons and then to the global add-ons menu. From here you'll be able to select 'Install From Zip File' and from the requester that appears, select your USB stick and navigate to your downloaded plugin. It will then install and configure itself just like any other plugin.


Videos - YouTube
7:08 PM
Channels
Playlists
Android: Netrunner - Com
Android Netrunner Revi...
World Championship 2...
Android Netrunner - 201...
How to Play Android Ne...
FFG Worlds 2013 - Netr...
Deck Heads, Android: N...
FFG Worlds 2012 - Netr...
Strategy Segment: Andr...
Android: Netrunner Ga...
Android: Netrunner Rev...
Android Netrunner - 201...
(53) Items - Page (1/4)

Plugins can be used to add audio and video sources, including YouTube and even iPlayer and Spotify from externally installed scripts.

Graham Morrison wasted his youth configuring *MythTV*. He thinks *Kodi* is the best thing since Jean Michel Jarre.

LV PRO TIP
If you enable SSH on OpenELEC, you can connect remotely to your media centre using the default username of 'root' with a password of 'openelec'.

LINUX VOICE TUTORIAL

MASTER YOUR EMAIL WITH KMAIL

GRAHAM MORRISON

Graham Morrison tames possibly the most powerful email client in the world and yet still doesn't reply to all his email.

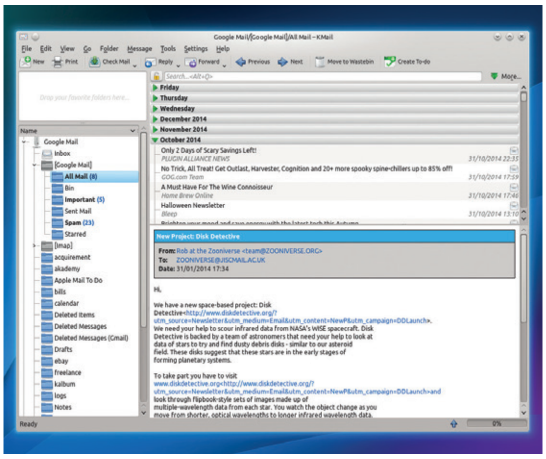
WHY DO THIS?

- Learn some of the most powerful sorting and aggregation functions of any email client that can help free your time.
- Impress your friends.

Not so long ago, it seemed the web browser had replaced our email clients. Services such as Google's Gmail became so convenient, powerful and easy to use, not to mention ubiquitously access from our phones, laptops and desktops, that running a standalone application seemed disconnected and old fashioned. But attitudes towards email have been changing, especially since Edward Snowden revealed that many online mail services did little to protect our privacy.

This has given fully-fledged desktop applications a boost as we mix our accounts and services, or start to use encryption. And there's a great selection of email clients for Linux. We love Gnome's *Geary*, for example, and it's done a wonderful job at re-inventing the simple and clean user interface for the web service generation. And Mozilla's *Thunderbird* is still the go-to application for many, thanks to its rock-solid development, the sympathetic aims of the Mozilla Foundation and configurability.

But there's another option that we don't think is often considered, and that's KDE's *KMail*. *KMail* is a great choice because it can be made to look fantastic, it's well integrated with any Linux desktop and utterly configurable. *GnuPG* encryption is almost transparent and there are powerful filters, anti-spamming and ad-blocking mechanisms. Its problem is that none of this is apparent when you first install and launch the application. *KMail* can look old-fashioned and difficult to tame, and features such as custom tagging and virtual folders are barely documented. These are some of the problems we're going to tackle over the next four pages, turning *KMail* into what we think is a contender for the best email client on Linux, and a great migration alternative to proprietary applications like OS X's *Mail* or whatever Microsoft's equivalent is these days.



Only 2 Days of Scary Savings Left!
PLUGIN ALLIANCE NEWS 31/10/2014 22:35
No Trick, All Treat! Get Outlast, Harvester, Cognition and 20+ more spooky spine-chillers up to 85% off!
GOG.com Team 31/10/2014 17:59
A Must Have For The Wine Connoisseur
Home Brew Online 31/10/2014 17:46
Halloween Newsletter
Bleep 31/10/2014 13:10

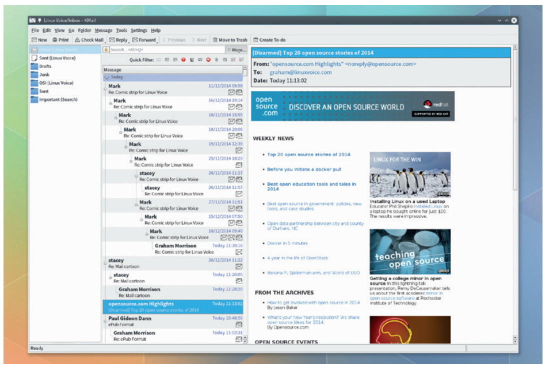
New Project: Disk Detective
From: Rob at the Zooniverse <team@ZOONIVERSE.ORG>
To: ZOONIVERSE@JISCMAIL.AC.UK
Date: 31/01/2014 17:34

Hi,

We have a new space-based project: Disk Detective<http://www.diskdetective.org/?utm_source=Newsletter&utm_medium=Email&utm_content=NewP&utm_campaign=DDLaunch>.
We need your help to scour infrared data from NASA's WISE spacecraft. Disk Detective is backed by a team of astronomers that need your help to look at data of stars to try and find dusty debris disks - similar to our asteroid field. These disks suggest that these stars are in the early stages of forming planetary systems.

To take part you have to visit www.diskdetective.org<http://www.diskdetective.org/?utm_source=Newsletter&utm_medium=Email&utm_content=NewP&utm_campaign=DDLaunch>and look through flipbook-style sets of images made up of multiple-wavelength data from each star. You watch the object change as you move from shorter, optical wavelengths to longer infrared wavelength data.

Without changing anything, *KMail* splits the main window into three – one panel for your folders, one for the message list and one that previews the email itself.



KMail is often overlooked because its default appearance filtering options don't give the best first impression.

Getting email

We've installed the latest version of *KMail* (version 4.14) into a KDE environment on both Arch and Kubuntu 14.10 for testing. In the years since its migration to the *Kontact* 'PIM' suite, the development pace has been slow and steady, which means there are still updates, but any version from the last year is basically functionally identical. We're not huge fans of the 'PIM' suite concept, where email is integrated with contacts, notes, a journal and todo lists (we think PIM stands for Personal Information Management, was popular when email clients wanted to look like Microsoft's *Outlook*).

For that reason, we run *KMail* independently of the *Kontact* suite, despite *Kontact* being its default incarnation. In Kubuntu, for example, *Kontact* is launched from the Favourites list, whereas *KMail* can be found under the Internet category. Whichever you choose, the *KMail* component behaves identically.

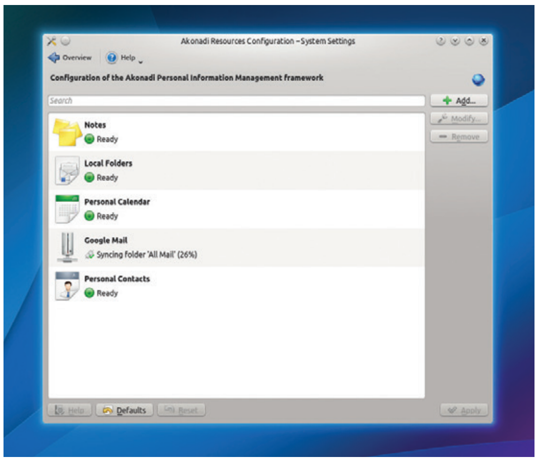
When *KMail* is launched for the first time, you need to navigate past both the Tip of the Day, which can be bypassed, and the Account Assistant. This assistant is the best way of configuring *KMail* for your specific email account, and it will first attempt to guess settings the correct parameters using Mozilla's connections database. If you're using Gmail, for example, Mozilla's database will provide the correct incoming and outgoing server details, and you'll be able to choose between IMAP or POP3. If you've not configured email manually before, IMAP and POP3

What is Akonadi

It may have a silly name, but *Akonadi* is an essential part of how KDE applications work. As with lots of other parts of KDE, the motivation behind *Akonadi* is to avoid duplication and re-invention, and it deals specifically with the way your personal information is stored and accessed by other applications. It's the framework that currently handles email, addresses, events, journals, calendars, notes and alarm settings. *KMail* uses *Akonadi* to manage your email while at the same time making it available to those applications and processes that you permit with their support for *Akonadi*.

This could mean allowing the desktop to search and index your email, for instance, so that results are delivered from a global search prompt, but it could equally be about having a single contacts list of names and addresses, and sharing access to that list with all the applications that request it. This is how the same contact addresses work with both instant messaging and email, or how the content of your emails are delivered as search results through *KRunner*.

The most important thing to know about *Akonadi* is that while it is a database, it doesn't actually store your email and you don't have worry about adding anything too specific to your backup routine. If your emails are stored on an IMAP email server, they'll still be stored there. If you're using POP3, your emails will still be downloaded to your local **maildir** folder, just as they would with any other email application. *Akonadi* is an interface to these repositories and only stores metadata relating to those sources, not the actual data itself. It's like Google for your personal information, with applications accessing this information through calls to the framework.


Akonadi Resources Configuration – System Settings

Configuration of the Akonadi Personal Information Management framework

Notes – Ready
Local Folders – Ready
Personal Calendar – Ready
Google Mail – Syncing folder 'All Mail' (26%)
Personal Contacts – Ready

In KDE 4, the resource configuration panel gives an overview of all the state of any *Akonadi* services you may be using.

are the two most popular protocols used to send email from the server to your client. Most people use IMAP, because it allows you to leave the email on the server and only browse through subjects or download email on demand, if you need to. *KMail* supports IMAP push, which will inform you immediately as soon as you receive an email, rather than relying on polling the server every set period. It also allows you to check your email through other portals, such as through a web browser through Gmail or *RoundCube*. This isn't normally possible with POP3 as emails are usually deleted after they're sent to your client (this is an option for IMAP too). You may prefer to download and delete emails, as it reduces your online footprint and pulls all your email onto your hard drive, under your direct control.

We're using online services like Gmail as a convenient example, but it's important to remember that you can also pay for or install your own email domains and servers, so that you're in control of every aspect of your email. This won't necessarily make it any more secure – email is still sent from one server to another as plain text, with no guarantee of validity unless there's a chain of encryption – but it removes your data from the meta-crawling context gathering of most online services. Even if you're using an email client with a web service, this is still a step in the right direction. *KMail* is more extensible and more powerful than even the best web interface. With your own server, as long as you're using the typical ports for IMAP or POP3, you'll be able to enter your details just as you would with any other mail service.

After entering the details, you need to click on Create Account rather than Next, as this will just re-detect the server settings for you. Create Account will add your details to the *KMail* configuration, and you'll be asked to create a KDE Wallet if you don't have one already. This is a global password holder that stores important information, such as your email login details, behind a single password. After setup has completed, click on Finish. *KMail* will now go off and download your email and the time this takes is going to be entirely dependent on how much email is stored on your account and the speed of your connection. 2GB of email took about 20 minutes to synchronise with *KMail* for us, with a decent internet connection.

The main window

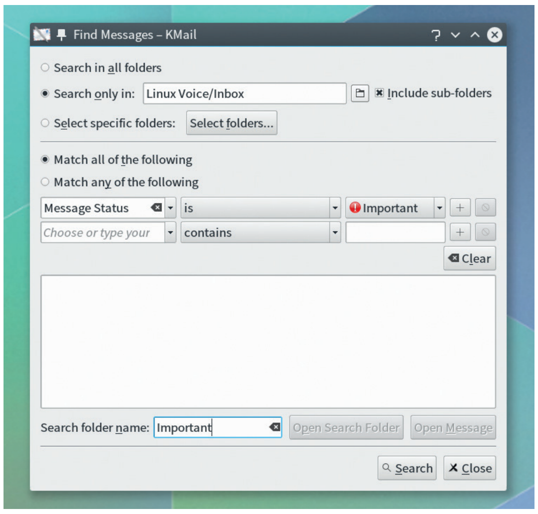
With a fresh installation, *KMail*'s default window will list your various email sources within a panel on the left-hand side and a simple 'Changes' page on the right listing whatever's new in each version. But after your accounts have synchronised, you'll find that the window on the right gets split into two. When you select an inbox or a folder from the panel on the left, the list of emails contained within that folder are listed in the top panel on the right, while the contents of any email you end up selecting will be shown in the bottom-right. You can change the size of any of these panels by click-dragging the divider. If you drag this divider to the border of the panel window, it will close one of the windows, and this can only be opened again by click and dragging the border down from where it's now locked in place. This is difficult to spot if you don't know there's a 'minimised' panel there.

Folders in *KMail* are powerful. Right-click on one and select Folder Properties to see what we mean. You can have a different identity for each folder, for instance. Identities can be added and edited from the main Configuration panel (Settings > Configure KMail) and using a different identity for a folder might be useful if it contains email forwarded on from a different account, or a different Linux alias. They're also useful for mailing lists, where you may want to use a different identity or email address, and there are further options for mailing lists hidden within its own tab. You can customise how your reply will look and how the folder appears and is sorted, completely

LV PRO TIP
You can only run one instance of *KMail* but you can tab different views and open more than one window. Most configuration panels can also be opened at the same time.

Virtual folders

Yet another of *KMail*'s great but undocumented features is its ability to create virtual folders. These are equivalent to Gmail's labels, and they enable you to sort email into dynamic groups without moving them from your inbox. You could use them to hold messages you've flagged as important, for example, or messages where you've added a descriptive tag, or even a folder to contain your previous searches. The secret to creating virtual folders is to use the search dialog, because it's from there that you can save your search as a new folder. Open this dialog window by pressing S or clicking on Find Messages in the Edit menu. The window allows you to specify an inbox or folder and then to create a list of conditions that will need to be met for the search to return results. When you click on Search, the list view will be populated with results. You will now be able to enter a search folder name, which you can save by clicking on 'Open Search Folder'. This creates a virtual folder that will be listed alongside your other folders, and we'd recommend making this a favourite in the same way you would your other folders. It will automatically update with any new results for the search you entered, and can be removed with a right-click and 'Delete Search'.



Search in all folders
[✓ Search only in: Linux Voice/Inbox] [✓ Include sub-folders]
Select specific folders: Select folders...

[✓ Match all of the following]
Match any of the following

Message Status | is | Important
Choose or type your | contains |

Search folder name: Important

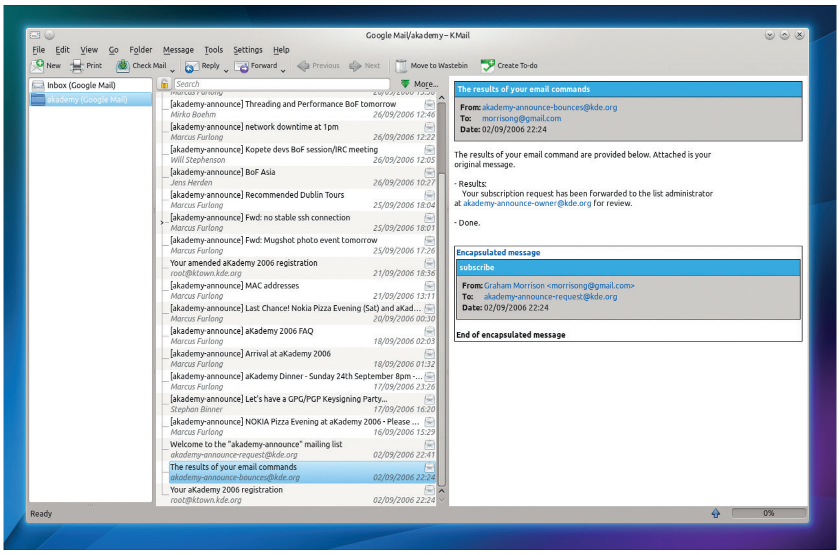
Virtual folders automatically update when an incoming email matches a certain search criterion. But you can also use them to list flagged messages, such as those you've marked as important.

LV PRO TIP
KMail is a little over-paranoid about HTML email, but it can render them without trouble if you enable the option in the security tab. We'd highly recommend sticking with plain text for your own emails, however.

independently of any global settings or how messages are listed in its parent folder.

One of the first improvements to the layout we'd recommend is dumping the default list of every email folder you've likely already created and switch to viewing only those you mark as favourites. This will help if you don't want to see folders such as Drafts, or Spam, or Sent. There's a separate panel (of course!) for favourites, and it's above the mail folder view. You can drag it down if it's hidden, and when empty, greyed-out text states "Drop your favourite folders here..." You can make any folder a favourite by right-clicking on it and selecting "Add Folder To Favourites". This may not make sense until you drag and hide the default list, leaving you only with those favourite folders you want to see. The Favourites view uses an icon view, but we find it more useful as as a list. Right-click within the panel and select the Mode sub-menu. This gives you the option of switching to the list view.

We're also not too keen on the way the message list appears above the message preview window. This is because we'd rather see a long list to provide context in the case of longer conversations, and the 6 or 7 messages that fit into the default view aren't enough. The ideal solution for us is to move the messages list view into its own panel, and the ability to do this is the last option hidden under the Layout tab in the main configuration panel. Select Show The Message Preview Pane Next To The Message List' and the main user-interface switches from two columns to three, with the folder list/favourites on the left, the message list in the middle and the preview pane on the right. You can now drag the borders between these to adjust their horizontal size. In our opinion, this makes *KMail* far more usable.



[akademy-announce] Threading and Performance BoF tomorrow — Mirko Boehm — 26/09/2006 12:46
[akademy-announce] network downtime at 1pm — Marcus Furlong — 26/09/2006 12:22
[akademy-announce] Kopete devs BoF session/IRC meeting — Will Stephenson — 26/09/2006 12:05
[akademy-announce] BoF Asia — Jens Herden — 26/09/2006 10:27
[akademy-announce] Recommended Dublin Tours — Marcus Furlong — 25/09/2006 18:04
[akademy-announce] Fwd: no stable ssh connection — Marcus Furlong — 25/09/2006 18:01
[akademy-announce] Fwd: Mugshot photo event tomorrow — Marcus Furlong — 25/09/2006 17:26
Your amended aKademy 2006 registration — root@ktown.kde.org — 21/09/2006 18:36
[akademy-announce] MAC addresses — Marcus Furlong — 21/09/2006 13:11
[akademy-announce] Last Chance! Nokia Pizza Evening (Sat) and aKad... — Marcus Furlong — 20/09/2006 00:30
[akademy-announce] aKademy 2006 FAQ — Marcus Furlong — 18/09/2006 02:03
[akademy-announce] Arrival at aKademy 2006 — Marcus Furlong — 18/09/2006 01:32
[akademy-announce] aKademy Dinner - Sunday 24th September 8pm -... — Marcus Furlong — 17/09/2006 23:26
[akademy-announce] Let's have a GPG/PGP Keysigning Party... — Stephan Binner — 17/09/2006 16:20
[akademy-announce] NOKIA Pizza Evening at aKademy 2006 - Please ... — Marcus Furlong — 16/09/2006 15:29
Welcome to the "akademy-announce" mailing list — akademy-announce-request@kde.org — 02/09/2006 22:41
The results of your email commands — akademy-announce-bounces@kde.org — 02/09/2006 22:24
Your aKademy 2006 registration — root@ktown.kde.org — 02/09/2006 22:24

The results of your email commands
From: akademy-announce-bounces@kde.org
To: morrisong@gmail.com
Date: 02/09/2006 22:24

The results of your email command are provided below. Attached is your original message.

- Results:
  Your subscription request has been forwarded to the list administrator at akademy-announce-owner@kde.org for review.

- Done.

