Full Circle

THE INDEPENDENT MAGAZINE FOR THE UBUNTU LINUX COMMUNITY

ISSUE #134 - June 2018

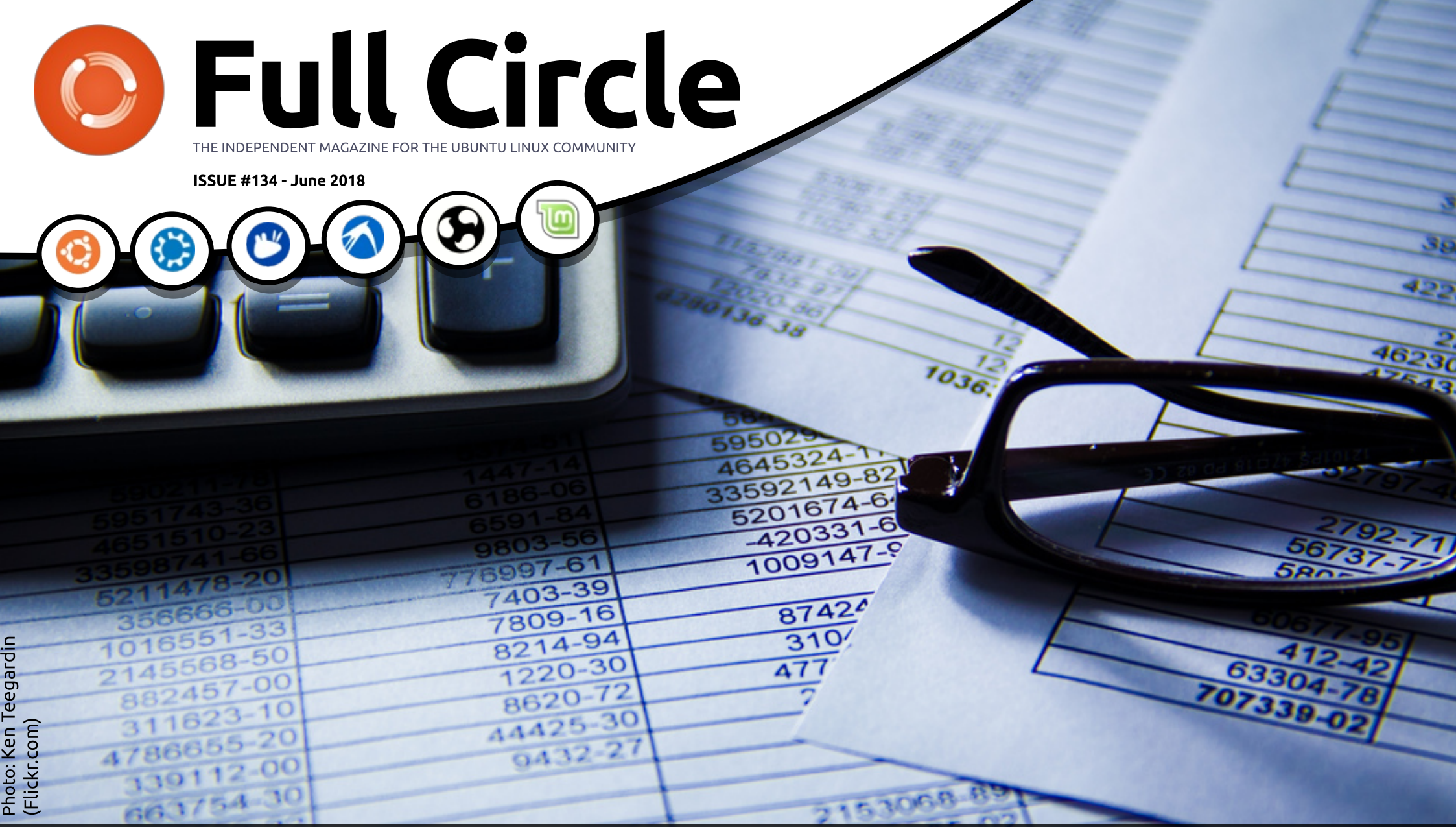


Photo: Ken Teegardin (Flickr.com)

STATISTICAL ANALYSIS

USING R AND GNUPLOT

Full Circle

THE INDEPENDENT MAGAZINE FOR THE UBUNTU LINUX COMMUNITY

EDITORIAL

Welcome to the latest issue of Full Circle.

As ever, we have more Python, Freeplane, Inkscape and the return of Great Cow Basic for you this month. Due to personal reasons Miguel and his Ubuntu Touch series is still out of action.

Unfortunately, due to medical reasons, we have to say goodbye to Gord. His Q&A section is probably one of the more popular parts of Full Circle, and I'm sad to see him go. He's been writing for FCM for a number of years. He'll still be in the background though. Fixing all my writing mistakes. If anyone is interested in taking over the Q&A section of the magazine, feel free to drop me an email: ronnie@fullcirclemagazine.org. Otherwise, it too will be retired (like Linux Labs).