Encapsulated message
subscribe
From: Graham Morrison <morrisong@gmail.com>
To: akademy-announce-request@kde.org
Date: 02/09/2006 22:24

End of encapsulated message

It makes more sense to have a three-column view for emails, especially with the wider aspect ratio of most modern displays and laptops.

Changing the message list

Two of *KMail*'s most powerful features are the themeing and sorting engines which manage how the message list is displayed, and both of which can be different for different folders or views. This stuff is complex and little-understood, so we're going to try and explain them both here and how you can create your perfect setup. If you get a lot of email, they both allow you to fine-tune the appearance of your inbox to give maximum context to your messages without you having to resort to clicking on them.

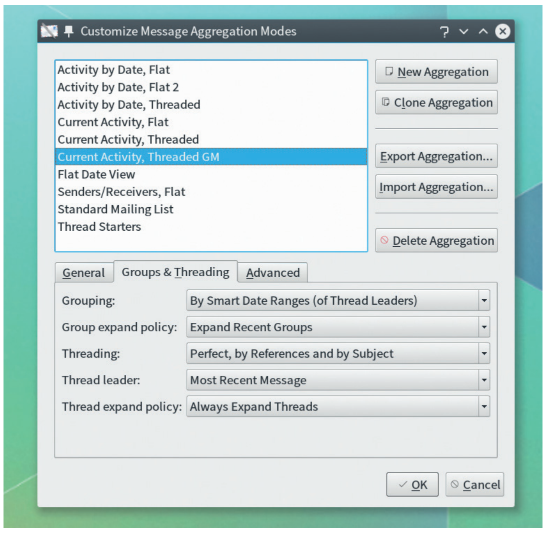
How messages are listed is the result of three specific configuration options. The first is the aggregation mode. This governs how messages are threaded, or appear tabbed in from parent to child so that messages are grouped by conversation. The second is how those parent messages and their children are sorted, as once you've included older messages in a thread, you need more control over which messages appear where. Finally, there's the theme options, which change how each header is displayed, along with the icons and actions that can be performed on a message. All three of these options are accessible from the View > Message List menu.

Before you can understand sorting, we need to understand what *KMail* calls message aggregation.

Aggregation in this case means the way that related messages are grouped together, mostly as a threaded conversation, but it also groups messages together according to the day. The 'Activity by Date, Flat' aggregation option, for instance, simply lists all the emails you've received today, with a marker dividing previous days. 'Flat' means there's no threaded view of the conversation. 'Activity by Date, Threaded' is the same only with the emails branched off the first email in a thread when you click on the small cross to open the thread contents. A thread will include emails earlier than the date if *KMail* thinks they're part of the same activity, with the latest email added to the bottom of the list. The next two options are very similar. 'Current Activity' is used to try to make an intelligent guess about the date of an email. Next there are aggregations that are easier to understand, including 'Thread Starters', a list of contacts and your conversations, a flat date view and a shallow list.

Most importantly, you can create your own aggregation theme by selecting the 'Configure' option at the bottom of the menu, which is exactly what we've done. We like to see threaded, open conversations arranged by the date, so we duplicated this configuration, opened the 'Configure' window and used the 'Grouping and Threading' tab to fine-tune the automatic expansion of threads, the smart date ranges and which messages are used at the head of a thread. You need to be slightly careful, because some of the aggregations will place new messages in past days as they attempt to keep new email together with an old thread. We also use 'Perfect And By References' as the threading policy, so that threads don't become huge as *KMail* tries to keep every email you've ever sent. You can then save your own aggregation and use this on specific folders or globally.

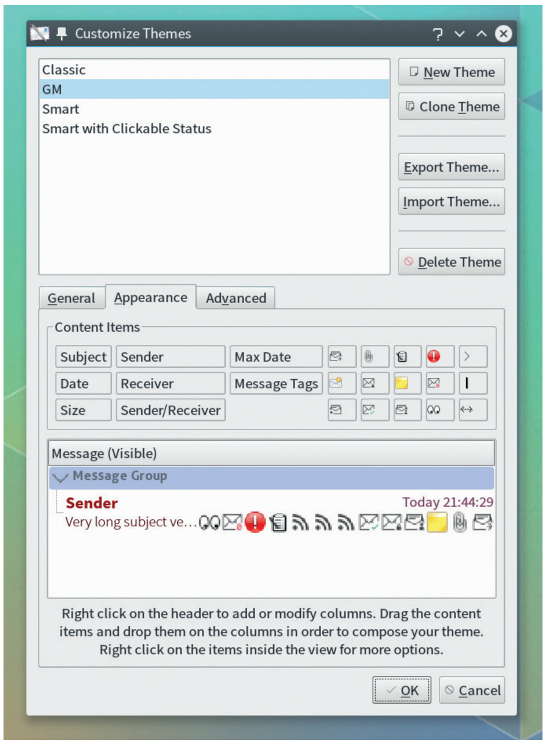
And that leads us onto something closely linked to aggregation – sorting. Sorting options for the message list can be adjusted by selecting 'Sorting' from the View > Message List menu. Message sorting comes after aggregation, so you'll still get the same view, but it allows you to switch between how activities and threads are displayed. Nearly all of this, such as 'By Date/Time', 'By Sender' or 'By Receiver' are self-explanatory, with perhaps the exception of 'By Date/Time of Most Recent in Subtree'. This is a reference to which message in a thread is used to group them together.


Customize Message Aggregation Modes

Activity by Date, Flat
Activity by Date, Flat 2
Activity by Date, Threaded
Current Activity, Flat
Current Activity, Threaded
Current Activity, Threaded GM
Flat Date View
Senders/Receivers, Flat
Standard Mailing List
Thread Starters

Grouping: By Smart Date Ranges (of Thread Leaders)
Group expand policy: Expand Recent Groups
Threading: Perfect, by References and by Subject
Thread leader: Most Recent Message
Thread expand policy: Always Expand Threads

How messages are grouped and threaded can also be customised to fit the way you work.


Customize Themes

Classic
GM
Smart
Smart with Clickable Status

Right click on the header to add or modify columns. Drag the content items and drop them on the columns in order to compose your theme. Right click on the items inside the view for more options.

Almost anything about the way the message list is presented can be changed with the theme editor.

Finally, we get to change how the message list is rendered. This uses a small selection of themes that will change the font, position and information for exactly what information is included in each list item, as well as where each component is placed. You can customise the theme and download new ones, and we think it's worth experimenting with because there are more options in the simple window that appears than you'll find in the whole of a supposedly more complicated application such as *Gimp*. You can change everything, from the icons' size and text colour, to how much the edges are rounded, and the way in which the background and foreground colours are rendered. Our layout has a 'bold' sender field in the most prominent position with the date and time to the right. Beneath this is the subject followed by the icon symbols. We feel this is the perfect balance. The only option we couldn't find was a way to reduce the size of the indentation in threaded items.

All this leaves us with a vastly improved email client and one that looks like it belongs in 2015 rather than 2005. That *KMail* offers all these options is brilliant, even if that leaves us floundering for their meaning, and we like the contrast you find in *KMail* with an application like *Geary*, which we also like a lot but we just wish it had more options. LV

Graham Morrison is a lapsed KDE developer and the editor of the magazine you're reading. He gets a lot of mail.

LINUX VOICE TUTORIAL

REGULAR EXPRESSIONS: WORK LESS, WORK SMARTER

MARCO FIORETTI

Save time and baffle your friends with one of the most powerful tools available to the Linux users – the magic regex.

WHY DO THIS?
- Process datasets of any kind without writing complex, low-level software.
- Always find the file(s) you were looking, looking at their name, content or both.
- Catalog and tag all sorts of documents, from photographs to office reports.

A regular expression, or regex for short, is a sort of textual formula which describes the structure ("pattern") of a generic snippet of text. Regexes are given as input to software programs called regex engines, to find and process all the strings inside one or more text files or streams that contain those patterns. Regex engines are embedded, or embeddable as libraries, in almost any relevant software language used today.

Regular expressions always seem, to beginners, like gibberish typed by somebody drunk in total darkness. Therefore, the first concept we have to make clear is why you should bother with them at all. Our answer is that, even if you are not a programmer, regexes can make your life a lot easier if you routinely produce or process anything made of plain text. Here are a few examples to show you that you likely are already doing something like that, or you should. All are real, non-programming uses we make of regular expressions on our desktops.

Regex applications for the rest of us

Exiftool and the *ImageMagick* suite can georeference, frame, resize, rename and catalog all your pictures in a few seconds, as long as you give them proper, suitably flexible instructions. And the most efficient way to do it is to generate them (also) with regular expressions.

The same applies to home and small business accounting, or any other data analysis activity. A tool like *Gnuplot* can plot countless charts for you, but extracting only the data you need from a 50,000-row spreadsheet would be a daunting task without regular expressions.

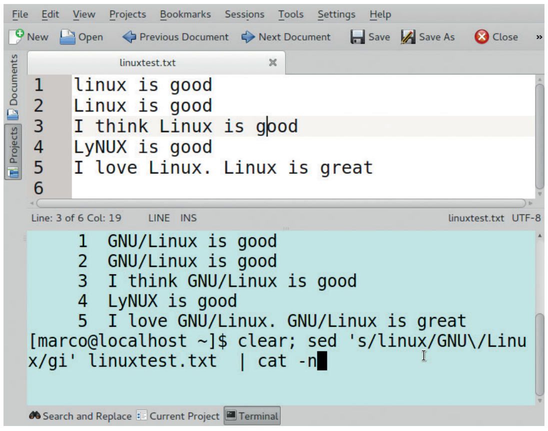
Ditto for database dumps and websites, which are both big blobs of plain text. Quick scripts based on regexes can reformat or update such collections of documents and raw data much more quickly than you could without them. Finally, office productivity: the OpenDocument format for texts, spreadsheets and presentations is, at its core, nothing more than zipped plain text, that is stuff that you can search, analyse and update with regular expressions.

The goal of this tutorial is to teach you just enough about regular expressions to create simple regexes and adapt many of the more complex ones you may find online to your needs. We hope to give you lots of great ideas and help you find the right tools to implement them. We will show you what can you describe (and find) with regular expressions; where, how many times and in which exact ways a regex engine may search for what you described; finally we will discuss what regexes can do for you.

As they are supported by almost all programming languages, regular expressions exist in a variety of dialects. What you read here should work in any of them, but don't be surprised if it requires adjustments. Another thing to keep in mind is the meaning of "character". For English and all the other alphabets contained in the ASCII set, each "character" is encoded in one byte, that is eight bits. The sequence 01011000, for example, corresponds to capital 'X'. Non-alphabetic symbols, and in general everything outside the ASCII set, may be encoded using more than one byte per character. Matching patterns of multibyte characters may require changes in your expressions, so be prepared for that too.

First, matching

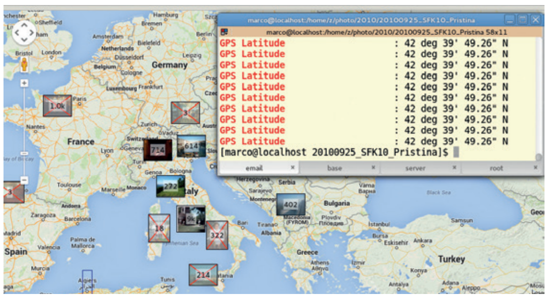
Regular expressions, or more exactly their engines, work in two main phases. First they find the strings of



```
linux is good
Linux is good
I think Linux is good
LyNUX is good
I love Linux. Linux is great
```

```
     1  GNU/Linux is good
     2  GNU/Linux is good
     3  I think GNU/Linux is good
     4  LyNUX is good
     5  I love GNU/Linux. GNU/Linux is great
[marco@localhost ~]$ clear; sed 's/linux/GNU\/Linu
x/gi' linuxtest.txt  | cat -n
```

A great way to practice regular expressions is to save several forms of the same string in one text file, then apply a regex to it, to see which lines match.


marco@localhost:/home/z/photo/2010/20100925_SFK10_Pristina
marco@localhost:/home/z/photo/2010/20100925_SFK10_Pristina 58x11
email
base
server
root
Germany
France
Italy
Spain
Bulgaria
Serbia
Albania
Greece
Turkey
Black Sea

```
GPS Latitude                    : 42 deg 39' 49.26" N
GPS Latitude                    : 42 deg 39' 49.26" N
GPS Latitude                    : 42 deg 39' 49.26" N
GPS Latitude                    : 42 deg 39' 49.26" N
GPS Latitude                    : 42 deg 39' 49.26" N
GPS Latitude                    : 42 deg 39' 49.26" N
GPS Latitude                    : 42 deg 39' 49.26" N
GPS Latitude                    : 42 deg 39' 49.26" N
GPS Latitude                    : 42 deg 39' 49.26" N
GPS Latitude                    : 42 deg 39' 49.26" N
[marco@localhost 20100925_SFK10_Pristina]$
```

The background of this snapshot is the map view of *Digikam*, showing photos grouped according to where they were taken. To georeference the pictures, we created a list of place names, each with its coordinates, and used find which ones should be written in each file.

text that matches the pattern described by the regex itself. The second phase, covered later in this tutorial, consists of actually doing something with that text. Patterns can be literal or contain special symbols that enrich their meaning. Besides, they are almost always delimited by slashes, are case sensitive and work on one line of input text at a time. This pattern:

```
/linux/
```

means "look for strings containing EXACTLY the sequence of lowercase letters l, i, n, u and x, in this order, one after another, in any position inside the whole string". Therefore, only the first two samples of this list would match a regex like that:

- I love linux.
- linux is great.
- How great is Linux?
- I just misspelled lynux.
- I did it again: li nux.

If there is more than one match for the same regex in a string, the engine will, by default, return the leftmost, longest one. More on this later.

Anchors

Matches like those in the previous paragraph happen regardless of where in the input string the pattern is found. You need extra meta (special) characters called anchors to say that you only care about matches that begin or end in certain exact positions. The two most common anchors are **^** and **$**: **(Fig. 1)**

By default, these two metacharacters work on lines. In strings that contain more than one line, **^** and **$** may match on the beginning or end of each of them. To only match at the actual beginning or end of whole multiline string, instead, you should use the anchors **\A** and, respectively, **\Z**.

Another anchor you must know is **\b**, which means "word boundary": **\bk** and **k\b** will match on strings with at least one word beginning and, respectively, ending, with the letter "k" (in regular expressions, a "word" is any continuous sequence of letters, digits, underscores, dashes and nothing else).

Character classes

You can make your regular expressions more compact and readable by using predefined character classes, or by building your owns. The most common standard classes are:

- **\d** Matches one digit (0...9).
- **\s** Matches one "white space", that is, characters like actual white spaces, tabs, line breaks etc.
- **\w** Matches one "word character", that is letters, digits, underscore and dash.
- **. (dot)** Matches any character except newline (escape it with a slash **\.** if you mean "one dot"!).

Upper case versions of the three identifiers above negate the corresponding classes: **\D** means "everything but one digit", **\S** "everything but one whitespace" and so on.

If these classes aren't enough, you can build your own inside square brackets, mixing both literal and meta-characters as you need, and remembering that an initial caret negates the whole class:

- **[aqF\d]** means "match one a OR one q OR one capital F or any digit".
- **[^rtw]** means "match any character but r, t and w".

Ranges of consecutive characters can be shortened with a dash. To match one occurrence of any of the first 10 letters of the English alphabet, lowercase, write **/[a-j]/**.

The main gotcha here is that a negated class does not necessarily mean what it may seem at first sight. This regular expression, for example:

```
/boar[^d]/
```

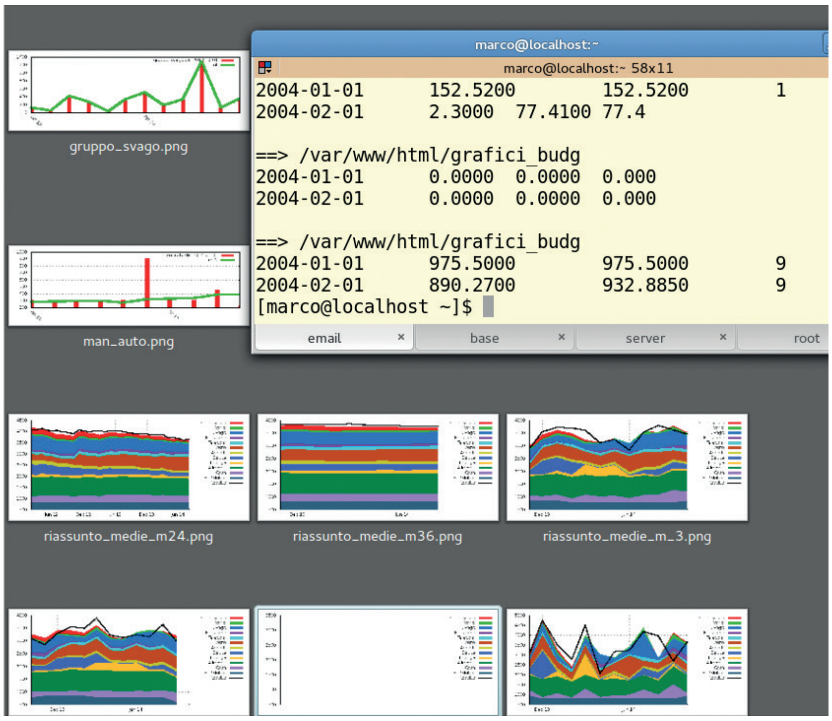
will not match the word "boar" if it is at the very end of a string! That regex in fact, means "search the sequence b, o, a, r, followed by at least one character which is not d"!

Modifiers

One important class of regex special characters is that of modifiers. These are characters that go outside the actual regex, because they modify the very way the whole matching happens. There are many modifiers, depending on which language you use, and they don't work in the same way in all languages, so we will list only a few ones to present the concept.

Adding an **i** after a regex makes its matching case-insensitive. A **g** forces it to go "global", that is to search for all the matches in a string, not just the first one. An **e**, instead, tells the regex engine to evaluate the right half of the expression (which we will explain in the second part of the tutorial) as if it were an actual command programming statement.

Finally, **m** and **s** turn on multiline and single-line mode: the first treats strings as multiple lines, making the **^** and **$** anchors match on the start or end of any



```
2004-01-01      152.5200        152.5200        1
2004-02-01      2.3000  77.4100 77.4

==> /var/www/html/grafici_budg
2004-01-01      0.0000  0.0000  0.000
2004-02-01      0.0000  0.0000  0.000

==> /var/www/html/grafici_budg
2004-01-01      975.5000        975.5000        9
2004-02-01      890.2700        932.8850        9
[marco@localhost ~]$
```

Here we have hundreds of different charts, all created automatically with *Gnuplot*, after extracting the corresponding datasets with regular expressions from the same, huge file of raw data in CSV format. Please note that the same could not be done on databases, as the filtering power of regular expressions greatly exceeds those of database engines.

LV PRO TIP

Try to seriously look at your computer activities as something that manipulates big quantities of text, or can be described by them. You will be surprised by how many of the things you do fit this description, and can therefore by managed with regular expressions.

of them. The **s** modifier does the opposite. You can use more modifiers in the same regex: to find all the occurrences of the 'linux' substring in a string, regardless of case, write **/linux/gi**.

Greediness, laziness and backtracking

By default, regex engines are greedy. This simply means that they will always try to match the longest possible string that respects the pattern described by a regex. As a dumb, but effective example, let's imagine we want to find all the URLs in some HTML page, and consider this regex:

/href=.*>/

Would its bold, higlighted part match the whole URL in this string, quotes included, and nothing else:

<a href="http://www.linuxvoice.com">Linux Voice</a> is <b>great</b>

or not? The answer is "no", and the reason is exactly that regex engines, by default, are greedy. A regex like that will match everything up to, and included, the "b" character at the end. Because it says to match as many characters as possible, whatever they are (that's what the dot quantified with the asterisk means), as long as they are followed by one **>**. That is why it won't stop at the end of the URL. To match only "http://www.linuxvoice.com", we would have to write:

/href=[^>]*>/

This is a proper description of quoted URLs and nothing else: as many characters as possible (the asterisk) between the "href=" substring and a **>** character, as long as they themselves are not **>** characters!