Having several sections being retired means the magazine will become less and less each month. Now is the time if you want to help out and write a monthly column. It can be on pretty much any topic. All I ask is that it has some relation to Ubuntu, or (at least) Linux. Email your ideas to me (Ronnie) at the email address above (and below).

The response to the survey has been well down on previous years, so it's still up for you to give us your comments: **http://bit.ly/fcm2018**.

All the best, and keep in touch!
Ronnie
ronnie@fullcirclemagazine.org

This magazine was created using :

Find Full Circle on:

goo.gl/FRTMl
facebook.com/fullcirclemagazine
twitter.com/#!/fullcirclemag
http://issuu.com/fullcirclemagazine
http://www.magzter.com/publishers/Full-Circle

Weekly News:

http://fullcirclemagazine.org/feed/podcast
http://www.stitcher.com/s?fid=85347&refid=stpr
http://tunein.com/radio/Full-Circle-Weekly-News-p855064/

Full Circle 2018 Survey

It's that time of the year again where we ask what you think of FCM, Ubuntu, and Linux.

Some questions are a requirement, some you can skip over if not applicable.

Your answers will help shape Full Circle, so please use your constructive criticism. If you don't tell us what you think, or what we're doing wrong, then we won't know.

Survey URL:
http://bit.ly/fcm2018

Dell Precision 'Developer Edition' mobile workstations run Ubuntu Linux and are RHEL certified

While Dell is mostly known as a Windows PC manufacturer, the company is also a big proponent of Linux. Its "Developer Edition" models can be configured with Ubuntu, for instance. Of course, despite this branding, non-developers can buy them too. The XPS 13 "Developer Edition" in particular is a svelte machine that should make many home Linux users very happy.

In addition to home users, Dell manufactures solid business-class mobile workstations, and the company recently announced four such Linux-powered models. These Precision "Developer Edition" laptops run Ubuntu and are RHEL certified. One of these notebooks, the Precision 3530, is available today, while the other three will be available soon.

"Today I'm proud to announce the new the Linux-based Dell Precision Mobile workstation line: the 3530, 5530, 7530 and the 7730. These systems, which represent the fourth generation of the Precision developer editions, come preloaded with Ubuntu and have been RHEL certified. These new thinner, lighter, premium-built Precision mobile workstations feature the latest Intel Core and Xeon processors, blazing-fast memory and professional graphics," says Barton George, Dell.

George further says, "As mentioned above, all four developer editions will be certified for RHEL 7.5 and the needed drivers will be included in the distro. That being said, the AMD and NVIDIA drivers that come with 7.5 are inbox drivers. We will be posting drivers for both graphic cards that will include professional features not included in the inbox drivers."

Source:
https://betanews.com/2018/05/27/dell-precision-developer-ubuntu-linux-rhel/

You know that silly fear about Alexa recording everything and leaking it online? It just happened

It's time to break out your "Alexa, I Told You So" banners – because a Portland, Oregon, couple received a phone call from one of the husband's employees earlier this month, telling them she had just received a recording of them talking privately in their home.

"Unplug your Alexa devices right now," the staffer told the couple, who did not wish to be fully identified, "you're being hacked."

At first the couple thought it might be a hoax call. However, the employee – over a hundred miles away in Seattle – confirmed the leak by revealing the pair had just been talking about their hardwood floors.

The recording had been sent from the couple's Alexa-powered Amazon Echo to the employee's phone, who is in the husband's contacts list, and she forwarded the audio to the wife, Danielle, who was amazed to hear herself talking about their floors. Suffice to say, this episode was unexpected. The couple had not instructed Alexa to spill a copy of their conversation to someone else.

"I felt invaded," Danielle told KIRO-TV. "A total privacy invasion. Immediately I said, 'I'm never plugging that device in again, because I can't trust it.'"

The couple then went around their home unplugging all their Amazon Alexa gadgets – they had them all over the place to manage various smart home devices, including a thermostat and security system – and then called the web giant to complain about the snooping tech.

According to Danielle, Amazon confirmed that it was the voice-activated digital assistant that had recorded and sent the file to a virtual stranger, and apologized profusely, but gave no explanation for how it may have happened.

Source: https://www.theregister.co.uk/2018/05/24/alexa_recording_couple/

Fedora 26 Linux to Reach End of Life on June 1, 2018, Upgrade to Fedora 28 Now

Released eleven months ago, on July 11, 2017, the Fedora 26 Linux operating system brought the GNOME 3.24 desktop environment, the DNF 2.5 package manager, a brand-new partitioning tool in the Anaconda installer for expert setups, as well as the Python Classroom Lab spin designed for educators and students.

Almost 10,000 updated packages were published for the Fedora 26 Linux release during its lifetime, which will end this Friday, June 1, 2018. After this date, Fedora 26 users will no longer receive security patches and bug fixes, nor enhancement updates and new packages to its software repositories.

Those still running the Fedora 26 Linux operating system on their personal computers have only three days to update to the most recent release, Fedora 28, though they can also update to Fedora 27, which will be supported for at least half a year, until December 2018 or January 2019, depending on the release of Fedora 29.

It is recommended that you update to Fedora 28 directly as it's more advanced than Fedora 27, and because it will be supported until summer 2019. Fedora 28 ships with the latest GNOME 3.28 desktop environment, Linux kernel 4.16, and other up-to-date components and the most recent GNU/Linux technologies.

Source: https://news.softpedia.com/news/fedora-26-linux-to-reach-end-of-life-on-june-1-2018-upgrade-to-fedora-28-now-521325.shtml

GNU Emacs 26.1 Released

GNU Emacs is a text editor that contains a variety of functions and can be equipped with any extensions through its programming interface. It includes enhancements for such diverse things as compiling and debugging programs, handling e-mail, and supporting the X Window System. For fun and to show what can be done with Emacs Lisp everything, the editor also includes a "psychotherapist" and various games. Emacs also includes a tutorial and documentation. Furthermore, Emacs is universally applicable and also available on many other platforms.

As Nicolas Petton announced, a new generation of applications is available over a year and a half after the last major release. The version 26.1 comes with many innovations and improvements. From the user's point of view, one of the most important new features is the revision of the presentation of content, which among other things should prevent flickering. In addition, the application has been given a limited form of parallelism with Lisp threads. Flymake has been completely redesigned and

optional line numbers can be displayed in the buffer.

Other recent additions to GNU Emacs 26.1 include enhancements to TRAMP, which now includes a new Google Drive connection method, a one-line horizontal scrolling mode, and support for 24-bit color on text terminals. Furthermore, Emacs will be delivered with a matching systemd-unit-file.

Source: http://www.pro-linux.de/news/1/25935/gnu-emacs-261-freigegeben.html

GNOME transitions to GitLab

GNOME is one of the most important open source projects on the planet. Even if you use an alternative desktop environment, such as KDE or Xfce, you cannot deny that GNOME's contributions have largely shaped the path of Linux on the desktop for the better. Don't forget, GNOME is much more than just a DE -- many of its apps are essential. That's probably why someone (or something) recently pledged to donate $1 million to it -- GNOME's existence and success is a must for the Linux community.

Today, The GNOME Foundation makes a very big announcement -- the project has fully transitioned to GitLab. In other words, yes, the GNOME Project is growing up a bit and becoming better organized. This move to the GitLab Git-repository manager is a breath of fresh air, quite frankly, and should really improve collaboration between GNOME Project contributors.

This doesn't mean a lot for end users -- at least not directly or initially. Developers and contributors to GNOME, however, should be more productive as a result. After all, you can have all the talent in the world, but without the right tools, they won't reach their top potential. Eventually, end users should benefit from the move to GitLab, as the GNOME Project should progress at a faster pace and yield better results -- that seems to be the goal, at least.

Source: https://betanews.com/2018/06/01/gnome-transitions-to-gitlab/

Firefox And Chrome Bug Leaked Facebook Profile Details For Almost A Year

A side-channel vulnerability existed in the implement of the CSS3 feature called “mix-blend-mode.” It allowed an attacker to de-anonymize a Facebook user running Google Chrome or Mozilla Firefox by making them visit a specially crafted website.

The flaw, now fixed, was discovered last year by the researcher duo Dario Weißer and Ruslan Habalov, and separately by another researcher named Max May.

The proof-of-concept created by the researchers enabled them to harvest data like the profile picture, username, and ‘like’ status of unsuspecting visitors, the researchers said in their blog post. All of this could be done in the background when a user visits a malicious site.

The visual data leak could happen on websites using iFrames that link to Facebook in the form of social plugins and login buttons. Because of a security feature called same origin policy, websites can’t access the content of iframes directly. The researchers can extract information by creating an overlay on the cross-origin iFrame to interact with the underlying pixels.

While the flaw has been patched for good, the researchers warn that the advanced graphics capabilities added to HTML and CSS could open doors for more attacks like these.

Source: https://fossbytes.com/firefox-chrome-side-channel-attack-leaked-facebook-profile-details/

Linus Torvalds decides world isn’t ready for Linux 5.0

Linus Torvalds has decided the world’s not ready for version 5.0 of the Linux Kernel, so he’s given us version 4.17 instead.

Torvalds toyed with the idea of calling this release 5.0, because it

passed the six million git objects mark. But he also said version numbers are meaningless and he might not call it 5.0.

The latter has now come to pass: in his regular Sunday afternoon (Pacific Time) state-of-the-kernel update, Torvalds announced that “I really didn't get the feeling that another week would help the release in any way, so here we are, with 4.17 released.”

“No, I didn't call it 5.0, even though all the git object count numerology was in place for that,” he added. “ It will happen in the not _too_ distant future, and I'm told all the release scripts on kernel.org are ready for it, but I didn't feel there was any real reason for it.”