There are times when negated character classes cannot do the trick. In such cases, you can still force the quantifiers to be "lazy", that is to stop at the shortest, not longest, possible match. Just put a question mark after them:

/href=.*?>/

This will make the engine stop at the first **>**, that is, unlike the first form you saw, only match the quoted URL. In practice you should always try to apply the first strategy – greedy matching of negated classes.

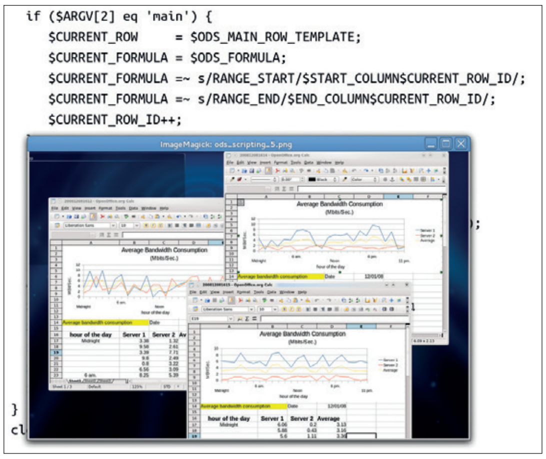
The reason is that regexes with that structure allow their engines to look at the string(s) only once, always in the same direction from left to right. This:

LV PRO TIP

Want to see by yourself which variant of the same regular expression is faster? Here is a quick and dirty way: write one long sentence in an empty file that may match it, then copy it many times (ie ten thousands or more), then run the script containing the regex on it with the **time** command.

LV PRO TIP

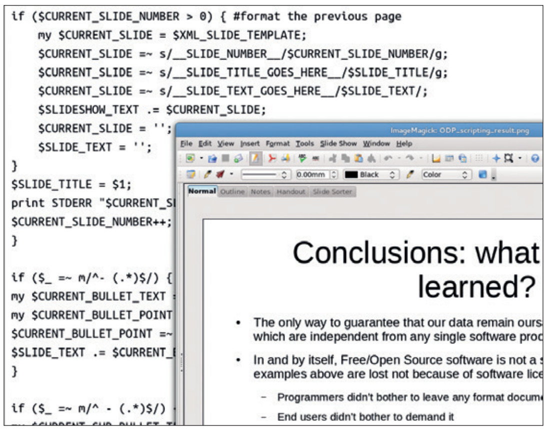
Regular expressions are also great for managing large quantities of files, without changing their content at all! The right "regex-enabled" tools for this kind of job are *egrep*, *sed* and *awk*. Combine them with the *find* command, and you will quickly sort and catalog thousands of files in any way you may want.



```
if ($ARGV[2] eq 'main') {
   $CURRENT_ROW     = $ODS_MAIN_ROW_TEMPLATE;
   $CURRENT_FORMULA = $ODS_FORMULA;
   $CURRENT_FORMULA =~ s/RANGE_START/$START_COLUMN$CURRENT_ROW_ID/;
   $CURRENT_FORMULA =~ s/RANGE_END/$END_COLUMN$CURRENT_ROW_ID/;
   $CURRENT_ROW_ID++;
```

ImageMagick: ods_scripting_5.png
Average Bandwidth Consumption
(Mbits/Sec.)
Average bandwidth consumption
hour of the day
Server 1
Server 2
Average

The same technique shown in Figure 4 can be applied to all types of office documents supported by the ODF standard. Create a template spreadsheet, fill it with bogus numbers or placeholder strings, and the right formulas and charts.



```
if ($CURRENT_SLIDE_NUMBER > 0) { #format the previous page
    my $CURRENT_SLIDE = $XML_SLIDE_TEMPLATE;
    $CURRENT_SLIDE =~ s/__SLIDE_NUMBER__/$CURRENT_SLIDE_NUMBER/g;
    $CURRENT_SLIDE =~ s/__SLIDE_TITLE_GOES_HERE__/$SLIDE_TITLE/g;
    $CURRENT_SLIDE =~ s/__SLIDE_TEXT_GOES_HERE__/$SLIDE_TEXT/;
    $SLIDESHOW_TEXT .= $CURRENT_SLIDE;
    $CURRENT_SLIDE = '';
    $SLIDE_TEXT = '';
}
$SLIDE_TITLE = $1;
print STDERR "$CURRENT_SI
$CURRENT_SLIDE_NUMBER++;
}

if ($_ =~ m/^- (.*)$/) {
my $CURRENT_BULLET_TEXT
my $CURRENT_BULLET_POINT
$CURRENT_BULLET_POINT =~
$SLIDE_TEXT .= $CURRENT_
}

if ($_ =~ m/^ - (.*)$/)
```

ImageMagick: ODP_scripting_result.png
File Edit View Insert Format Tools Slide Show Window Help
Normal Outline Notes Handout Slide Sorter

Conclusions: what learned?

- The only way to guarantee that our data remain ours which are independent from any single software proc
- In and by itself, Free/Open Source software is not a examples above are lost not because of software lice
  - Programmers didn't bother to leave any format docum
  - End users didn't bother to demand it

What you see in the foreground is a standard ODF slideshow. In the background there are some of the regular expressions that created it, filling a blank template with content loaded by a plain text file.

/href=[^>]*>/

means "you can stop as soon as you find a ">". Writing this, instead:

/href=.*?>/

lets the engine free to match as much as it can till the end of the string, that is free to be greedy... as long as, once it's there, it then starts to parse the string backwards, to check if maybe there is also a shortest match, and choose that one. This behaviour, called backtracking, is very powerful and is often the only way to write some complex regular expressions. However, it is also easy to see, if you think a bit about it, that it may take much longer to find the right match.

Backreferences

The last bit of theory and regular expression syntax we need to introduce is called backreferencing. This is the mechanism that you must use to tell a regex engine to remember what it found, in order to reuse it. Backreferencing can be used several times in the same regex, and is done by means of parentheses:

/I use Linux, my name is (\w+) and my favourite distro is (.+)$/

Those two pairs of parentheses tell the engine to store the name of the user in the first of the special internal variables it uses for backreferencing, and the name of the distribution in the second one. Depending on the programming language, these variables are named **$1**, **$2** or **\1**, **\2**, or something else on the same line, but they always have the same meaning. Perl, for example, uses the dollar notation, and this is what it would store in **$1** and **$2** when applying the regex above to two sample strings (**Fig. 4**)

Relax: we have finally laid out on the table enough features of regular expressions to explain with simple, but realistic examples why one should bother to learn and use them. In a nutshell, a program using a regex engine can do three things once that engine has found a match for a regular expression: execute some predefined action, pass the match result to the next instruction for further processing, or immediately replace the match with some other string, which may be a constant or something calculated on the fly. The first behaviour is what command line utilities like **grep** and **egrep** do. If you launch **egrep** in this way:

```
#> egrep '^(Debian|Centos)' install-list.txt
```

it won't have any choice but to print in the terminal all and only the lines of the **install-list.txt** file that begin (did you notice the caret?) with either Debian or CentOS. This snippet of Perl code shows the second type of action:

```
$_ =~ m/www\.(.*)\.com$/;
print " .com domain name found: .$1\n" if ($1);
```

In Perl, **$_** is the variable that contains the current line of text input, and the **=~ m** operators mean, more or less, "check if this string matches this regex".

If a match is found it is stored by the parentheses in the special **$1** variable. The second line prints that variable, but only if it is not empty. Therefore, that code will print all and only the internet domain names in the text stream with a "www.SITENAME.com" structure.

Finally, there is substitution. Did your boss come and tell you to change the format of your company's customers list? No problem, if you know regular expressions. For simplicity, let's assume that you must only swap first and last names, so that

```
First Name: Winston Last Name: Churchill
```

in this way:

```
Last Name: Churchill First Name: Winston
```

Regular expressions that substitute text have an **s** before them, and the text pattern to use for replacement at the end. This is how you would generate a new list of names with the required format, using the **sed** program:

```
#>  cat namelist.txt | sed -E 's/First Name: (.*)Last Name: (.*)/
Last Name: \2\t\tFirst Name: \1/' > newnamelist.txt
```

Processing office documents

OpenDocument spreadsheets are zipped archives of several files. The actual content of the spreadsheet is in the one called **content.xml**. Each row of the spreadsheet is represented with a long string of text like this (here heavily trimmed for clarity!):

```
<table:table-row><table:table-cell office:value-
type="string"><text:p>SOME_NUMBER</text:p></table:table-
cell>.....</table:table-row>
```

See what we mean? Once you have created a spreadsheet template with predefined names in certain cells, it is trivial to make however many copies you want of that template, placing different values in those cells each time, with regular expressions like:

```
s/MY_LABEL_STRING/$CURRENTVALUE/
```

where $CURRENTVALUE may be a number taken from a database, calculated on the spot or provided by the user. Powerful, isn't it?

What to read next

The complete ODF scripts used by the author to generate the slideshows and spreadsheets shown in these pages, together with several others of the same kind, are available online at **http://freesoftware.zona-m.net/tag/odf-scripting**. The ones about *Gnuplot* are in the same blog, but in a different section: **http://freesoftware.zona-m.net/tag/gnuplot**. A very comprehensive online reference for regular expressions, which is still general enough to be usable in most languages, is the website **www.regular-expressions.info**. Finally, if you fall in love with this programming tool, go for the ultimate guide for it: the wonderful book by Jeffrey Friedl titled *Mastering Regular Expressions*, 3rd Edition, 2006, by O'Reilly Media. Besides teaching you more than you could ever imagine about the inner workings of regex engines, and how to make the most of them, it's great reading. In our opinion, few programming books show better than this how to be both more productive and creative at the same time on the job, using the right software tools.

Marco Fioretti is a Free Software and open data campaigner who has advocated FOSS all over the world.

Fig. 1

Character	Example	Matches?	
		Linux is great	**I love Linux**
^ (start of line)	/^Linux/	Y	N
$ (end of line)	/Linux$/	N	Y

Fig. 2: Other metacharacters

Character	Meaning
|	Alternation: '/(cat|dog)/ means "match cat OR dog" (don't forget the parentheses!)
\	Transforms the following special character in a literal one (e.g. \\\| if you want to match one actual "pipe" character)
\n	newline
\t	tab

Anchors and shorthands for character classes are just two of the special characters that you may use to describe text patterns with whatever structure. This table lists almost all the other metacharacters you must learn **(Fig. 2)**

Fig. 3 Quantifiers

Quantifier	Match:	Example	Matches strings like:
?	0 or 1 time	/liy?nux/	linux, liynux
*	0 or many times	/li.*nux/	linux, liynux, liiinux, liabc1234nux...
{m}	exactly m times	/liy{5}nux/	liyyyyynux
{m,}	at least m times	/liy{6}nux/	Left as exercise for the reader...
{m, M}	from m to M times	/liy{2,4}nux/	liyynux, liyyynux, liyyyynux

It is very common, when building a regular expression, to have to specify optional, or repeated characters, for example to find and correct typing errors. To do this, use these quantifying metacharacters and constructs.

Fig. 4

String	Content of $1	Content of $2
I use Linux, my name is Paul and my favourite distro is Debian	Paul	Debian
I use Linux, my name is John and my favourite distro is Red Hat	John	Red Hat

LINUXVOICE TUTORIAL

UNIX, LINUX AND HOW WE GOT WHERE WE ARE TODAY

JULIET KEMP

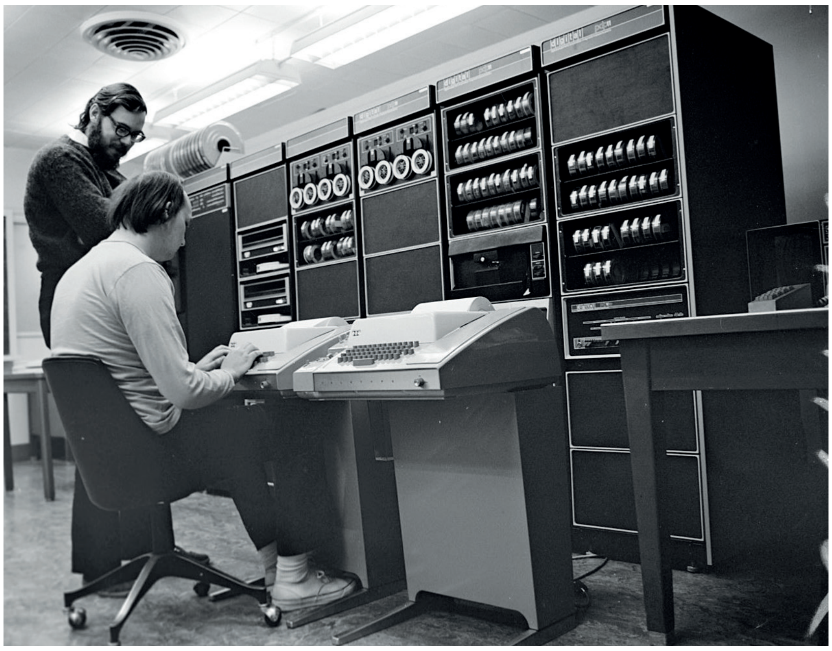
Linux didn't just come from thin air you know. Take a look back at how we got from the 1970s to the OS we know and love today.

Unix is the oldest operating system that's still widely used today. Linux -- which, if you're reading this, you're presumably interested in -- is just one of the many Unix clones and descendants that are kicking around the computer world, alongside UNIX (the trademark is all-caps) itself. Unix and Unix-like systems have always been popular as servers, with a rather smaller population of 'users' alongside them; but with the rise of the smartphone, nearly a billion people worldwide now have a Unix-type box in their pocket. Pretty good for a 45-year-old Bell Labs side project!

Bell Labs: starting out

Back in 1968, Bell Labs was involved with a shared project called Multics, an early time-sharing operating system for the GE-645 mainframe. However, the project, while functional for those working on it, wasn't producing the widely-available OS that Bell was after, and they pulled out. The last of the Bell people working on it (Ken Thompson, Dennis Ritchie, Doug McIlroy and Joe Ossanna) were keen not to lose their own access to interactive computing, so spent 1969 partly trying to persuade management to buy them a computer, and partly developing what would become the Unix filesystem. In his spare time, Thompson rewrote a game called *Space Travel* (a sort of solar system simulator where the player also piloted a ship), together with a bunch of supporting packages, to run on a spare PDP-7 that was kicking around.

Ken Thompson (sitting) and Dennis Ritchie at a PDP-11. Possibly even working on C at the time!
Copyright: CC-BY-SA Peter Hamer



Preparing programs for the PDP-7 was complicated, requiring them to be created on a GE 635 and the paper tapes carried by hand to the PDP-7. So Thompson began implementing a full operating system on the PDP-7: filesystem, processes, small utilities, and a simple shell. It was 1970 when Kernighan suggested calling the new system Unics or Unix – a play on Multics.

The filesystem developed on that first machine had i-nodes, directories, and device files, just like modern Unixes, but there were no path names (they were substituted by a complicated linking system). The system had processes, too, but they were very limited, with no forking or waiting. A fascinating 1979 paper by Dennis Ritchie (available online from Bell Labs – **http://cm.bell-labs.com/who/dmr/hist.html**) goes into detail about the various calls and processes.

In 1970–1 the system was rewritten for a new PDP-11, together with a text editor and formatter (*roff*). The machine began offering a text-processing service to the Patent department, and three typists from that department came to use it. This was an important part of demonstrating that Unix was genuinely useful, even if it got a little in the way of the programmers!

It was also in 1971 that Ken Thompson and Dennis Ritchie began working on the C programming language. Thompson had already developed a language called B, but although some general systems programs were written in it, the operating system basics were still in assembler (see page 106 for more on this minimalist way of coding). Once C was developed, it was used to rewrite the kernel into C in 1973 – the first time that an operating system had been written in anything other than assembler. This also let them demonstrate just how genuinely useful C was (and continues to be: it's still under the hood of plenty of programs).

In the mid-70s, UNIX began to be shipped out under licence, with its full source code included. It was versioned according to editions of the user manual, so Fifth Edition UNIX and UNIX Version 5 are the same. By 1978, when UNIX/32V was released for the new VAX system, over 600 machines were running some variety of UNIX, and various people (such as the BSD folk) were adapting it. At this point, an ongoing antitrust case still prevented AT&T from releasing a commercial product. When this was finally resolved in

1983, they released a commercial licence version of UNIX System V. Since the licence conditions weren't great for university use, BSD became more popular. (And, indeed, the BSD networking code made it back into the main Unix kernel.) Various other companies also developed their own versions of UNIX under licence; which turned in due course into the "Unix wars" where different companies tried to promote rival standards. POSIX was eventually the most successful, designed to be easy to implement on both BSD and System V.

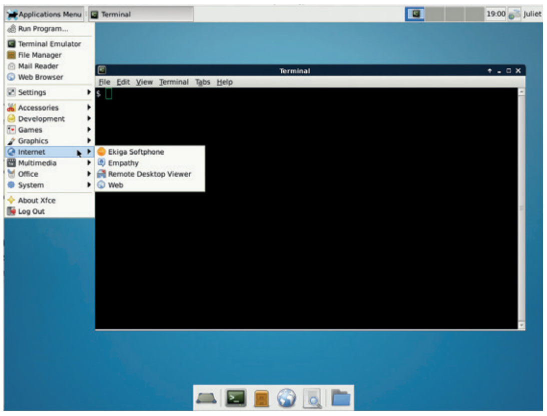
AT&T sold UNIX to Novell in the late 1980s, then (after UnixWare did badly) Novell transferred it to the X/Open Consortion, which now sets UNIX specification standards. Some parts of the licensing business were also sold to SCO, which in due course led to the SCO/Linux legal action (see boxout).

There are five commercial UNIX-certified OSes still available: OS X, HP-UX, Solaris, Inspur K-UX (used on mainframes), and AIX. Unfortunately you can't try out HP-UX or IBM's AIX without going through HP/IBM (or a partner) and spending a large sum of money; and for Inspur K-UX you need a mainframe. But there's more below on OS X and Solaris.

BSD Unix

The University of California, Berkeley, had a Unix Version 4 system running in 1974, and when Ken Thompson was there in 1975 he helped install Version 6. As more people, and other universities, became interested in the system, Bill Joy, a Berkeley grad student, started creating the Berkeley Software Distribution. This was an add-on to Version 6 Unix which included a Pascal compiler and the *ex* line editor (written by Joy). This was possible because Unix was still being released with full source code at the time. The second release, 2BSD, in 1979, included the text editor *Vi* and the C shell *csh*, both still available on Unix systems. (I wrote this article in *Vim*, an extended version of *Vi*, which dates back to 1991.)

BSD became increasingly popular as it improved – it was the OS of choice for VAX minicomputers (used for timesharing) at the start of the 80s. It included



```
$
```

I couldn't get Gnome running on FreeBSD but Xfce worked fine – the apps in the menu are probably from the abortive Gnome install though.