“I suspect that around 4.20 - which is I run out of fingers and toes to keep track of minor releases, and thus start getting mightily confused - I'll switch over. That was what happened for 4.0, after all.”

Source: https://www.theregister.co.uk/2018/06/04/linux_4_17_released/

Ubuntu 18.04-based Linux Mint 19 'Tara' Beta is here with Cinnamon, MATE, and Xfce

Ah, Linux Mint. This operating system has its detractors, but for the most part, it is beloved by both Linux beginners and experts alike. True, most of the praise is due to its excellent Ubuntu base, but the Mint team understands what many other distribution maintainers don't -- the overall experience matters. When you install Mint, you are in for a polished treat -- it is clear that the developers truly care about the end users.

Today, Linux Mint 19 "Tara" Beta finally sees release. Three desktop environments are available -- Cinnamon (3.8), MATE (1.20), and Xfce (4.12). All of these DEs are excellent, but the shining star is Cinnamon. Tara is significant as it is based on the newest Ubuntu 18.04 and will receive updates until the year 2023. The included Linux kernel is version 4.15 and not 4.17 -- understandable, as it was only just released.

In Linux Mint 19, the star of the show is Timeshift. Although it was introduced in Linux Mint 18.3 and backported to all Linux Mint releases, it is now at the center of Linux Mint's update strategy and communication. Thanks to Timeshift you can go back in time and restore your computer to the last functional system snapshot. If anything breaks, you can go back to the previous snapshot and it's as if the problem never happened. This greatly simplifies the maintenance of your computer, since you no longer need to worry about potential regressions. In the eventuality of a critical regression, you can restore a snapshot (thus canceling the effects of the regression) and you still have the ability to apply updates selectively (as you did in previous releases).

Source: https://betanews.com/2018/06/04/linux-mint-19-tara-beta/

4MLinux 25.0 Distro Hits Stable with Full Zstd Support, Linux Kernel 4.14.39 LTS

With the 4MLinux 23 series reaching end of life on June 3, 2018, the 4MLinux 25.0 operating system has been promoted to the stable channel in the same day, allowing users to upgrade their installations as soon as possible. 4MLinux 25.0 has been in development for the past six months and includes numerous improvements, updated components, and new features.

Powered by the long-term supported Linux 4.14.39 kernel, 4MLinux 25.0 is the first release of the Linux-based operating system to ship with full support for Facebook's Zstandard (Zstd) data compression algorithm. It also improves handling of CA certificates so you won't have to accept them manually and finally lets users disable the login screen.

The software collection of 4MLinux not only was updated with some of the most recent versions, but it suffered some interesting changes, such as the addition of MPV as default media player with the GNOME MPV front-

end. Other media players like SMPlayer, MPlayer, VLC, and Xine can be installed through downloadable extensions.

Source:
https://news.softpedia.com/news/4mlinux-25-0-distro-hits-stable-with-full-zstd-support-linux-kernel-4-14-39-lts-521421.shtml

Intel Claims Its Low Power Display Tech Can Cut Battery Use By 50%

Among many announcements made at Computex 2018, one from Intel could give a sense of satisfaction to the users who concerned about the battery life of the computers. Because, the display is the component that gobbles battery the most.

Intel claims its new tech called Low Power Display can cut the LCD power consumption by half. To bring it to fruition, Intel co-engineered it with Sharp and Innolux. The two manufacturers are developing 1 Watt display panels using the technology.

On paper, this is 3 hours more than what's said in the case of the recently announced Snapdragon 850 SoC designed for Arm-based Windows 10 PCs.

Intel demoed its tech on a Dell XPS 13 featuring a low power display. It goes without saying that the advantage would be available in upcoming devices. Also, it would require the device to be running Intel graphics.

These power efficient displays could be a part of Intel's efforts to retain its market. It may not be the very next day, but, Arm-powered PCs could be a potential threat to Intel's decades of dominance. The company has already lost the battle in the smartphone segment.

Source:
https://fossbytes.com/intel-low-power-display-tech-battery-life-half/

Purism's Librem 5 Privacy, Security-Focused Linux Phone Arrives in January 2019

With the promise to be world's first community-owned smartphone ecosystem, the security and privacy-focused Librem 5 managed to raise nearly $2.5 million during its crowdfunding campaign, and it's now on track to be released worldwide in January 2019 as Purism announces today major strides in manufacturing and development of the mobile device.

It would appear that Purism already managed to finalize the specifications for Librem 5's hardware platform, placing an order to its manufacturer for developer kits that have the same base specifications as the final design of the mobile phone. Also, they managed to finalize the case design and UI shell of Librem 5, as well as the Contacts and Calls apps.

Promising to give users the freedom they need and complete control over their mobile phones, Librem 5 will bear features not seen in mainstream smartphones, including encrypted calling, native VoIP (Voice over IP) capabilities, end-to-end encrypted storage, SMS, and email, preconfigured VPN services, support for a wide-range of Linux-based operating systems, and a slot for an encryption smartcard.