GNU tools

Here's just a few of the GNU tools you may be using:

- The Bash shell.
- The coreutils package, which provides ls, mv, rm, cat, and so on.
- The boot loader Grub.
- The sysutils package, which provides utilities to manage users and groups.
- tar and gzip.
- grep for searching through text files.
- make and autotools for building software.
- glibc, the C library, underlies a huge range of user software. You may never use it directly, but it's essential to your system.
- The GNU Compiler Collection compiles languages including C, C++, and Java.
- The graphics program Gimp.
- The Gnome desktop project (although this is effectively a separate entity now, it is officially a GNU project).
- The venerable text editor GNU Emacs… as well as scientific software, desktop software, internet software, and a whole plethora of development tools.

delivermail (a *sendmail* precursor) and the *curses* library, among other useful bits of software.

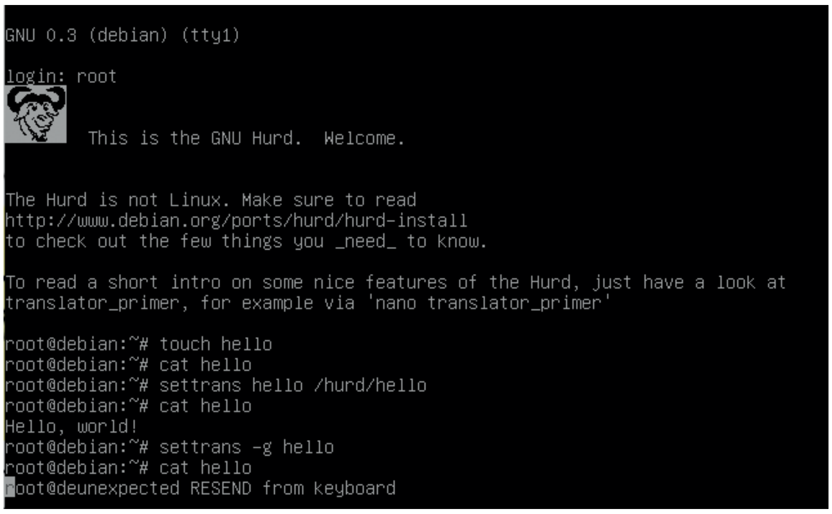
In 1989, BSD released its networking code separately, under the BSD licence. Prior to this, all BSD releases included AT&T Unix code, so post-1983 had required an (expensive) AT&T licence. The networking code had been developed entirely outside this, and various people were interested in acquiring it separately. The general BSD distribution was continuing to improve, and in 1990 the BSD team decided to rewrite all the AT&T-dependent code, resulting in Networking Release 2 in 1991, a freely available OS that was the basis for ports to Intel 80386 architecture and which would later become NetBSD and FreeBSD.

Networking is probably the BSD team's most important contribution to computing. Berkeley sockets, the first Internet Protocol libraries available for Unix, became the standard internet interface. The POSIX API is basically Berkeley with a few changes, so all modern OSes have some implementation of the Berkeley interface.

Unfortunately, the then-owners of the Unix copyright sued in 1992, and while the lawsuit was settled in 1994 largely in the favour of BSD (only three of the 18,000 files had to be removed and a handful more modified), development slowed massively during those two years. As it happens, this was while Linux was being developed. The slow release of 386BSD was part of what prompted Torvalds to create the Linux kernel.

The modern operating systems of FreeBSD, OpenBSD, and NetBSD are all descendants of the 386 port and of 4.4BSD-Lite. They in their turn have various descendants, including SunOS and Mac OS X. Most of these are open source and available under the BSD Licence. *Sendmail*, *Vi*, *curses*, and *csh* are a few of the BSD programs and utilities still in use today.

Of the currently available BSDs, FreeBSD is probably the most friendly to the average non-developer user (though they all have good points, and NetBSD has the distinction that you can install it on a toaster). You can download FreeBSD from **https://www.freebsd.org/where.html**, which also has good user documentation. FreeBSD's install is text-based and will be familiar if you've installed Debian in text mode.



```
GNU 0.3 (debian) (tty1)

login: root

    This is the GNU Hurd.  Welcome.

The Hurd is not Linux. Make sure to read
http://www.debian.org/ports/hurd/hurd-install
to check out the few things you _need_ to know.

To read a short intro on some nice features of the Hurd, just have a look at
translator_primer, for example via 'nano translator_primer'

root@debian:~# touch hello
root@debian:~# cat hello
root@debian:~# settrans hello /hurd/hello
root@debian:~# cat hello
Hello, world!
root@debian:~# settrans -g hello
root@debian:~# cat hello
root@deunexpected RESEND from keyboard
```

The Hurd running on *VirtualBox*. No graphical desktop! (Though X is supported.) The 'translator' trial from the README is shown.

LV PRO TIP

The Free Software Foundation argue that as GNU software makes up a significant part of a 'Linux' system (more than the kernel does in many systems), it should be referred to as GNU/Linux. This isn't reflected in mainstream usage.

The basic install is deliberately very sparse; afterwards you'll need to install any packages you want. The binary package management system is *pkg*, so to get the Gnome desktop I logged into the new system as root, then typed:

```
$ /usr/sbin/pkg   # this bootstraps pkg itself
$ pkg install xorg
$ pkg install xfce
$ echo "exec /usr/local/bin/startxfce4" > /home/juliet/.xinitrc
```

pkg search name is a useful command to find other available packages, and the documentation and other support for FreeBSD seems good. However, don't expect to log on immediately into a fully-featured desktop system; it expects you to decide the details of what you install for yourself.

BSD to SunOS to Solaris

In 1982, Bill Joy, one of the main BSD developers, joined three Stanford graduates to found Sun Microsystems. Their first generation of workstations and servers were based around a design created by Andy Bechtolsheim (co-founder of Sun Microsystems) while still studying at Stanford. The very first software was Sun UNIX 0.7, based on UniSoft Unix v7, but a year later, SunOS 1.0, based on 4.1BSD, was released.

SunOS continued to be based on BSD until the final update on SunOS4 in 1994. One of their major developments was the creation, in 1984, of the NFS (Network File System) protocol, allowing client computers to access files (largely) transparently over a network. NFS is an open standard so can be implemented by anyone. It's still used in modern networks, especially Unix and Unix-like ones.

In the late 1980s, AT&T and Sun began a joint project to merge BSD, System V, and Xenix, resulting in Unix System V Release 4 (SVR4). In 1991, Sun replaced SunOS4 with Solaris, based on SVR4 instead of on BSD. (So, still a Unix derivative, just with a different parent.) Solaris included OpenWindows (a GUI) and Open Network Computing. SunOS (current release SunOS5.11) still exists as the core of Solaris (current release Solaris 11.2), but the Solaris brand is used externally.

Solaris has always been heavily associated with Sun's own SPARC hardware, but it's also used on i86pc machines worldwide, and is supported by several of the major server manufacturers including Dell, IBM and Intel. Linux distros are also available for SPARC and i86pc hardware. Since 2007, Sun has also supported the open source OpenSolaris project, although it is now known as Solaris 11 Express.

Solaris 11 is free (but not freely licenced) to download for personal use. I tried out the live CD, which was very slow to download, but once there, booted fine on a 64-bit virtual machine.

The basic terminal commands of Solaris 11 are the same as in Linux, and you can find further documentation on the Oracle website. The differences between Solaris and Linux become (in my experience) more noticeable as you delve further into the guts of the system; the average desktop user may not notice anything beyond the difference in package availability.

NeXTSTEP/Mac OS/Darwin

NeXT was founded by Steve Jobs after he was pushed out of Apple in 1985. They developed NeXTSTEP, an object-oriented, multitasking OS to run on their workstations, based on Unix (including some BSD code) and various other bits and pieces. Tim Berners-Lee developed the first browser, *WorldWideWeb*, on a NeXT cube running NeXTSTEP; *Doom* and *Quake* were also developed on NeXT machines. In 1993, OPENSTEP was created by Sun and NeXT; before Apple decided to use it as the basis of what would become Mac OS X, and bought out NeXT. (Jobs, of course, returned to Apple along with his company.)

Mac OS X is based on the XNU (X is Not Unix) kernel, developed for NeXTSTEP, with all the usual Unix commands and utilities available on the command line. The kernel has code from FreeBSD along with other improvements and changes. It's POSIX compliant, which means that many BSD/Linux/Unix packages can be recompiled for OS X with a bit of work (as with HomeBrew, Fink, and other similar projects). The core of OS X is released as the open-source Darwin. iOS is also based on OS X, and Android (Linux-based) and iOS between them have a 90% share of the smartphone market. So it's very likely that your smartphone runs a Unix-like OS, giving Unix today an unprecedented userbase.

The GNU Project

Richard Stallman started the GNU Project in 1983, with the aim of creating "a sufficient body of free software [...] to get along without any software that is not free". GNU stands for 'GNU's Not Unix': the proposed GNU operating system was Unix-like, but Unix was proprietary and GNU was to be free.

The first piece of software released by the GNU project was *GNU Emacs* (an implementation of the existing *Emacs* text editor). They had a debugger, parser, and linker; they also needed a free C compiler

SCO/Linux lawsuit

Various bits of Unix were sold on to Novell in 1993, which then sold parts of it again to what became SCO. In 2003, SCO filed a lawsuit against IBM for $1 billion (later $5 billion), claiming that IBM had transferred SCO property into Linux. Another four major lawsuits followed.

SCO's right to be identified as the 'owner' of UNIX was challenged by Novell, so SCO sued them too. Assorted legal wranglings followed. SCO also claimed that some UNIX code had been transferred line-for-line into Linux, but seemed reluctant to specify the details.

In 2010, after several court rulings and a jury trial, Novell was found to be the owner of the UNIX copyright, and has announced that "We don't believe there is Unix in Linux". As of December 2014, SCO's case against IBM for 'devaluing' its version of UNIX remains open, though now with a reduced scope.

and tools. By 1987 they had an assembler, nearly the *GCC* C compiler, *GNU Emacs*, and a bunch of utilities, together with an initial kernel. Although the rest of the software development carried on at a decent pace, by 1992 they had all the major components except the kernel. The GNU Hurd kernel started development in 1990, based on the Mach microkernel, but for various reasons moved very slowly (it is still not ready for production environments, although the existence of Linux has doubtless slowed development).

You can try out the GNU/Hurd project courtesy of Debian. Note that it is not yet complete and it's not recommended for production use. If you just want to give it a quick go, you can get a virtual image thus, and run it on *KVM*:

```
$ wget http://ftp.debian-ports.org/debian-cd/hurd-i386/current/README.txt
$ tar zxf debian-hurd.img.tar.gz
$ kvm -no-kvm-irqchip -drive file=debian-hurd*.img,cache=writeback -m 1G
Or on VirtualBox if you convert it to the correct format:
$ VBoxManage convertfromraw debian-hurd*.img debian-hurd.vdi --format vdi
```

(This information from **http://ftp.debian-ports.org/debian-cd/hurd-i386/current/README.txt**; more detailed information available there.)

The Hurd's notion of 'translators' is new to me: a translator basically translates between one sort of data structure and another, for example from disk storage to the traditional filesystem. Check out the GNU Hurd website (**https://www.gnu.org/software/hurd/index.html**) for more information on this and other features of the Hurd. If you want to install the Hurd, the instructions suggest that this is a lot like installing Linux was about 15 years ago when I first tried it out, and requires a fair amount of messing around with text files and configuring by hand. (Ah, nostalgia…) Currently only about 50% of the Debian packages are available for the Hurd.

Linux

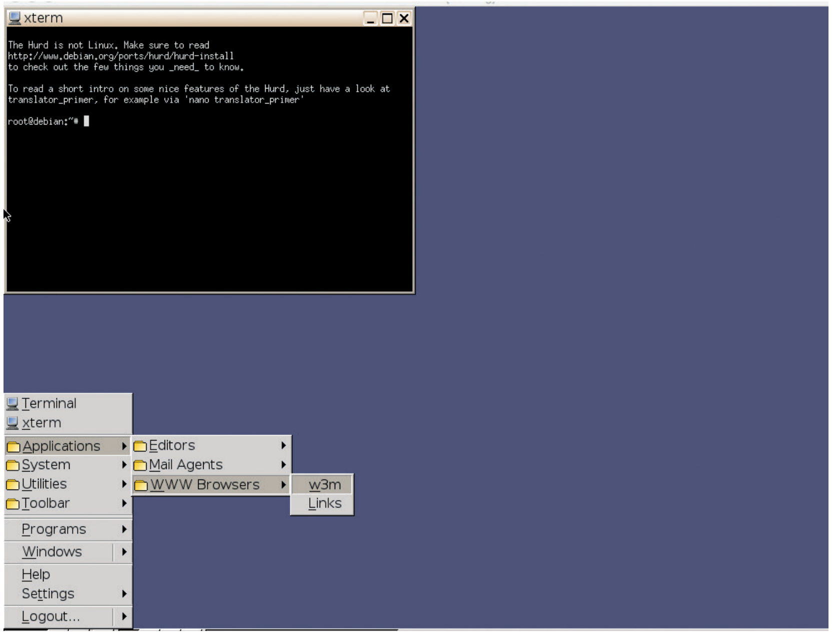
Finally, we come, of course, to Linux. Linux is not, in fact, an actual Unix variant. It's related to Minix, which was system-call compatible with Seventh Edition Unix but was created from scratch. In 1991, Linus Torvalds was irritated by the lack of a free kernel (GNU Hurd didn't exist and BSD were having problems), so started writing one. He developed it on a Minix system using the GNU C compiler, and was influenced by many Minix design decisions, but there was no actual code overlap (see the boxout for SCO's legal claims). The first release was on 25 August 1991. Unlike Minix's microkernel (a microkernel has as little software as possible in the kernel and moves functions like device drivers and filesystems into userspace), Linux has a monolithic kernel, where all the operating system is in kernel space.

Initially, it was just a kernel distribution, the idea being that you would also get hold of the GNU tools and that would give you a full system. GNU and Linux also had different licences. In Dec 1992 Linux was released under the GNU GPL, which in due course meant the whole thing could be distributed as an integrated system.

From there… well, there are a huge number of Linux distros, you can build your own, and you're reading a whole magazine dedicated to Linux. While it isn't Unix, it's largely Unix-compatible (it adheres to POSIX standards even if uncertified) and broadly speaking, if you know Linux you can find your way around Unix (though as any sysadmin will tell you there are a fair few gotchas in the details of utilities and syntax).

If you're interested in exploring the various Unixes further, try out some of the systems I tried, or one of the many others. For more on Unix, here's a cool (but huge) Unix family tree; there are also links at the bottom of this page – **www.levenez.com/unix**. And here's a Unix timeline (**www.unix.org/what_is_unix/history_timeline.html**). LV

Juliet Kemp is a scary polymath, and is the author of Apress's *Linux System Administration Recipes*.



```
The Hurd is not Linux. Make sure to read
http://www.debian.org/ports/hurd/hurd-install
to check out the few things you _need_ to know.

To read a short intro on some nice features of the Hurd, just have a look at
translator_primer, for example via 'nano translator_primer'

root@debian:~# 
```

Here's the Hurd after running **startx** from the console. All very basic by default (no graphical browser here…).

LINUX VOICE TUTORIAL

ARDUINO HARDWARE ENABLEMENT

NICK VEITCH

Plug an OLED display into your Arduino, script a driver and learn C++ library programming, all without requiring any experience.

WHY DO THIS?

- Learn how to plug awesome displays into cheap hardware and code your own interface using the Arduino IDE and a smattering of learner-level C++.

This tutorial exists because I am a lazy miser. These are two undersung qualities, which in my opinion make the best engineers, but maybe I am biased as well as lazy and a miser. In any case, because of my character flaws, I have learnt and will now pass on to you some amazing things:

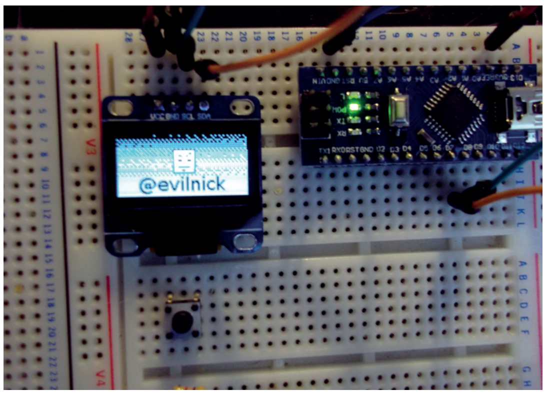
- How to use a cheap OLED display with an Arduino.
- How to create and package an Arduino library.
- How to save precious dynamic memory.
- How to save some I2C pins.
- How to do your own hardware enablement projects.
- How to convert bitmap images into code.

So, everyone's a winner. Except people who make expensive OLED displays. This is going to be the first of a two-part tutorial where we learn to tame both displays and the Arduino by writing our own code and building our own libraries. This month we're going to work out how to communicate with the screen and create a library that encapsulates those commands into a library before next month, creating lots of pretty works and pictures with our library.

Cautionary beginnings

OLED displays can easily be powered long-term by batteries, giving you freedom to use the serial port connection to find out what is going on inside its tiny mind. I wanted to build a device with a display that would log and tell me the temperature in the water tank in my attic (relayed by another Arduino), because I am too lazy to go up to the attic to find out. And that is when I happened on a cheap supply of OLED displays online, which were selling for $3 a unit instead of the $15 to $20 I was used to.

When they arrived I discovered that these displays differed significantly enough from the "standard" ones that no existing library would work with them. I could have adapted an existing library, but there were some other implementation issues. Cheapskatiness trumped laziness and I decided to write my own hardware enablement. Hurrah!



You too can delight your friends and confound your enemies by writing custom drivers for cheap displays!

I (2) See

The I2C interface is a fairly standard way of connecting microcontrollers to things. There are several slightly different ways of doing it, but a common way is to use two wires: a clock (SCL) and a dataline (SDA). By cunning signalling, both can be used to signify the beginning and end of transmissions too. The Arduino has a library (the *Wire* library) to take care of this for you, but it uses hardwired pins. The trouble is that, although you can have multiple devices on the I2C bus, there are limits, and if you want to drive more than one display, you will find they have a very limited range of addresses. For these reasons, I decided to implement the protocol in my library so I could use any pins I liked, and also to reduce the overhead on having another library (*Wire* isn't that big, but it does contain a lot of stuff we won't need). For the master device (the Arduino in our case), transmission of data goes like this:

- Bring SCL and SDA HIGH.
- Bring SDA and SCL LOW.
- Load bit into SDA.
- Pulse SCL (High then Low).
- … continue transmitting bits until end of byte.
- Send control bit (SDA HIGH) and signal end of transmission.
- Bring SCL and SDA LOW.
- Bring SCL and SDA HIGH.

The actual bits you are transmitting all follow this procedure, so we can start off by writing the low-level bits and then write higher level functions to send bytes and commands etc.