The device will also support GSM, UMTS, 3G, 4G, and LTE mobile networks, Wi-Fi and Bluetooth wireless connections, and have hardware kill switches for the camera and microphone to prevent privacy breaches. The Librem 5's source code will be publicly available after launch to allow users to modify it as they see fit or to develop additional apps and services for the mobile phone.

Source:
https://news.softpedia.com/news/purism-s-librem-5-privacy-security-focused-linux-phone-arrives-in-january-2019-521437.shtml

Canonical Announces Ubuntu for Amazon's Elastic Container Service for Kubernetes

Officially launched on Tuesday, Amazon Elastic Container Service for Kubernetes (Amazon EKS) is engineered to deliver Kubernetes, the open-source and

production-grade container orchestration tool as a managed service on the AWS (Amazon Web Services) cloud computing services. As Ubuntu is the most widely used container host operating system, especially for Kubernetes deployments, it can now be used to host containers in Amazon's EKS.

Designed for container portability, Ubuntu on EKS promises a set of great optimizations over Ubuntu on AWS. Among these, we can mention up to 30 percent faster kernel boot speeds, i3.metal support, better i3 instance class support with NVM Express (NVMe) storage disks for extreme IO, Elastic Network Adapter (ENA) with support for up to 25 Gbps network interfaces, as well as continuous security updates and image maintenance to address critical flaws.

In the private infrastructure, the Ubuntu on EKS container hosts offer the same runtime dynamics as those of other Ubuntu-based Kubernetes deployments. It consists of worker nodes created using Canonical's brand-new minimal Ubuntu base image designed to dramatically shrink both the security cross-section and the image size for Ubuntu on AWS. This makes it easier to customize Ubuntu EKS container host nodes to match your enterprise needs.

Source: https://news.softpedia.com/news/canonical-announces-ubuntu-for-amazon-s-elastic-container-service-for-kubernetes-521446.shtml

Support for Linux 3.2 and 4.1 finished

With a total of 151 modified files, 1139 insertions, and 583 deletions, Linux 3.2.102 was released as probably the last planned version of the kernel released in 2012. As Ben Hutchings writes on the kernel mailing list, "in all likelihood," official support for the long-term kernel ended. Of course you can not be sure of that, because unexpected gaps or other circumstances could lead to Linux 3.2 still undergoing another update. However, users should not rely on it and update their systems as soon as possible.

Also users of Linux 4.1 are to subject their systems to an update. As with Linux 3.2, the release of a new version - in this case Linux 4.1.52 at the end of May - also marks the cessation of support. This can be checked, for example, on "Active kernel releases", where only four supported versions can still be found.

In addition to the kernel 3.16 maintained by Ben Hutchings, which will be updated until at least April 2020, users will also be able to access the Greg Kroah-Hartman supported kernel versions 4.4, 4.9 and 4.14. With version 4.9, however, it should be noted that it will only be supplied with updates for a good half a year. Alternatively, LTS kernel variants of the distributors can be imported or adapted. These are maintained directly by the manufacturers and are subject to other support periods. Details and support periods can be found in the corresponding products.

Source: https://www.pro-linux.de/news/1/25965/unterst%C3%BCtzung-f%C3%BCr-linux-32-und-41-beendet.html

GitLab Ultimate and GitLab Gold for open source projects for free

Microsoft's announced acquisition of Github, due to be completed by the end of the year, has triggered a lot of speculation and reaction. The most plausible theory of why Microsoft is spending $ 7.5 billion on the platform is the lock-in effect created by the proprietary code that powers Github. Projects that use Github can not switch to another platform without leaving much of their data behind, including all information and discussions about errors and user preferences. Because there is no way for Github to export this data. Already years ago, the FSF had warned against dependence on a provider and pointed out that Github does not even begin to fulfill its criteria for "ethical" project hosting sites.

In response to the increased interest, GitLab has now announced that its fee-based project hosting packages GitLab Ultimate and GitLab Gold are now free for open source projects and educational institutions. GitLab

Ultimate is a package that runs on customer-owned servers while GitLab Gold runs on GitLab servers. Entitled are all non-profit educational institutions and all open source projects with OSI recognized free licenses. Again, the exclusion of projects that achieve profit directly with the software applies. Compared to the freely available community edition of GitLab both packages contain additional functions, only support is excluded. If support is needed, it can be purchased for five percent of the regular price, $ 4.95 per user per month.

Source: http://www.pro-linux.de/news/1/25968/gitlab-ultimate-und-gitlab-gold-f%C3%BCr-open-source-projekte-kostenlos.html

Full Circle 2018 Survey

It's that time of the year again where we ask what you think of FCM, Ubuntu, and Linux.

Some questions are a requirement, some you can skip over if not applicable.