Initially, it is useful to write this code directly into an Arduino sketch – it keeps everything in one place and makes it a bit easier to test. So, our Arduino functions would look something like this:

```
void i2cStart()
{
   digitalWrite(dSCL, HIGH);
   digitalWrite(dSDA, HIGH);
   digitalWrite(dSDA, LOW);
   digitalWrite(dSCL, LOW);
}
void i2cStop()
```

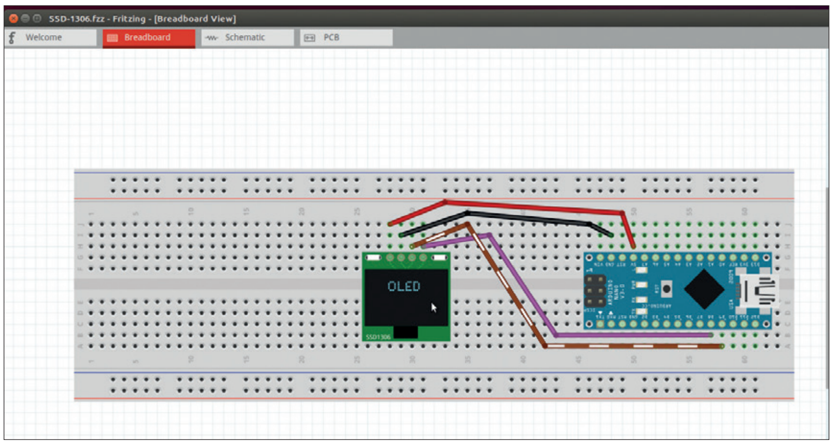
```
{
  digitalWrite(dSCL, LOW);
  digitalWrite(dSDA, LOW);
  digitalWrite(dSCL, HIGH);
  digitalWrite(dSDA, HIGH);
}
void sendByte(unsigned char b)
{
  char i;
  for(i=0;i<8;i++)
  {
          if((b << i) & 0x80){
                    digitalWrite(dSDA, HIGH);
          }else{
                    digitalWrite(dSDA, LOW);
          }
          digitalWrite(dSCL, HIGH);
          digitalWrite(dSCL, LOW);
  }
  digitalWrite(dSDA, HIGH);
  digitalWrite(dSCL, HIGH);
  digitalWrite(dSCL, LOW);
}
```

This presupposes that in the main code somewhere we define the pins (here called dSCL and dSDA) and set them as outputs. The loop in the **sendByte()** function merely uses a bitshift operator, **<<**, to iterate through the bits in the byte supplied to the function, and transmit them one at a time.

Now, to go any further than this, we need some specifics about the device we are communicating to. In this case, the OLED display uses the common SSD1306 control chip. This is a multi-protocol chip, though our hardware is hardwired to only supply a write-only I2C interface. The address of the device on the i2c bus is either 0x78 or 0x7A, which is set via connecting a pin on the SSD1306 device. Since these displays usually come on a board, you may have a jumper (with the cheap hardware I have, it is hardwired to be 0x78).

The address is important, as you need to signal this on the I2C bus to get a device to listen (remember, the bus is designed to have potential for more than one occupant). You will also need to know what commands you can and should send. For this, we need to find the datasheet for the SSD1306. A quick Google search should bring up some candidates, or you can request one direct from the manufacturers. A lot of the datasheet is not relevant to us because we will be using only one of the connection modes. What is highly relevant are the setup commands though – we need to send these to get the display to turn on and work correctly. To send a command we have to initiate the bus, send the slave address, send the control byte (telling the device we are writing to it), then send the actual command byte and close the bus again. We can wrap this up in a higher level function like this:

```
void sendCmd(unsigned char cmd)
{
  i2cStart();
  sendByte(0x78);  //Slave address,SA0=0
  sendByte(0x00);    //write command
  sendByte(cmd);
  i2cStop();
}
```



The breadboard view shows why an Arduino Nano is really very very useful for working on this sort of project – it just slots right in.

The actual list of commands to initiate the display is long and consists of things we don't need to know much about (the slew rate seems to be a function of the size of the display, and there are various different ways of addressing the memory). For the moment we can just make an array out of the commands:

```
char init_codes[] {
 0xAE,0x00,0x10,0x40,0xB0,0x81,0xCF,0xA1,
 0xA6,0xA8,0x3F,0xC8,0xD3,0x00,0xD5,0x80,
 0xD9,0xF1,0xDA,0x12,0xDB,0x40,0x8D,0x14,
 0xAF
};
```

which we would declare in the main loop, and then create a function to loop through sending these when we want to initialise the display:

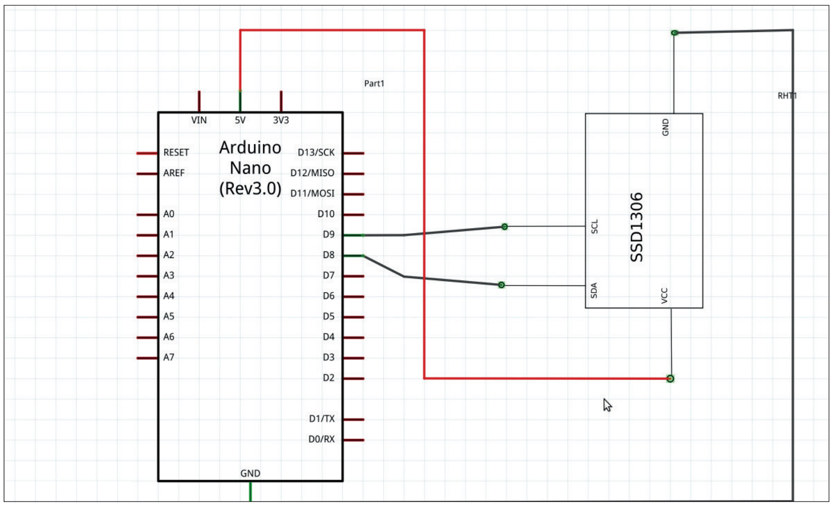
```
void init()
{
 for (i = 0;i < 25;i++) {
  sendCmd(init_codes[i])
  }
}
```

We now have enough code to bring up the display. But how will we even know that it is on? We need to stick something on it.

Addressing

The SSD1306 has three different modes of addressing the display – a horizontal mode, a vertical mode and a paged mode. The 'pages' are basically lines 8-bits deep across the display, which will be very useful for when we want to display characters. However, the horizontal mode will be useful for things like blanking the display.

Horizontal mode, it turns out, doesn't mean what you think it does. Each byte you send still corresponds to a vertical slice of 8 pixels, it just means that when you get to the end, the address pointer is updated to the beginning of the next row (see diagram). This means though that we can write 1024 bytes in



The fritzing diagram showing the connections for an Arduino Nano. It is straightforward though – just direct connections for the power and the two pins we use for I2C (in this case D8 and D9).

sequence to fill up the whole screen, which is perfect for a **cls()** type routine.

To do this we need to:

- Send the commands to initiate horizontal mode.
- Initiate the data connection.
- Send 1024 bytes.
- Close the connection.

For now, we can just build this in the main loop of the Arduino code

```
void loop() {
char init_codes[] {
 0xAE,0x00,0x10,0x40,0xB0,0x81,0xCF,0xA1,
 0xA6,0xA8,0x3F,0xC8,0xD3,0x00,0xD5,0x80,
 0xD9,0xF1,0xDA,0x12,0xDB,0x40,0x8D,0x14,
 0xAF
};
init();
sendCmd(0x20); // send the command to initiate horizontal
mode
sendCmd(0x00);
i2cStart();
sendByte(0x78); // identify the slave device
sendByte(0x40); // signal that what follows is data rather than
commands
for (unsigned int n=0;n<512;n++)
        {
     sendByte(0xAA);
     sendByte(0x55);
        }
i2cStop();
delay(1000);
}
```

This is a useful way to prototype functions that you may wish to develop – just bash them out in the main code, experiment with rationalisations and shortcuts, and then encapsulate them into a function.

Here we have initialised the display as discussed before. Then we send the commands to the device to set it into the horizontal addressing mode. To send the stream of data, we initiate the I2C connection and send the identifier byte (to address the correct device), and indicate that we are writing data to the display memory (0x40). Then we just send the bytes. In this case, a chequerboard pattern). If you keep writing after 1024 bytes, the address counters on the SSD1306 will just reset and you will end up writing at the beginning again. Now that we have verifiable code that can be proved to work with the display, it is time to look into libraries.

Oh, I C++

Arduino libraries are written in C++. This may seem surprising and daunting to some, but it shouldn't be – the Arduino code you're used to writing is basically C++, albeit a version that hides a lot of the tricky stuff and wraps everything else in a layer of simplification. The point is, that for the most part, the actual code writing part should feel familiar; it's just the structures surrounding it which will be new to some.

The simplest library consists of just two files. There will be the **.cpp** file which contains the code itself, and a **.h** or header file.To start with, we need to create a directory in the place where user libraries live. This will be (on nearly all Linux distros) in the path **~/Arduino/libraries**. The only exception is if you are in the habit of running the Arduino IDE as root, which is very naughty! The reason some people do this is because then you don't have to change permissions for some of the devices used; consider instead following the Linux instructions here: **http://playground.arduino.cc/Linux/All**.

In the **libraries** directory, simply make a new directory (I called mine **evilOLED**), then we can create some files and directories

```
$ tree
.
├──evilOLED.cpp
├──evilOLED.h
├──examples
└──utils
```

The directories (**utils** and **examples**) we can forget about for now. The first thing to do is open up your favourite code editor and start editing the **evilOLED.cpp** file. The very first thing we need to put in the file is the **include** line, which adds this file's own header:

```
#include "evilOLED.h"
```

The next important order of business is to create a class (skip this paragraph if you already know what that is!). A class is really like a special datatype. Think of it like this: instead of defining an integer or a string, we are going to define a display. Along with that we have to provide the code for all of its interactions in the form of functions. We also have to allocate space for any variables and data that instances of the class will need when they are created. The class isn't an instance, it is the recipe for creating one, in almost the same way that the Arduino code knows that an **int** is an integer, and what to do when creating one, or adding or subtracting or printing one.

Our class is defined like this:

```
evilOLED::evilOLED(char sda, char scl)
{
  _sda = sda;
  _scl = scl;
  _col = 0x00;
  _row = 0x00;
  init();
  cls(0xff);
}
```

We can explain this a little bit. The **constructor** is the first line. This is like a special function (with no return type at all) which is called whenever an instance of the class is first created. In it, we want to put definitions for any special data we want the instance to have, and any functions it should call. The argument or parameters in the constructor are special bits of data that will be passed in by the code creating the instance. In this case, we want to specify which pins we are going to use for communication.

The bit inside the braces (curly brackets **{** and **}**) is the code that will run. The first two bits may seem a little strange – we have taken the values passed in by the constructor and copied them to new variables (which haven't even got a type!). Then we set two more variables, and all of them begin with an underscore. The underscore is the convention which means that these variables are 'private' to the class – that means that only functions belonging to that class can see them – they can't be read or changed by any code outside of this class. We actually explicitly declare this in the header, but we have more to do here before we get on to that. The last two lines call other functions of the class – we will have to transfer them from the Arduino code we wrote, which should be up by now on **www.linuxvoice.com/code/lv012/arduino**.

They function are practically unchanged from the Arduino code – they have just been updated to use our new private variables, plus the function definitions now begin with **evilOLED::<name>**, which declares them as part of our class.

To complete this fairly minimal version of our library, we also need to generate the header file. This is also on **www.linuxvoice.com/code/lv012/arduino**. The opening **#ifndef** statement is a common convention which basically prevents the header file from being parsed twice, as may happen on larger projects where several code files may include it. The following block of code, up until the **#endif**, will not be processed more than once. For completeness we've include the Arduino library and the **pgmspace**. These, as it turns out, are not explicitly used by the header file, but they are used in the main C++ file. It is a common convention to put all the necessary **include**s in one place, the header file, so the main code only has one **include** in it – it just means it is easier to track things down if you aren't searching two files for it.

We have also included a **#define** statement for the slave address of the display here. This is just an example of something you may want to do in your header file. As there is more than one possible value for this, it is probably more useful to have it as a variable rather than a compiled-in value, but it is also a useful reminder of what the default value actually is.

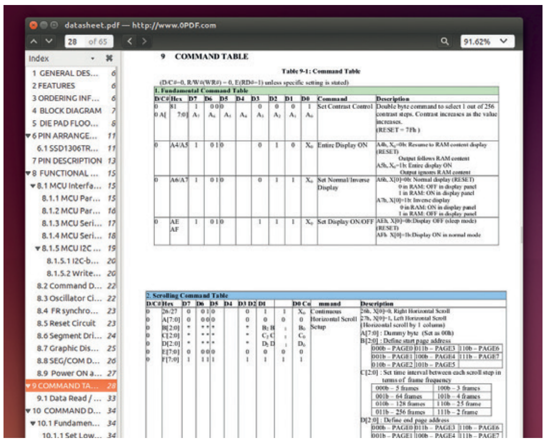
The class definition that follows is what is called a 'prototype'. This outlines the parameters accepted and returned by all the member functions (including the constructor) as well as a list of the private data or functions used by the code. Basically, to add your function to the header, you can just copy and paste it, then edit out the **evilOLED::** prefix. There is no functional code here, but this demonstrates how the code works, and if well commented, can tell you everything useful you need to know about the library.

The **examples** directory is where you will put any complete sample code using your library, which will then turn up in the relevant Arduino menu. It is a good idea to include as much, well commented, functionality as possible in these, as people tend not to read the instructions!

To make sure your library is usable by the Arduino IDE, you should put it where the rest of the libraries are. This is usually **~/arduino/libraries** but may depend on how and where you installed it. The foolproof way is to put the directory somewhere findable, then open up the IDE. Create a new sketch and then choose Sketch > Import Library > Add Library, and use the requester to specify the location of your library directory. You will find all the files are copied to wherever the library storage for Arduino happens to be. Note that you can continue to edit the library in situ – the code is recompiled each time you compile source that refers to it (uploading or checking sketches) so this can be a handy way to test changes. LV

The best Arduino?

This is a small aside about the Arduino Nano, which I have come to believe is the best model for prototyping on. The reason is simply that it comes on a board, ready to plug into a breadboard. All the pins are single pitch around the edge so it doesn't take up much room, and it still has the very useful programming header on it if you need/want to program it that way. Coincidentally, it is also quite cheap, thus fulfilling my pinchfist proclivities.



- 1 GENERAL DES... 6
- 2 FEATURES 6
- 3 ORDERING INF... 6
- 4 BLOCK DIAGRAM 7
- 5 DIE PAD FLOO... 8
- 6 PIN ARRANGE... 11
  - 6.1 SSD1306TR... 11
- 7 PIN DESCRIPTION 13
- 8 FUNCTIONAL ... 15
  - 8.1 MCU Interfa... 15
    - 8.1.1 MCU Par... 15
    - 8.1.2 MCU Par... 16
    - 8.1.3 MCU Seri... 17
    - 8.1.4 MCU Seri... 18
    - 8.1.5 MCU I2C ... 19
      - 8.1.5.1 I2C-b... 20
      - 8.1.5.2 Write... 20
  - 8.2 Command D... 22
  - 8.3 Oscillator Ci... 22
  - 8.4 FR synchro... 23
  - 8.5 Reset Circuit 23
  - 8.6 Segment Dri... 24
  - 8.7 Graphic Dis... 25
  - 8.8 SEG/COM D... 26
  - 8.9 Power ON a... 27
- 9 COMMAND TA... 28
  - 9.1 Data Read / ... 33
- 10 COMMAND D... 34
  - 10.1 Fundamen... 34
    - 10.1.1 Set Low... 34

9 COMMAND TABLE

Table 9-1: Command Table

(D/C#=0, R/W#(WR#) = 0, E(RD#=1) unless specific setting is stated)

1. Fundamental Command Table

D/C#	Hex	D7	D6	D5	D4	D3	D2	D1	D0	Command	Description
0 0	81 A[7:0]	1 A_7	0 A_6	0 A_5	0 A_4	0 A_3	0 A_2	0 A_1	1 A_0	Set Contrast Control	Double byte command to select 1 out of 256 contrast steps. Contrast increases as the value increases. (RESET = 7Fh)
0	A4/A5	1	0	1	0	0	1	0	X_0	Entire Display ON	A4h, X_0=0b: Resume to RAM content display (RESET) Output follows RAM content A5h, X_0=1b: Entire display ON Output ignores RAM content
0	A6/A7	1	0	1	0	0	1	1	X_0	Set Normal/Inverse Display	A6h, X[0]=0b: Normal display (RESET) 0 in RAM: OFF in display panel 1 in RAM: ON in display panel A7h, X[0]=1b: Inverse display 0 in RAM: ON in display panel 1 in RAM: OFF in display panel
0	AE AF	1	0	1	0	1	1	1	X_0	Set Display ON/OFF	AEh, X[0]=0b:Display OFF (sleep mode) (RESET) AFh X[0]=1b:Display ON in normal mode

2. Scrolling Command Table

D/C#	Hex	D7	D6	D5	D4	D3	D2	D1	D0	Command	Description
0	26/27	0	0	1	0	0	1	1	X_0	Continuous Horizontal Scroll Setup	26h, X[0]=0, Right Horizontal Scroll 27h, X[0]=1, Left Horizontal Scroll (Horizontal scroll by 1 column)
0	A[7:0]	0	0	0	0	0	0	0	0		A[7:0] : Dummy byte (Set as 00h)
0	B[2:0]	*	*	*	*	*	B_2	B_1	B_0		B[2:0] : Define start page address 000b – PAGE0 011b – PAGE3 110b – PAGE6 001b – PAGE1 100b – PAGE4 111b – PAGE7 010b – PAGE2 101b – PAGE5
0	C[2:0]	*	*	*	*	*	C_2	C_1	C_0		C[2:0] : Set time interval between each scroll step in terms of frame frequency 000b – 5 frames 100b – 3 frames 001b – 64 frames 101b – 4 frames 010b – 128 frames 110b – 25 frame 011b – 256 frames 111b – 2 frame
0	D[2:0]	*	*	*	*	*	D_2	D_1	D_0		D[2:0] : Define end page address 000b – PAGE0 011b – PAGE3 110b – PAGE6 001b – PAGE1 100b – PAGE4 111b – PAGE7
0	E[7:0]	0	0	0	0	0	0	0	0		
0	F[7:0]	1	1	1	1	1	1	1	1		

The datasheet is useful, even though a lot of it doesn't apply. there is some info about other modes you may care to implement, such as hardware scrolling!

Nick Veitch has edited computer magazines for 1,000 years. He now works at Canonical and collects gin bottles.

LINUX VOICE TUTORIAL

CODE NINJA: RESTFUL APIS

BEN EVERARD

Access resources from around the web and integrate them into your own site in a simple, uniform manner.

WHY DO THIS?

- Understand how huge databases handle billions of transactions.
- Gain the flexibility of not having a schema.
- Use the trendiest database in town.

The web is a great way of sharing data. Anyone, anywhere in the world can upload information and make it instantly available to almost everyone in the world. However, as wonderful as these web pages are, they do have a problem: it's hard for computers to understand them. Web browsers can obviously render web pages, but they can't easily extract information. Take, for example, a web page of a weather forecast. The browser knows what text to put where, and which images should be displayed, but it's hopeless at understanding the forecast, and can't easily pull out the data so it can be displayed in a different forecast.

When a person or organisation wants to make their information available for computers to understand it, they need to create an API (Application Program Interface). This is a method for programs to extract the information they need in a computer-readable format. The most popular method for doing this on the internet is through REpresentational State Transfer (REST) APIs.

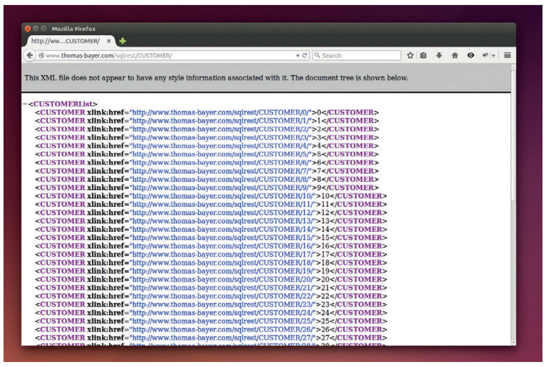
A RESTful API must be client–server; stateless; cacheable; layered; code on demand (optional); and have a uniform interface.

Let's look at these properties by analysing the largest RESTful system, the world-wide web. This is a series of HTML documents that are sent and received using Hyper Text Transfer Protocol (HTTP). The web is client–server. This means there's a separation between the software that browses the web (such as *Firefox*) and the software that serves the web pages (such as *Apache*).

The web (or rather, web servers) is stateless. This means that if you make the same request, you should get the same result. It doesn't matter what requests you've sent before this. This still allows for things like authentication, because this session data can be sent in the HTTP request along with the URL.

Web resources can be cacheable. To statisfy the RESTful requirements, not everything has to be cached, but information needs to be included about what can and can't be cached. This is taken care of in HTTP headers.

When you connect to a website, you could be connecting directly to the server, or through some proxy. This is all handled transparently because HTTP is layered. Code on demand is the only optional aspect of the REST principals. However, the web does allow it. This is when the server sends some code to be executed on the client. In terms of the web, this is usually JavaScript or a Java applet.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
-<CUSTOMERList>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/0/">0</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/1/">1</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/2/">2</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/3/">3</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/4/">4</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/5/">5</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/6/">6</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/7/">7</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/8/">8</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/9/">9</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/10/">10</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/11/">11</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/12/">12</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/13/">13</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/14/">14</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/15/">15</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/16/">16</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/17/">17</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/18/">18</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/19/">19</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/20/">20</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/21/">21</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/22/">22</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/23/">23</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/24/">24</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/25/">25</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/26/">26</CUSTOMER>
   <CUSTOMER xlink:href="http://www.thomas-bayer.com/sqlrest/CUSTOMER/27/">27</CUSTOMER>
```

The XML pages are viewable in a web browser, but the links aren't clickable, so you'll have to copy and paste them into the URL bar.

URLs are in a uniform structure, and there's a uniform series of four HTTP requests that you can perform on them (GET, PUT, POST and DELETE) . Although four are all available in theory, in practice almost all websites only use GET and POST, the latter of which is used when you submit a form.

There's an example API that links an SQL database to a RESTful service at **www.thomas-bayer.com/sqlrest**. If you point your browser there, you'll see an XML document describing all the resources available.

You should notice that they're all resources one level above **/sqlrest/**. This is part of the uniform architectural constraint. If you point your browser to one of the listed resources – **www.thomas-bayer.com/sqlrest/CUSTOMER/** – you should then see a list of the customers. Again, each listed resource is one level above **/sqlrest/CUSTOMER/**.

This is a public API that allows anyone to change and delete information without authentication, so it's possible that the exact examples we use now won't exist any longer in the database. If they don't, you'll have to substitute in other values.

You can view one of the customers by viewing a link in this list, for example, **www.thomas-bayer.com/sqlrest/CUSTOMER/16**. Unlike the other queries we've done, this doesn't just return a list of resources, but the actual customer information as an XML file.

We can see that the format of the URLs for this database is: **www.thomas-bayer.com/sqlrest/<table>/<id>/**. Using this, we can extract any information from the database. We can also manipulate the database using other HTTP methods. POST, PUT and DELETE map to amending a record, adding a record and deleting a record respectively. However, you can only generate GET requests using the URL bar of most web browsers, so we'll need another tool to send these requests. There's a web-based HTTP tool at **http://thomas-bayer.com/restgate/** that will do what we need.

To add a new item to the database, we use the PUT method on the URL for the table. So, for example, to add a new person at ID 0, you'll need to go to **http://thomas-bayer.com/restgate/**, then enter the URL **http://www.thomas-bayer.com/sqlrest/CUSTOMER**, select the method PUT, then in the content text box (that will appear when you select PUT), enter the following XML:

```
<CUSTOMER>
<ID>0</ID>
<FIRSTNAME>Ben</FIRSTNAME>
<LASTNAME>Everard</LASTNAME>
<STREET>xxx</STREET>
<CITY>Brizzle</CITY>
</CUSTOMER>
```

Note that you'll have to use a different ID when you do it as this one's taken. At the time of writing, this request returned an error, however, the actual record was updated. You can see for yourself at **http://www.thomas-bayer.com/sqlrest/CUSTOMER/0/**.

Using the same web form, you can amend a customer's details. This time, enter the URL **http://www.thomas-bayer.com/sqlrest/CUSTOMER/0/** (again, you'll need to change the ID to match one you've created), the method POST and the content:

```
<STREET>Colston Av</STREET>
```

Finally, you can use DELETE to remove customers.

So far, what you've seen is a slightly awkward way of accessing a database through a browser. This isn't at all the point of a RESTful API. The point is to make it easy for other software to interact with our service.

Doing it in Python

In Python, we can use the **requests** module to interact with this RESTful API. This is easy to use, so we'll use an Python interactive session, which you can start by typing **Python2** at the command line. Then you need to import the **requests** module with:

```
>>> import requests
```

A GET request is then simply done with:

```
>>> r = requests.get("http://www.thomas-bayer.com/sqlrest/CUSTOMER/0/")
>>> print r.content
```

This will print out the raw XML. If you were doing this in a real program, you could either manipulate the XML as strings, or you could use one of the XML parsing modules to do it for you.

The requests module can also issue POST, PUT and DELETE requests.

```
>>> mike_data='''<CUSTOMER>
...     <ID>897324</ID>
...     <FIRSTNAME>Mike</FIRSTNAME>
...     <LASTNAME>Saunders</LASTNAME>
...     <STREET>xxx</STREET>
...     <CITY>Munich</CITY>
... </CUSTOMER>'''
>>> r = requests.put("http://www.thomas-bayer.com/sqlrest/CUSTOMER/", data=mike_data, headers={'Content-Type':'application/xml'})
>>> r = requests.get("http://www.thomas-bayer.com/sqlrest/CUSTOMER/897324/")
>>> print r.content
```

This should display the new record we've just created for Mike Saunders. We can now update it to his nicknames with:

```
>>> r = requests.post("http://www.thomas-bayer.com/sqlrest/CUSTOMER/897324/", data="<FIRSTNAME>Mikeyboy</FIRSTNAME>", headers={'Content-Type':'application/xml'})
>>> r = requests.get("http://www.thomas-bayer.com/sqlrest/CUSTOMER/897324/")
>>> print r.content
```

Then delete it with:

```
>>> r = requests.delete("http://www.thomas-bayer.com/sqlrest/CUSTOMER/897324/")
```

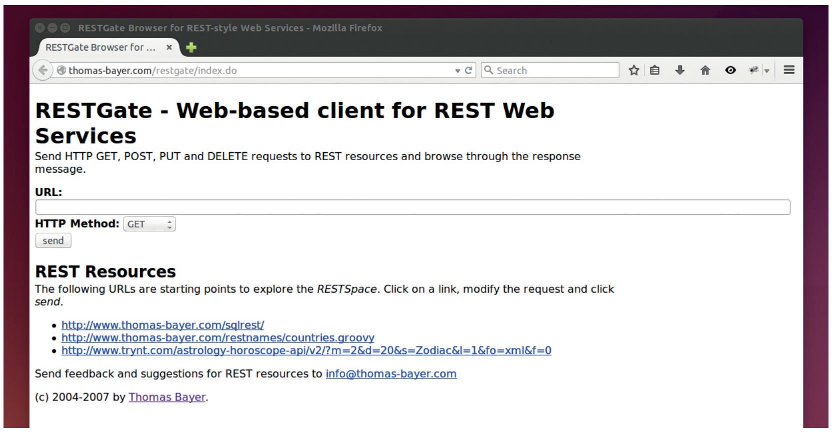
In this case, we've been updating a database, but RESTful APIs exist for all sorts of data source. You should be able to use the same methods we've used here to access the vast majority of them which use HTTP. In many cases, you'll need to authenticate yourself before you can perform any action (especially those that modify data), but this should be fully documented as it differs between APIs.

To go back to the first problem, how could our program get a weather forecast:

```
>>> r = requests.get("http://api.openweathermap.org/data/2.5/weather?q=Bristol&mode=xml"
)

>>> print r.content
```

For more information on this weather API, go to **http://openweathermap.org/current**.

We've looked at Python here, but you should find equivalent libraries in most languages that make it just as simple. Once you've mastered sending HTTP requests, the process is quite straightforward. It's this simple approach that makes HTTP-based RESTful APIs so simple and ubiquitous.


RESTGate Browser for REST-style Web Services - Mozilla Firefox

RESTGate Browser for ...

thomas-bayer.com/restgate/index.do

RESTGate - Web-based client for REST Web Services

Send HTTP GET, POST, PUT and DELETE requests to REST resources and browse through the response message.

URL:

HTTP Method: GET

send

REST Resources

The following URLs are starting points to explore the *RESTSpace*. Click on a link, modify the request and click *send*.

- http://www.thomas-bayer.com/sqlrest/
- http://www.thomas-bayer.com/restnames/countries.groovy
- http://www.trynt.com/astrology-horoscope-api/v2/?m=2&d=20&s=Zodiac&l=1&fo=xml&f=0

Send feedback and suggestions for REST resources to info@thomas-bayer.com

(c) 2004-2007 by Thomas Bayer.

The RESTgate webpage can be used to interact with almost any HTTP RESTful API, not just the ones we've used here.

LINUX VOICE TUTORIAL

ASMSCHOOL: GET STARTED WITH ASSEMBLY LANGUAGE

MIKE SAUNDERS

Part 1: Explore beyond the limits of high-level languages and discover exactly how your CPU really ticks.

WHY DO THIS?
- Learn what compilers do behind the scenes.
- Understand the language of CPUs.
- Fine-tune your code for better performance.

Most people see assembly language as some kind of black magic, part of a dark and scary world where only the top 0.01% of developers ever dare roam. But it's actually a fascinating and surprisingly accessible subject. It's also well worth learning the basics to help you understand how compilers generate code, what CPUs actually do, and get a good all-round picture of how computers work. Assembly language is ultimately a textual representation of the instructions that the CPU executes, with some extra bits 'n bobs to make programming easier.

Nobody in their right mind would write a large desktop application in assembly language today. It'd be monstrously complicated, very hard to debug, and a colossal effort to port to other architectures. But assembly is still used in various places: many drivers in the Linux kernel have chunks written in assembly, partly because it's the best language to use when you're interfacing directly with hardware, and partly for speed reasons. In certain cases, hand-written assembly language can perform better than code generated by a compiler.

Over the next few issues we'll delve into the world of assembly language. We'll explain the basics here, move onto some more advanced logic next month, and finish up with a simple bootable operating system – it won't do a great amount, but it'll be your own code, running on bare hardware, with no other OS required. Sounds good, right? Let's go...

1 YOUR FIRST ASSEMBLY PROGRAM

Many assembly guides start off with huge, bland, tiresome chapters that spend ages talking about binary arithmetic and CPU design theory, before even showing any real code. That's no fun, so we'll get straight into a program. Then we'll go through it step by step so that you can learn how assembly works from a practical example.

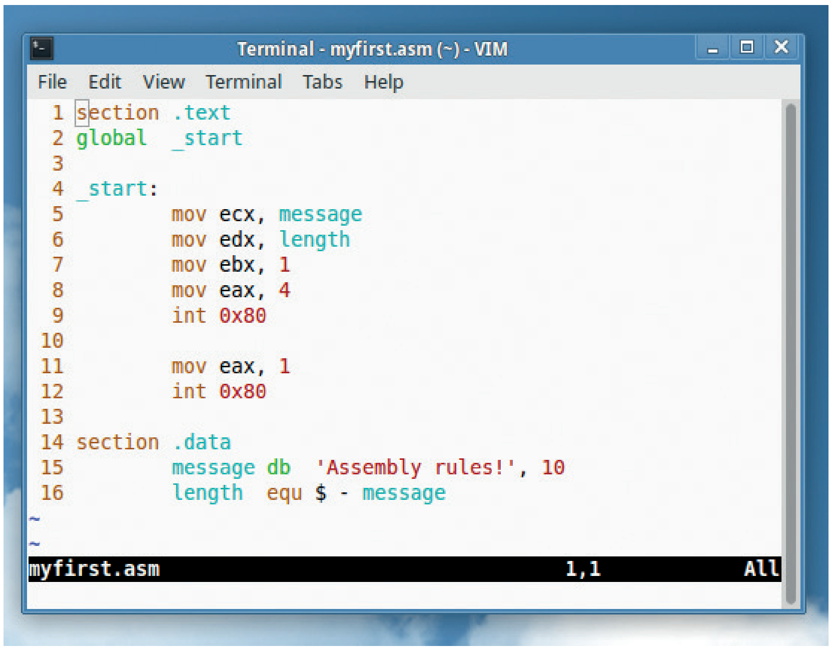
Type this into a plain text editor and save it as **myfirst.asm** in your home directory:

```
section	.text
global	_start

_start:
	mov ecx, message
	mov edx, length
	mov ebx, 1
	mov eax, 4
	int 0x80

	mov eax, 1
	int 0x80

section .data
	message	db 'Assembly rules!', 10
	length	equ $ – message
```

(Note: you can use tabs to indent the code here, or whitespace – it doesn't matter.) This program simply prints the text "Assembly rules!" onto the screen, and then exits.

The tool we're going to use to convert this assembly code into an executable binary file is called, funnily enough, an assembler. There are many assemblers out there, but our favourite is *NASM*; it's available in almost every distro's package repositories, so get it via your graphical package manager, or **yum install nasm**, or **apt-get install nasm**, or whatever's best for your distro.

Now open a terminal window and enter the following commands:


Terminal - myfirst.asm (~) - VIM
File Edit View Terminal Tabs Help

```
section .text
global  _start

_start:
        mov ecx, message
        mov edx, length
        mov ebx, 1
        mov eax, 4
        int 0x80

        mov eax, 1
        int 0x80

section .data
        message db  'Assembly rules!', 10
        length  equ $ - message
~
~
myfirst.asm                                    1,1          All
```

Some text editors, such as **Vim**, include syntax highlighting for assembly (try **set syn=nasm**).

```
nasm -f elf -o myfirst.o myfirst.asm
ld -m elf_i386 -o myfirst myfirst.o
```

The first command here uses *NASM* to generate an object (executable) file called **myfirst.o**, in ELF format (the executable format used on Linux). What exactly is an executable format, you might be asking – why not just use plain binary CPU instructions for the CPU to execute? Well, you could use plain binary back in the 80s, but modern operating systems have more demanding requirements. ELF binaries include information for debugging, they split up code and data into separate sections to stop one from overwriting another, and so forth.

Later in this tutorial series, when we look at writing code to run on bare metal (our mini operating system), we'll explore plain binaries.

A link to the past

So now we have **myfirst.o**, the executable code containing our program. It's not quite complete yet, though; using **ld**, the linker, we link it with some system startup code (ie boilerplate code that's run at the start of every program) to generate an executable called **myfirst**. (The **elf_i386** bit describes the exact binary format – in this case, it means you can use 32-bit assembly code, even if you're running a 64-bit distro.)

If everything has gone smoothly, you can now execute your program as follows:

```
./myfirst
```

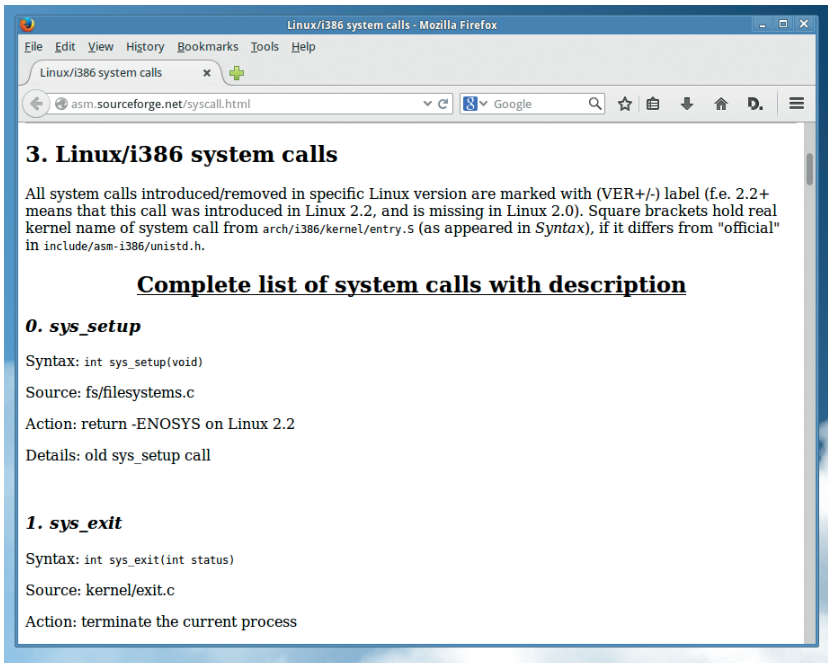
And you should see this output: "Assembly rules!" And there we have it – a complete, standalone Linux program, written entirely in assembly language. Sure, it doesn't do very much, but it's a good way to get started and get an overview of the structure of an assembly program, and see how the source code is converted into binary.

But before we dive into the code itself, it's useful to check out the size of the program. Enter **ls -l myfirst** and you'll see that it's around 670 bytes. Now consider the size of the C equivalent:

```
#include <stdio.h>

int main()
{
		puts("Assembly rules!");
}
```

If you save that as **test.c**, compile it (**gcc -o test test.c**) and look at the resulting **test** binary, you'll see that it's much larger – 8.4k. You can remove some debugging information (**strip -s test**) but it still remains around 6k. This is because *GCC* adds a lot more of the aforementioned boilerplate startup and shutdown code, and also links to a large C library. But it also demonstrates why assembly is the best language to use when space is tight.

Many assembly programmers make good money writing code for extremely restricted embedded devices, for instance, and it's why assembly was the only real choice for writing games back on the old 8-bit consoles and home computers.


Linux/i386 system calls - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Linux/i386 system calls
asm.sourceforge.net/syscall.html
Google

3. Linux/i386 system calls

All system calls introduced/removed in specific Linux version are marked with (VER+/-) label (f.e. 2.2+ means that this call was introduced in Linux 2.2, and is missing in Linux 2.0). Square brackets hold real kernel name of system call from `arch/i386/kernel/entry.S` (as appeared in *Syntax*), if it differs from "official" in `include/asm-i386/unistd.h`.

Complete list of system calls with description

0. sys_setup

Syntax: `int sys_setup(void)`

Source: fs/filesystems.c

Action: return -ENOSYS on Linux 2.2

Details: old sys_setup call

1. sys_exit

Syntax: `int sys_exit(int status)`

Source: kernel/exit.c

Action: terminate the current process

Using system calls, you can ask the kernel to perform various tasks relating to files and text input/output.

Disassembling code

Writing new code is fun, but it can be even more interesting to pick apart someone else's work. Using a tool called **objdump** (part of the Binutils package), you can "disassemble" an executable file – essentially turning the binary CPU instructions into their text-mode equivalents. Try it on the **myfirst** example we've been working on in this tutorial, like so:

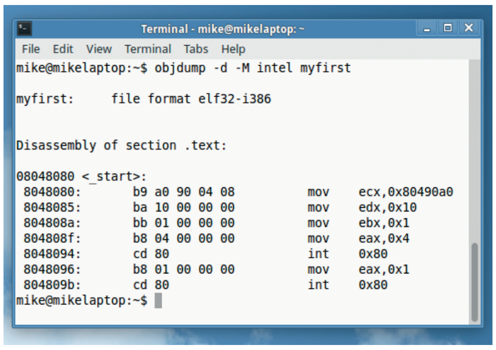
```
objdump -d -M intel myfirst
```

You'll see a list of instructions from the text section of the binary. For instance, the first instruction, where we placed the location of our string into the **ecx** register, looks like this:

```
mov ecx,0x80490a0
```

During assembly, NASM replaced "message" with the actual numerical location of the string in the data section. So disassembled binaries are less useful than the original code, as they're missing things like comments and labels, but they can be useful to see how a program implements a time-critical routine, or performing hacks. Back in the 80s and 90s, many coders used disassembly tools to identify and remove copy protection routines from games, for instance.

You can also disassemble programs written in other languages, but the results can be immensely complicated. Run the above **objdump** command on **/bin/ls**, for instance, and you'll see many thousands of lines of code in the text section, generated by the compiler from the original C source.


Terminal - mike@mikelaptop: ~
File Edit View Terminal Tabs Help