Your answers will help shape Full Circle, so please use your constructive criticism. If you don't tell us what you think, or what we're doing wrong, then we won't know.

Survey URL:
http://bit.ly/fcm2018

This past month has been spent streamlining my development process, in order to more quickly create prototypes and advance onward to finalized styles. Part of this included watching a video from Adam Wathan (one of the creators of Tailwind CSS). In this video on YouTube, he used Tailwind to quickly create the new https://refactoringui.com/ design. It inspired me to give it a shot, and this article will offer some examples, and my insights.

What is Tailwind CSS?

Tailwind CSS is a framework that offers utility classes (similar to some functional CSS frameworks). The utility classes essentially store typical CSS settings in a class - such as the class "shadow" instead of defining a shadow by hand. However, it also offers a function "@apply" to apply a combination of classes within CSS, which can cut down on the repetition that appears in HTML. This is a feature I did not know about, until using it. You can also customize Tailwind massively by using their tailwind.js file.

Why?

Prototyping typically requires a bit of back and forth on different CSS settings, and the defaults shipped with Tailwind (and the ability to customize it) are perfectly sane. So instead of jumping between an HTML file and a CSS file (and possibly waiting for your preprocessor to compile), you can instead stick to the HTML view and simply attach classes as you need to the various elements. I find this approach is much faster than jumping between HTML and CSS, and also gives you a starting point for appearance (as opposed to creating your own default shadows, borders, backgrounds, etc.).

Then, once you've prototyped the system, you can reduce repetition of classes by creating your own custom class names and styles, and using the exact settings from the prototype thanks to @apply. For examples please see the codepen and following sections.

Preamble

In order to follow along with this article (and to have easy access to all the files I used), I created the following two codepens:

- https://codepen.io/lswest/full/XYKVVw/ - This is Part 1, where we use only the utility classes Tailwind offers.
- https://codepen.io/lswest/full/LrZeaZ/ - This is Part 2, where we extract repeated combinations of classes, and turn them into our custom styles.

Note: To view the HTML in the above examples, you'll need to change the view to "editor view", or view the source of the page.

You'll also need to have nodejs and npm installed.

I would also recommend (not required) using something like Atom or Visual Studio Code set up with Emmet (automatically available in Code).

Getting Set Up

- Create a folder you want to work on this project.
- Visit one of the codepens listed above, and copy the contents of the package.json file into your own package.json file
- Run npm install to install all the relevant packages.
- Run ./node_modules/.bin/tailwind init in order to generate your own tailwind.js file.
- Visit the first codepen above, and copy all the files into the relevant locations (listed in the pen).
- Once the files are in the correct locations, you'll want to run either npm run or npm run watch. This will run webpack and generate the ./dist/styles.css file.

Note: If you're installing these packages by hand into a custom package.json file, at the time of writing you will need to install extract-text-webpack-plugin@next (the beta version) for it to be compatible with Webpack 4.

Note #2: While Tailwind is available via CDN, using it removes some of the flexibility, as you have no control over the tailwind.js file. If you just want to try it out, feel free to use the CDN. If, however, you're looking to customize it, definitely use some form of precompiler as explained in the documentation.

Explaining the HTML

In both codepens, the HTML is pretty similar. I used pre tags (for preformatted) to ensure the formatting of the files worked alright, and simply included a small header to indicate what project this is. The majority of the boilerplate was generated using the emmet html:5 abbreviation, and then the viewport was added (meta:vp).

I originally threw all the content into the file using blank HTML elements. This means the header originally looked like this:

```
<header>

      <h1>Command &amp;
Conquer: Tailwind CSS</h1>

</header>
```

None of the HTML is too crazy, and so I will leave the explanation here. What follows now is the prototyping phase, where we utilized the Tailwind CSS utility classes.

Adding Utility Classes

Anyone who has copied the HTML from my first codepen link, will notice that there are a lot of styles added. I'll run through some of the classes, but they are generally self-explanatory.

```
<body class="font-sans
leading-tight antialiased h-
screen">

    <header class="bg-
orange">

        <h1 class="text-
center text-grey-lighter py-
8">Command &amp; Conquer:
Tailwind CSS</h1>

    </header>

    <main class="container
mx-auto mt-16">
```

This is the first 5 lines of the pen. The first class on the body tag are telling the browser to use the font-sans class, which is where Tailwind defines a sans font family. The other classes are simply informing the browser to antialias the fonts, and apply tight leading (a typography term referring to the vertical spacing between lines of text).

The header gets the default orange as a background, and the h1 is centered, made a light grey color, and receives padding on the y-axis (top and bottom) of "8". This "8" is defined in the tailwind.js file, and by default is 2rem (for a total of 4 rem, split between the top and the bottom). These numbers (by default) are the rem value * 4.