```
mike@mikelaptop:~$ objdump -d -M intel myfirst

myfirst:     file format elf32-i386


Disassembly of section .text:

08048080 <_start>:
 8048080:       b9 a0 90 04 08          mov    ecx,0x80490a0
 8048085:       ba 10 00 00 00          mov    edx,0x10
 804808a:       bb 01 00 00 00          mov    ebx,0x1
 804808f:       b8 04 00 00 00          mov    eax,0x4
 8048094:       cd 80                   int    0x80
 8048096:       b8 01 00 00 00          mov    eax,0x1
 804809b:       cd 80                   int    0x80
mike@mikelaptop:~$
```

Here's the disassembly of our sample program, showing hexadecimal codes and the instructions.

2 ANALYSING THE CODE

So, let's now see what each line of our program actually does. We start off with these two:

```
section .text
global  _start
```

These are not CPU instructions, but directives given to the *NASM* assembler; the first one says that the following code belongs in the "text" section of the final executable. Slightly confusingly, the text section doesn't contain plain text (like our "Assembly rules" string), but executable code instead – ie CPU instructions. Next we have **global _start**, which tells the **ld** linker where execution should begin in our file. This is useful if we don't want to start execution right at the beginning of the text section, but somewhere else. The **global** part makes it readable by other tools than just the assembler, so that **ld** can see it.

"Assembly language is really just a bunch of mnemonics for CPU instructions."

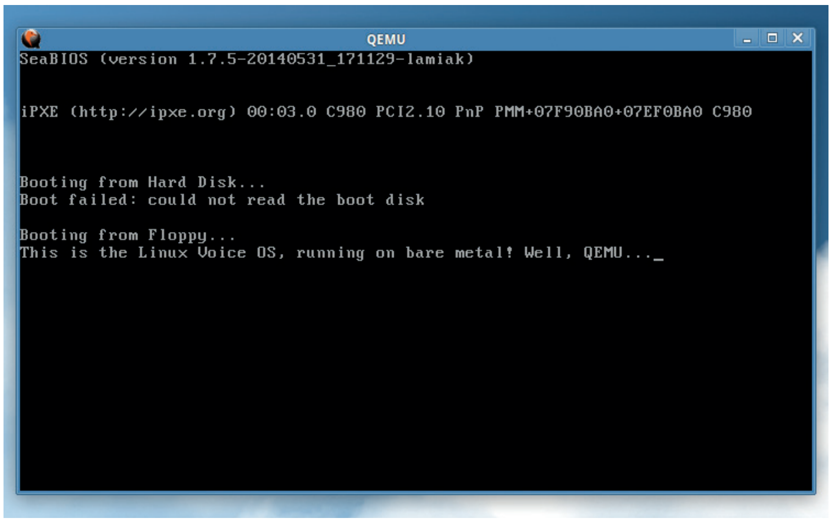
So, we've said that execution should begin at the **_start** position. And then we define this location explicitly in our code:

```
_start:
```

Standalone words ending in colons are known as labels, and are locations in the code we can jump to (more on that next issue). So, program execution begins here! And now we come to our first real CPU instruction:

```
mov ecx, message
```

Assembly language is essentially a bunch of mnemonics for CPU instructions (aka machine code). Here, **mov** is one such instruction – it could also be written in the raw binary format that the CPU understands, like 10001011. But working with raw binary would be a nightmare for us puny humans, so we use these slightly more readable variants instead. The assembler simply converts the text instructions to their binary equivalents – although it can do more, as we'll see in later tutorials.



```
QEMU
SeaBIOS (version 1.7.5-20140531_171129-lamiak)


iPXE (http://ipxe.org) 00:03.0 C980 PCI2.10 PnP PMM+07F90BA0+07EF0BA0 C980


Booting from Hard Disk...
Boot failed: could not read the boot disk

Booting from Floppy...
This is the Linux Voice OS, running on bare metal! Well, QEMU..._
```

And just as a teaser for what's to come: here's bare-metal code, running on a PC emulator! And we'll show you how to boot it on a real box...

Anyway, to understand this line of code, we also need to understand the concept of registers. CPUs don't do anything especially fancy – they simply move memory around, perform calculations on it, and then do other operations depending on the results. The CPU has no idea what a display is, or a mouse, or a printer. It simply moves data around and performs a few calculations.

Now, the main storage area for data that the CPU uses is your RAM banks. But because RAM is held outside of the CPU, accessing it takes a lot of time. To make things faster and simpler, the CPU includes its own small group of memory cells called registers. CPU instructions can use these registers directly, and in this line of code we're using the register called **ecx**.

This is a 32-bit register (so it can store numbers from zero to 4,294,967,295). You'll see in subsequent lines of code that we also work with **edx**, **ebx** and **eax** – these are all general-purpose registers that we can use for any task, as opposed to special registers that we'll explore next month. And if you're wondering where the names came from: **ecx** started as **c** in the 8-bit days, when was extended to **cx** for 16-bit, and then **ecx** for 32-bit. So the names look a bit odd now, but back on the old CPUs, you had nicely named general-purpose registers like **a**, **b**, **c** and **d**.

Moving on up

Back to the code: the **mov** instruction moves (actually, copies) a number from one place to another, from right to left. So in this case, we're saying "place message in the **ecx** register". But what is message? It's not another register – it's a location. Down at the bottom of the code, in the data section, you'll see the **message** label followed by **db**, which defines some bytes that are placed at the **message** location in the code. This is really useful, as we don't need to know the exact location of the "Assembly rules" string in the data section – we can just refer to it via the **message** label. (The number 10 after our string is simply a newline character, like adding **\n** to a C string.)

So, we've moved that location into our **ecx** register. But what we're about to do is especially cool. As mentioned, the CPU has no real concept of hardware devices – to print something on the screen, you have to send data to the video card, or move data into video RAM. We have absolutely no idea where this video RAM is, and everyone has a different video card, X server configuration, window manager etc. So directly printing something on the screen is, for us, virtually impossible in a short program.

So what we do is tell the kernel to do it for us. The Linux kernel has a bunch of system calls that low-level programs can use, to get the kernel to do various jobs. One of these calls is to output a string of text. Then the kernel handles it all – and indeed, it offers an even deeper layer of abstraction, in that it can output

Once you pop, you can't stop

One thing we'll be looking at next month is the stack, so we'll get you prepared for it here. The stack is an area of memory where temporary values can be placed, when you need to free up your registers for other purposes. But the most important feature of the stack is the way data is stored in it: you "push" numbers onto the stack, and "pop" them out. It's LIFO (last in, first out), so the most recent item you place onto it is the most recent you'll pull out.

Imagine you have an empty Pringles can, for instance, and you place into it a cream cracker, an Alf Pog and a GameCube disc – in that order. If you now retrieve the items, the first will be the GameCube disc, the second the Alf Pog, and so forth. Here's how it works in assembly language:

```
push 2
push 5
push 10
pop eax
pop ebx
pop ecx
```

After these six instructions, **eax** will contain 10, **ebx** will contain 5, and **ecx** 2. So the stack is a great way to liberate some space temporarily; if you have some important values in **eax** and **ebx**, for instance, but need to do a quick job with them, you can push them onto the stack, do your work, and then pop back the previous values to return to your previous state.

The stack is also used when calling subroutines, in that the return address of the code is pushed onto the stack. This is why you have to be careful when using the stack – if you overwrite it with the wrong data, you might not be able to return to a previous place in the code, and you've taken a one-way trip to crashland!

the string to a plain text terminal, or via a terminal emulator in the X Window System, or even redirect it to an open file.

Before we ask the kernel to print the string, though, we need to provide it with more information than just the string location in the **ecx** register. We also need to tell it how many characters to print, so that it doesn't just keep printing after the end of our string. That's when this line comes into play, in the data section towards the bottom:

```
length equ $ – message
```

Here we have another label, **length**, but instead of using **db** to include some data next to it, we use **equ** to say it's the equivalent of something (a bit like **#define** in C). The dollar sign refers to the current location in the code, so here we're saying: **length** should be equal to the current location in the code, minus the "message" location. In other words, this gives us the length of the "message" string.

Back in the main code, we put this value into the **edx** register:

```
mov edx, length
```

So far so good: two registers are populated with the string location and the number of characters to print. But before we tell the kernel to do its work, we need to provide a bit more information. First, we need to tell the kernel which "file descriptor" to use – in other words, where the output should go to. This topic is beyond the scope of this assembly tutorial, but we need to use **stdout**, which basically means: print to the screen. The number for this is 1, and we put it in the **ebx** register.

Now we're tantalisingly close to using the kernel, but there's one more register to fill. The kernel can actually do lots of different things, such as mounting filesystems, reading data from files, deleting files and so forth. These facilities are provided by the aforementioned system calls, and before we hand control over to the kernel, we need to tell it which call to use. If you look at **http://asm.sourceforge.net/syscall.html**, you'll see some of the many calls available to programs – in our case we want **sys_write** ("write to a file descriptor"), which is number 4. So we place that in the **eax** register:

```
mov eax, 4
```

And that's it! We've set up everything we need to use a kernel system call, so now we hand control over like so:

```
int 0x80
```

Here, **int** stands for "interrupt", and literally interrupts the current program flow by jumping into the kernel. (The 0x80 is hexadecimal here – you don't need to concern yourself with it for now.) The kernel prints the string pointed to in the **ecx** register, then hands control back to our program.

To end our program, we need to call the kernel's **sys_exit** routine, which has the value 1. So we place that number in **eax**, interrupt our program again, and the kernel neatly terminates our program, putting us back at the command prompt. There you have it: a complete (albeit very small) assembly language program, hand written and without the need to use any fat libraries.

We've raced through a lot in this tutorial, and as mentioned, we could've focused entirely on theory instead. But we hope you've found it useful to see a real example in action, and next issue we'll spend more time with some of the concepts we've introduced here. But we'll also take it up a notch, by adding logic and subroutines to our program – assembly versions of IFs and GOTOs.

Meanwhile, as you get familiar with this program, here are some things you can try:

- Print a different, longer string.
- Print two strings, one after another.
- Change the exit code that the program hands back to the shell (this could take a bit of Googling!).

If you get stuck or need any help, pop by our forums on **http://forums.linuxvoice.com** – this author will be at hand to point you in the right direction. Happy hacking!

Mike Saunders has written a whole OS in assembly (http://mikeos.sf.net) and is contemplating a Pi version.

LINUX VOICE MASTERCLASS

You wouldn't want other people opening your letters and your data is no different. Encrypt it today!

SECURE EMAIL WITH GNUPG AND ENIGMAIL

Send encrypted emails from your favourite email client.

JOHN LANE

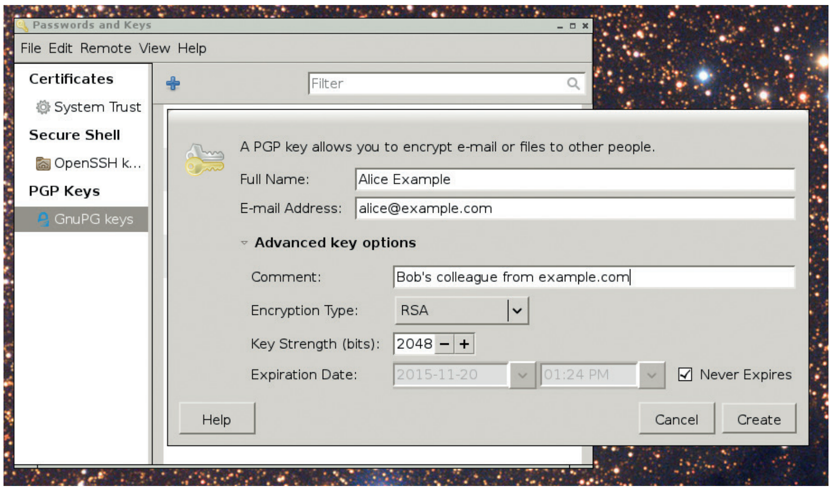
Your typical email is about as secure as a postcard, which is good news if you're a government agency. But you wouldn't use a postcard for most things sent in the post; you'd use a sealed envelope. Email is no different; you just need an envelope – and it's called "Encryption".

Since the early 1990s, the main way to encrypt email has been PGP, which stands for "Pretty Good Privacy". It's a protocol for the secure encryption of email that has since evolved into an open standard called OpenPGP.

My lovely horse

The *GNU Privacy Guard* (*GnuPG*), is a free, GPL-licensed implementation of the OpenPGP standard (there are other implementations, both free and commercial – the PGP name now refers to a commercial product owned by Symantec). An easy way to get started with encryption is to use *Seahorse*, a *GTK* graphical user interface for *GnuPG* that should be available in your chosen distribution's package repository. It's installed by default on Ubuntu, where it's called "Passwords and Keys".

Seahorse makes it easy to manage PGP keys, and its step-by-step key creation dialog is a good way to get started.



A PGP key allows you to encrypt e-mail or files to other people.

Full Name: Alice Example

E-mail Address: alice@example.com

Advanced key options

Comment: Bob's colleague from example.com

Encryption Type: RSA

Key Strength (bits): 2048

Expiration Date: 2015-11-20 01:24 PM ☑ Never Expires

The first thing that you need to do is create a key to represent your identity in the *OpenPGP* world. You'd typically create one key per identity that you have. Most people would have one identity, being themselves as a person. However, some may find having separate personal and professional identities useful. It's a personal choice, but starting with a single key will help while you're learning.

Launch *Seahorse* and click on the large plus-sign icon that's just below the menu. Select 'PGP Key' and work your way through the screens that follow to supply your name and email address and then generate the key.

You can, optionally, use the Advanced Key Options to add a comment that can help others identify your key and to select the cipher, its strength and set when the key should expire.

The final step requires you to supply a passphrase – you'll need to supply this later on when you use your key. After doing so, a new key is generated in the background. There's no feedback while this is happening (which gives the impression that it isn't working) but, eventually, you should see your key in the main *Seahorse* window (select 'GnuPG Keys' in the main window to see it).

Get the key...

Key generation can take a few minutes, because the algorithms need to collect sufficient entropy (randomness) to generate a secure key. If your system struggles to provide this, you can install a package called **haveged** from your distribution's repository; it's a daemon that gathers entropy from unpredictable background system events and will quickly deliver the entropy needed to satisfy key generation.

Once you have a key, you can right-click to view its properties, which will show you the details that you supplied when creating it, plus its Key ID. This is shown as an eight-digit hexadecimal number but the actual key ID is 64 bits. The lower 32 bits are usually sufficient to identify a key (you can see the full key ID on the Details tab).

Know the secret...

Your key is complex: it contains a secret part that you should keep for yourself, plus a public part that you can share with others. Someone wishing to send you encrypted email will need your public key. If you look at the Details tab, you'll see that you actually have two keys; the first one is your primary key and the other one is a subkey. The subkey is used for encryption and the primary key is used for *GnuPG*'s other purpose: signing. You can sign other people's keys to affirm your trust and you can sign emails and other files to affirm their authenticity, which others may verify using your public key.

For your key to be useful, its public part needs to be accessible by anyone who may want to send encrypted documents to you, or anyone wishing to verify a document or key that you've signed. You could personally issue your public key to those that need it but it's better to publish it to a key server. Likewise, you can use a key server to get any public keys that you need. *Seahorse* will do this for you if you select its Remote > Sync And Publish Keys menu option.

You can browse key servers that have web interfaces – try **http://pool.sks-keyservers.net:11371** which, as its URL suggests, is actually a pool of many key servers. Some key servers may offer an option to upload keys directly through the website. Key servers propagate changes so, if you're looking for a key, you can use any server.

Seahorse will help you manage your own keys as well as any public keys that you've downloaded and, because it's a front-end to *GnuPG*, they're stored at **~/.gnupg** within your home directory. You can use them with any applications that support *GnuPG*, and your email client is an obvious candidate here.

Enigmail is a *GnuPG* add-on for *Mozilla Thunderbird* that you can install by selecting *Thunderbird's* Tools > Add-ons from its menu bar and searching for 'enigmail'.

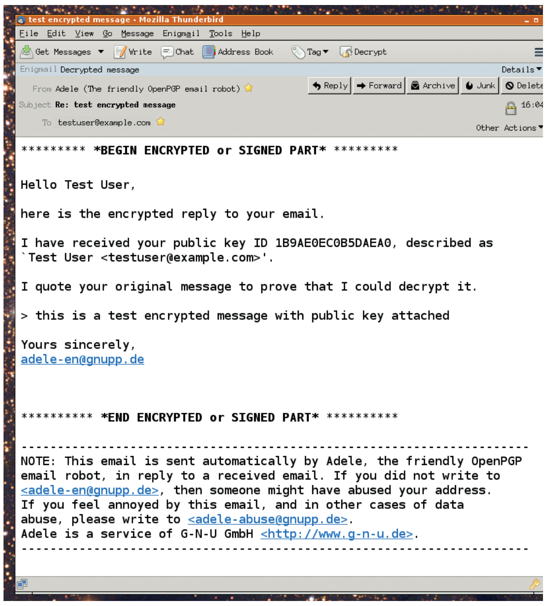
Encrypting email

It adds a new menu with options for encryption and signing, plus key management that provides similar functionality to *Seahorse* (try both and see which you prefer). It automatically decrypts or verifies inbound email using your keys once you've configured your email account to use it. Use *Thunderbird's* Edit > Account Settings to do that – you'll notice there is a new OpenPGP Security settings page there with a check box that enables *OpenPGP* support.

By default, *Enigmail* locates your key with your email address, and this should be sufficient unless you have multiple of either. You can, however, use the Account Settings screen to explicitly select a key if you need to. You can also configure default behaviour there, such as whether to sign or encrypt messages.

With the configuration done, you can write your first email. *Adele*" is "The friendly OpenPGP email robot" and it can help you test your setup. Just send a message to **adele-en@gnupp.de** and you'll get a reply that will highlight any issues with your ability to send encrypted messages. You can attach your public key to the message if you haven't published it to a key server.

"You may find it useful to have separate personal and private identities for use with GnuPGP."



Enigmail Decrypted message

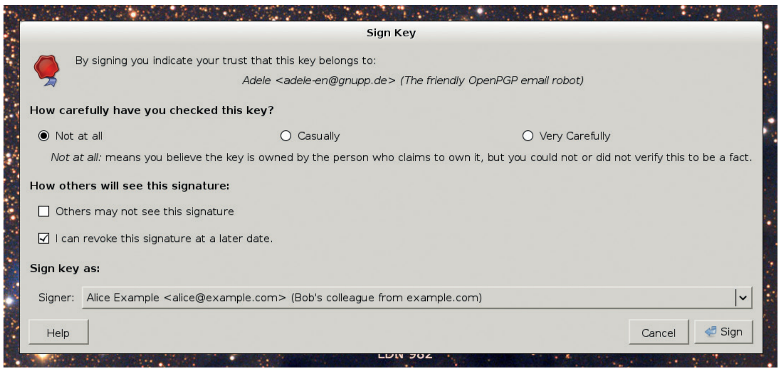
From Adele (The friendly OpenPGP email robot)
Subject **Re: test encrypted message**
To testuser@example.com
16:04

```
********* *BEGIN ENCRYPTED or SIGNED PART* *********

Hello Test User,

here is the encrypted reply to your email.

I have received your public key ID 1B9AE0EC0B5DAEA0, described as
`Test User <testuser@example.com>'.

I quote your original message to prove that I could decrypt it.

> this is a test encrypted message with public key attached

Yours sincerely,
adele-en@gnupp.de


********** *END ENCRYPTED or SIGNED PART* **********

------------------------------------------------------------------
NOTE: This email is sent automatically by Adele, the friendly OpenPGP
email robot, in reply to a received email. If you did not write to
<adele-en@gnupp.de>, then someone might have abused your address.
If you feel annoyed by this email, and in other cases of data
abuse, please write to <adele-abuse@gnupp.de>.
Adele is a service of G-N-U GmbH <http://www.g-n-u.de>.
------------------------------------------------------------------
```

Enigmail uses your secret key to open encrypted messages that you receive, like this from "Adele", which confirms that sending encrypted messages works too.

Web of trust

Key servers offer a convenient way to distribute your key and to obtain others' keys, but they don't prove authenticity in they way that an SSL certificate authority would. PGP uses a "Web of Trust" model instead, where trust is left to the users. This means that it's up to you to decide if a key obtained from a key server really does belong to who you think it does. You should use offline methods to establish a key's authenticity, in-person if possible. To help with this, *Seahorse*'s Key Properties screen displays a human-readable "Key Fingerprint" that can be easily compared. You can then indicate your trust by selecting a trust level from the drop-down menu on the same screen. If you want to share that trust publicly then you can use the "Trust" tab to sign the key. A popular way to build your web of trust is to attend a key-signing party, as we explain on page 113.



By signing you indicate your trust that this key belongs to:
Adele <adele-en@gnupp.de> (The friendly OpenPGP email robot)

How carefully have you checked this key?

[✓ Not at all] Casually Very Carefully

Not at all: means you believe the key is owned by the person who claims to own it, but you could not or did not verify this to be a fact.

How others will see this signature:

☐ Others may not see this signature
☑ I can revoke this signature at a later date.

Sign key as:

Signer: Alice Example <alice@example.com> (Bob's colleague from example.com)

You can sign a key that you trust, but be honest about how much you trust it.

LV PRO TIP
Seahorse can also be used to manage SSH keys and X.509 certificates.

LV PRO TIP
PGP keys are often referred to as a 'key pair' and the secret and public parts as a 'Private Key' and a 'Public Key'.

USE GNUPG ENCRYPTION FROM THE COMMAND LINE

JOHN LANE

Powering the desktop applications and plugins we've just looked at is **libgcrypt**, the *GnuPG* cryptographic library, and **gpg**, is its command line utility. If you've already installed a GUI tool then you'll already have **gpg**; if not, you can install it from your repository and confirm the version that you have:

```
$ sudo apt-get install gpg
$ gpg --version
gpg (GnuPG) 2.0.26
libgcrypt 1.6.2
```

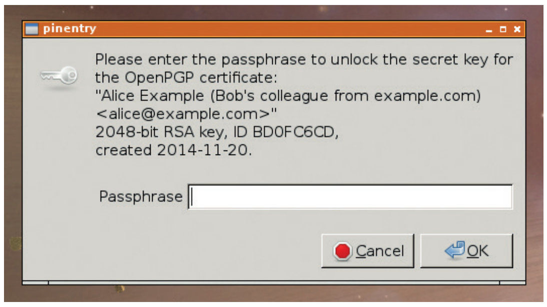
You'll need to create a key if you're starting out and, as **gpg** affords you more control than the graphical tools that we looked at previously, it's worth using it to create your key if you're comfortable on the command line. The simplest way to create a key is to use **gpg** interactively and follow its prompts:

```
$ gpg --gen-key
```

Alternatively, you can write your settings into a text file, say **alice.keygen**, like this:

```
Key-Type:     RSA
Key-Length:   4096
Key-Usage:    sign
Subkey-Type:  RSA
Subkey-Length: 3072
Subkey-Usage:  encrypt
Preferences:   SHA512 SHA384 SHA256 SHA224 AES256
AES192 AES CAST5 ZLIB BZIP2 ZIP Uncompressed
Name-Real:    Alice
Name-Comment:  Alice from example.com, the well-known
participant in examples.
Name-Email:   alice@example.com
Passphrase:   your secret is safe with me
```

This example creates a 4096-bit RSA primary signing key and a 3072-bit subkey for encryption. Longer keys are harder to crack, but they may not be compatible with all software, so as an example we've given our **subkey** a different size. The long list of **Preferences** contains our preferred hash, cipher and compression algorithms. If you omit the **Passphrase** entry from the file, you'll be prompted to enter it. Either way, your passphrase protects your key – choose it wisely and don't forget it!



Please enter the passphrase to unlock the secret key for the OpenPGP certificate:
"Alice Example (Bob's colleague from example.com)
<alice@example.com>"
2048-bit RSA key, ID BD0FC6CD,
created 2014-11-20.

There are pass phrase dialogs for *GTK 2*, *Qt 4*, and if you aren't running a graphical desktop, *Curses*.

Use the file as input to batch mode:

```
$ gpg --gen-key --verbose --batch alice.keygen
gpg: key 4A924D1C marked as ultimately trusted
```

The **--verbose** argument is there to show what's going on. The last line of output shows your key ID.

Your key is written into files in your **~/.gnupg** directory; the secret part in **secring.gpg** and the public part in **pubring.gpg**. Collectively these files are your Key Ring and, as you use **gpg**, you'll also store other people's public keys in there. Central to this key sharing is the Key Server, and you can tell **gpg** which one to use by adding an entry to its configuration file, **~/.gnupg/gpg.conf**:

```
keyserver hkp://pool.sks-keyservers.net
```

You should send your public key to the key server so others can find it and you can retrieve their public key using their email address or key ID:

```
$ gpg --send-key 4A924D1C
$ gpg --search-keys bob@example.com
$ gpg --recv-key 4E4A3DB3
```

Who do you trust?

The fact that anyone can upload to key servers means that you should challenge the authenticity of any key that you acquire. An individual who believes a key to be genuine can sign it to acknowledge that view, but it's still your decision to trust their opinion. Likewise, you can also sign keys that you trust. If you don't want to sign a key, you can still record your own trust level. This "Explicit Trust" is private to you, and you use the key editor to do it:

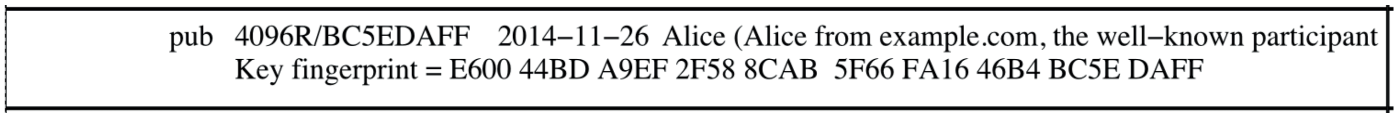
```
$ gpg --edit-key bob@example.com
gpg> trust
Please decide how far you trust this user to correctly verify
other users' keys
(by looking at passports, checking fingerprints from different
sources, etc.)
 1 = I don't know or won't say
 2 = I do NOT trust
```

GnuPG Modern

GnuPG version 2.1 was released on 6 November 2014 after a long beta. The 2.0.6 version series retains the "Stable" moniker, with this new release being called "Modern".

It has many new features, but most notable is that it manages private keys differently, delegating responsibility entirely to its **gpg-agent**. The secret keyring file **secring.gpg** is no longer used and **pubring.gpg** is now **pubring.kbx**, as the release notes explain. See **https://www.gnupg.org/faq/whats-new-in-2.1.html**.

It's unlikely to be in your distro yet, unless you use a rolling release like Arch Linux.


pub 4096R/BC5EDAFF 2014–11–26 Alice (Alice from example.com, the well–known participant
Key fingerprint = E600 44BD A9EF 2F58 8CAB 5F66 FA16 46B4 BC5E DAFF

gpg-key2ps quickly produces these key slips for you to print out, which are useful if you're getting your key signed.

```
3 = I trust marginally
4 = I trust fully
5 = I trust ultimately
m = back to the main menu
Your decision? 3
```

Using your key

Once you have a key, you can get on with the nitty-gritty details of its use, the main ones being signing and encryption. Let's look at signing first. There are two ways to do it; the first method produces a signed copy of an input file that is compressed and signed. It works like this:

```
$ gpg --sign mydocument.odt
```

which leaves the original file intact and creates a new binary file with a **.gpg** extension that is a compressed and signed copy. The signature can be verified:

```
$ gpg --verify mydocument.odt.gpg
gpg: Signature made Tue 25 Nov 2014 16:38:36 UTC using RSA
key ID 89572049
gpg: Good signature from "Alice (Alice from example.com, the
well-known participant in examples.) <alice@example.com>"
[ultimate]
```

You have to use **--decrypt** to recover the original document, even though it's only compressed inside the binary container file:

```
$ gpg --decrypt --output mydocument.odt mydocument.odt.gpg
```

The need to decrypt can be avoided with a "detached signature". This signs the document as before but only writes a signature file that you can pass along with the original file to anyone who needs it and they will be able to verify its authenticity using the signature file:

```
$ gpg --output mydocument.sig --detach-sig mydocument.odt
$ gpg --verify mydocument.sig mydocument.odt
gpg: Signature made Tue 25 Nov 2014 16:50:39 UTC using RSA
key ID 89572049
gpg: Good signature from "Alice (Alice from example.com, the
well-known participant in examples.) <alice@example.com>"
[ultimate]
```

Validating software packages

Many open source applications sign their packages so that their authenticity can be validated. As an example, we'll validate the latest *GnuPG* source:

```
$ wget ftp://ftp.gnupg.org/gcrypt/gnupg/gnupg-2.1.0.tar.bz2
$ wget ftp://ftp.gnupg.org/gcrypt/gnupg/gnupg-2.1.0.tar.bz2.sig
$ gpg --verify gnupg-2.1.0.tar.bz2.sig gnupg-2.1.0.tar.bz2
```

We downloaded the package and its signature but will also need the public keys belonging to those who signed the package, as **gpg** will report an error if it can't find them:

```
gpg: Signature made Wed 05 Nov 2014 15:44:27 UTC using RSA
key ID 4F25E3B6
gpg: Can't check signature: No public key
```

You can download the key (assign trust if you want to) and then verify again:

```
$ gpg --recv-keys 4F25E3B6
$ gpg --verify gnupg-2.1.0.tar.bz2.sig gnupg-2.1.0.tar.bz2
gpg: Good signature from ...
```

We've covered the basics of key management, document signing and encryption, but there's much more to *GnuPG* and *OpenPGP* in general. How far you go depends on how paranoid you are!

John Lane provides technical solutions to business problems. He has yet to find something that Linux can't solve.

The GPG agent and Pinentry

The **gpg-agent** is a daemon that manages secret keys for **gpg**, which starts it automatically on demand, making it something that you don't normally need to be concerned about. *Pinentry* is a helper tool used by **gpg** when it requires a user to enter a passphrase. The agent reads configuration from an optional file **~/.gnupg/gpg-agent.conf**, and one reason to use it is to specify the **pinentry** path if **gpg** has difficulty finding it, which an entry like this will achieve:

```
pinentry-program /usr/bin/pinentry-curses
```

You can start the agent manually:

```
$ eval $(gpg-agent --daemon)
```

To re-start it after changing configuration:

```
$ pkill --signal HUP gpg-agent
```

You can also ask it to reload itself like this:

```
$ echo RELOADAGENT | gpg-connect-agent
```

Should you want to terminate it:

```
$ pkill --signal TERM gpg-agent
```

The agent can also serve in place of the **ssh-agent** see **man gpg-agent** for more.

Key signing party

A key signing party is any event where individuals can meet to verify each others' identities before signing each others' keys to build their web of trust. You need to bring along some form of identification, a photo ID like a driving licence or passport, and a hard-copy of your "Key Fingerprint" which you can output from your secret key:

```
$ gpg --list-secret-keys --fingerprint
```

If you're on a Debian derivative, the **signing-party** package contains a utility that prints a page containing multiple copies of your fingerprint that you can cut into slips and use at a keysigining event:

```
$ gpg-key2ps -p a4 BC5EDAFF > keysheet.ps
```

At the event you'll verify each individual's identity and collect their email address and key fingerprint. After the event, you should obtain, verify and sign their keys and then send each person an encrypted copy of their key that you signed, which they should use to update their key and upload it to their keyserver. Here's how:

```
$ gpg --search-keys bob@example.com
$ gpg --fingerprint bob@example.com
$ gpg --sign-key bob@example.com
$ gpg -a --export bob@example.com | gpg --encrypt -r bob@
example.com -a | mail bob@example.com
```

It's up to you how much identity verification you do, but you're more likely to be trusted if you're known to take it seriously. If you're interested in key signing, websites like **biglumber.com** and **keysigning.org** have event listings.

/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!

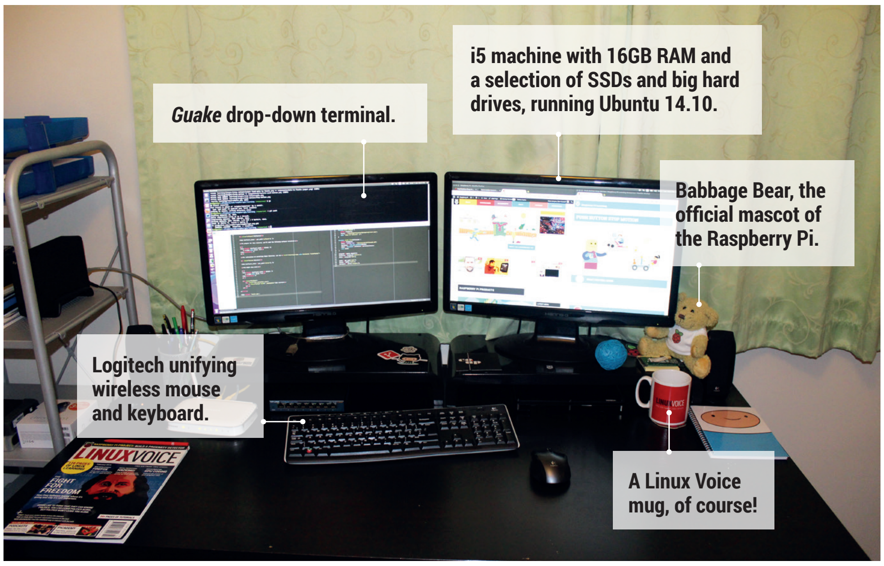
One of the very great things about Linux is the many splendid flavours available. Some of them cropped up at the recent CES show, where Intel unveiled its "PC on a stick", a dongle-sized quad Atom thing (running Ubuntu) for next to no money. I am still thinking about what I would use it for. Maybe I need more than one.

Of course, Linux featured heavily in the TV section too. Not just the endless Android powered devices (it still runs a sort-of-Linux kernel after all), but the latest Samsung Smart TV (which uses the Tizen OS it originally shipped with some phones), and one or two featuring Roku (the operating system that's also available in some media boxes of the same name). It is a bit of a wonder that nobody has tried to put *MythTV* into an actual TV.

Supreme Leader's distro

There seems to be a Linux for everything. Unless you are in North Korea, in which case there is only one Linux deemed suitable for the Sony-hating masses – Red Star Linux, which in its latest incarnation seems to have borrowed quite a lot of look and feel from somewhere else - surely they don't take field-trips to Cupertino? Anyhow, it is testament to what one can do with KDE (see for yourself – **http://goo.gl/DxOPSu** courtesey of Wikimedia), but I think you may have to travel there to actually buy a copy.

Of course, this celebration of distros can't miss out on one of my favourites (or should that be three of my favourites, since the development team decided to split it up into bits?): Fedora. While the new versions are all very exciting, I did notice some "end of life" messages for an older version.Yes, release 19, codenamed "Schrödinger's cat", is officially dead.


Guake drop-down terminal.
i5 machine with 16GB RAM and a selection of SSDs and big hard drives, running Ubuntu 14.10.
Babbage Bear, the official mascot of the Raspberry Pi.
Logitech unifying wireless mouse and keyboard.
A Linux Voice mug, of course!

My Linux Setup Ben Nuttall

Education and outreach human at the Raspberry Pi Foundation.

Q What version of Linux are you using at the moment?

A I have a desktop at home and a laptop for work – both running the latest Ubuntu release. I also have a little HP ARM Chromebook, which is great for when I'm travelling. Chrome OS is nice for browsing (and even SSH), but I can switch to an Ubuntu container with *crouton* if I need to hack at something.

Q What was the first Linux setup you ever used?

A I ran my computer solely from an Ubuntu live disc for a good few weeks once when my hard drive was playing up, back in around 2007. A couple of years later I started using it more, and by then I'd met a bunch of people who used it day-to-day so I had more support, and that's really important.

Q What Free Software/open source can't you live without?

A One of the first things I install on a new machine is *Guake* – a simple drop-down terminal. It suits the way I work really well, as I like to have focus on some window, like a web browser, and have quick access to a terminal or three, without juggling different windows.

Q What do other people love, but you can get on without?

A A lot of people use the GitHub desktop GUI, but I guess that's Windows and Mac only anyway. I'm perfectly happy using *git* from the command line.

Q Is there one single piece of proprietary software you wish were open source?

A I'm a fan of the *Sublime Text* editor, and although you can download it for free it's not open source. I'm starting to use GitHub's open source editor *Atom* more, which is still in beta but it's more feature heavy and really nice to use. Having said that, I probably use *Vim* more than anything – but graphical editors definitely have a place.

If you support open source software, The Open Source Initiative needs your support.

I love open source & want to **help**

I support open source software and the OSI

I support the free & open source movement

Open is the better way

Open Source is an idea that has changed the world for the better The OSI helps keep that idea alive

Because I believe in open source

Wow. Been seeing a ton of tweets from @OpenSourceOrg the last few weeks Glad to see it's getting some energy back

I deeply believe in the OSI mission

OSI represents hope

I want to help the OSI because it is time to gave back after all the support they have provided

I am a firm believer in what the OSI, and other open source communities, are doing to keep software transparent

The OSI is redefining business

I have enjoyed the benefits of open source for many years and, now an open source contributor, would like to help support the open source infrastructure

Open source shapes our future

The open source movement needs this kind of **support**

The future depends on new ideas,alternatives and paradigms

talented people to collaborate on finding solutions

The OSI promotes and protects **open source**

Open source software also spurs innovation and collaboration allowing different people to experiment with their own ideas about how the software could work

Open Source Initiative provides a powerful **community** from across the globe

The **Open Source Initiative** (OSI) was founded on February 3rd, 1998, or "2/3/98" and in 2012, the OSI transitioned to a membership-driven organization, offering both Affiliate Memberships to non-profit open source projects and Individual Memberships to open source software developers, enthusiasts, supporters and end-users.

To celebrate both our founding and promote our commitment to a representative, community driven organization, the **Open Source Initiative** is launching our first ever

Individual Membership Drive

on our anniversary — February 3, 2015 — with a goal of 2,398 new Individual Members.

We invite you to join our Affiliate Membership, including Debian, Drupal, Eclipse, FreeBSD, Linux Foundation, Mozilla, Python, Wikimedia, Wordpress and many more, as well as our corporate sponsors, Google, HP, IBM, Linux Journal and Nginx to name a few, and of course the many other Individual Members from around the globe, who have already committed to promoting and protecting open source software development, projects and the communities that ensure its continued adoption and success.

Please join now at:

www.opensource.org/join

The **Open Source Initiative** is a member-driven non-profit corporation with global scope formed to educate about and advocate for the benefits of open source development and to build bridges among different constituencies throughout the open source community. Open source is a development method for software that harnesses the power of distributed peer review and transparency of process. The promise of open source is higher quality, greater reliability, more flexibility, lower cost, and an end to predatory vendor lock-in. As a member of the Open Source Initiative you'll help develop and drive the direction of open source software, ensure the integrity of the Open Source Definition for the good of the community and protect open source licensing, creating a nexus of trust around which developers, users, corporations and governments can organize open source cooperation.