The main element is a container (where the width is limited based on screen size), and the mx-auto class adds auto margins on the left/right (x-axis). This essentially centers the container in the middle of the screen. The mt-16 adds padding to the top. By default tailwind.js only defines up to mt-8. In order to add mt-16 like I'm using, you'll need to add the following line:

```
'16': '4rem',
```

This line should go at the end of the margin array (simply search the file for "margin:"). Once webpack has recompiled the styles, you'll have access to mt-16.

Customizing Tailwind

As you saw at the end of the last section, changing settings and defaults for Tailwind is extremely simple. For prototyping something, you may not want to remove any defaults, and instead simply add to it (as we did with the margins above). Once you're ready to go into production (or if you're following a style guide), then I would highly recommend removing all unnecessary colors, and adding/editing them to match the given styles. This will reduce the resulting CSS file immensely (as each color is used for backgrounds, text and borders).

Naturally, as you continue your project, you may need to adjust screen sizes, margins, shadows, etc. - which you are free to do at any time. I simply recommend starting with the colors, as these are fundamental, and can have a large impact on the number of classes.

The Design

The design of my codepen is

pretty simple - I chose an orange color because that is prevalent in Full Circle Magazine, and otherwise just stuck to grey and black for legibility.

I make ample use of the inline/block, overflow, rounded, and the bg/text classes. The rounded class simply rounds the corners of the pre tags, and the inline/block classes add display:inline or display:block to the elements. The overflow-x-auto is used on the pre tags to avoid text running outside of the grey background on smaller devices, and the webpack.config.js file gets overflow-y-auto and a height (64 = 16rem). This is due to the length of the file - I wanted to keep the HTML file from getting too long, and the webpack file will remain unchanged throughout this article.

This prototype took me less than 10 minutes in total (to the state of codepen part 1). While it isn't the most complex, it's also visually attractive enough to leave it be at that point. I would have previously either had much less polished CSS, or simply not be finished at this point.

The Repetition

Instead of removing all the classes from Tailwind and doubling my efforts by giving every element specific styles or a specific class, I instead focused only on those elements I repeated. This is essentially 3 styles to define:

- code blocks (block display, grey background, vertical padding of 2rem, rounded corners, and overflow-x: auto;)
- pre - in the process steps, I repeat "inline" classes a lot. As such, I made the display: inline; the default one, and simply added class="block" to the file tree, and integrated it into the code class.
- h3 - These should be orange. While it is a single class that we are replacing, imagine if you decide you want to use a red color? You wouldn't want to define "orange" as a red color, nor would you want to adjust a bunch of styles in various HTML locations. Instead, I use an element selector, and simply move the text-orange setting into CSS. This way any changes occur in one location.

Using @apply

Using @apply is extremely simple. After the imports in the src/styles.css file, you simply define your class as normal, and then write @apply <classes from HTML>. Do keep in mind that you need to add a period in front of the class names if you copy them directly from the HTML, as you are writing CSS selectors. Example:

```
.code {

      @apply .block .bg-grey
.py-4 .rounded .overflow-x-
auto

}
```

When webpack recompiles the file, this @apply will be replaced with the actual styles from the classes listed. For the other two styles, check the second codepen for the contents of my src/styles.css file.

Extra Features

Tailwind also offers special classes to allow you to define styles by screen size, or to style a hover state. For example:

```
<button class="w-full md:w-
1/3 hover:bg-orange">

Button

</button>
```

The classes in that example defines the button's width to be 100% by default. On medium devices (breakpoint is defined in tailwind.js) the width is then ⅓, and on hover the background should become orange.

If you want to use these classes in @apply, you will need to remove the definition before the colon. So for the hover state:

```
button:hover {

      @apply .bg-orange

}
```

Similarly with the md, you'd need to place that within a media query.

Next Steps

Once you've created your website using Tailwind CSS, you may want to reduce the file size of your CSS. This can be done by removing unused settings from the tailwind.js file, and possibly adjusting the modules available. The documentation for Tailwind has details here: https://tailwindcss.com/docs/controlling-file-size

Conclusion

I hope this article is of interest to some people. Hopefully it will also encourage a few people to try out Tailwind after having previously dismissed it (as I originally did). If this was interesting, but you're more of a designer than a developer, feel free to look at Refactoring UI. If you enjoyed this format of article, please let me know by writing to me at lswest34+fcm@gmail.com. If you want to request an article, or see something that needs to be corrected, I can be reached at the above email for that as well.

Further Reading

https://refactoringui.com/ - Project by Steve Schoger and Adam Wathan where they offer design tips.

https://youtu.be/17OBlxY2C_0 - The video I refer to in the introduction. Adam Wathan does stream a lot of live coding, if that is of interest to you as well.

https://tailwindcss.com/docs/what-is-tailwind - The Tailwind CSS documentation.

Lucas has learned all he knows from repeatedly breaking his system, then having no other option but to discover how to fix it. You can email Lucas at: lswest34@gmail.com.

Python In The REAL World - Pt.84

Last month, I told you that we would build a GUI client for our sensor project, so let's get started.

If you set the wayback machine for February 2012 and open Full Circle Magazine issue 58, you will find an article written by me about Page a Tkinter GUI designer for Python. It was a two part article finished the next month in issue 59. Well, Page is still around and still being actively maintained and has gone through many changes since then. Page runs on Linux, Microsoft Windows, MacOS and the Raspberry Pi. The current version is 4.14 and can be found on his sourceforge website at https://sourceforge.net/projects/page/ . You will need to download the latest version to continue this project so head up there and pull it down. I'll wait for you.

Hmmm humm hummmmm. Laaa ti daaaa ti daaa.

See, it didn't take very long. Now, before we start talking about installation, you need to decide if you are going to put Page on your local computer, your Raspberry Pi or both. The reason for this is that the RPi needs some extra steps (not too many and not very hard, but extra steps nonetheless). I'll assume that you will be using Linux (rather than Windows or MacOS) for this install. If not, there are plenty of instructions included in the Page package for the other operating systems.

Page uses the Tcl and the Tk Tkinter toolkit to make the GUI. Most Linux distros come with Tcl and Tk already installed. That being said, it is a good idea to get the latest and greatest ActiveTcl, a free community Tck/Tk that you can download from https://www.activestate.com/activetcl.

In a terminal, type:

```
tar zxf /path/to/ActiveTcl-download.tar.gz
```

Change to the folder you extracted the files to and run the installer script. You should do this as root, since the installation goes into the opt folder. In the terminal type:

```
sudo ./install.sh
```

Finally, you need to ensure that your PATH variable includes the directory that contains the installed executable files. In the terminal type:

```
export PATH="/opt/ActiveTcl-8.6/bin:$PATH"
```

That's it for Tcl and Tk. Now we can unpack the Page distribution. (I use the file manager to unpack Page. You can put it wherever you want. The Page program will run from this folder, so make sure that it is easy to get to.

Now in a terminal, change to your Page folder. It is named 'page'. Type:

```
./configure
```

within the page folder that will create the script that will invoke Page. Finally, check to see if there are any '.pagerc' files in the main folder and remove them if there are.

If you are running Page on the Raspberry Pi, Tcl and Tk are not installed by default in the latest Raspbian distribution. However they are available in the repository. In a terminal, you will need to type:

```
sudo apt-get install tcl tk
```

Once you've done this, install Page as above.

One thing worth mentioning here. There is a wonderful PDF tutorial that comes with Page. It is located in the 'docs/tutorial/' folder and is called 'Learning Page - A Python GUI Designer' (shameless self promotion written by me).

Starting Page

In a terminal, change to your Page folder and type:

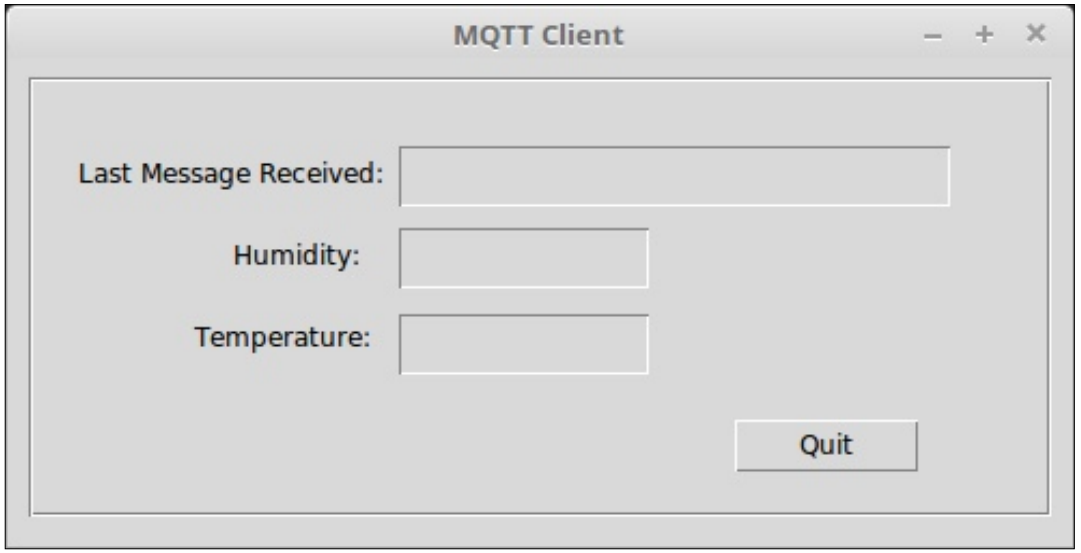
```
./page
```

This will bring up the Page

system and open five windows on your desktop. From the top left, clockwise, these windows are:

- The main Page control window
- The Attribute Editor
- The Widget Tree
- The Toolbox
- And finally, in the center, is the blank topmost widget.

If you don't remember the original articles on Page, you basically click on a widget in the toolbox, then place it by clicking on the topmost widget. You can move the new widget, resize it, set its attributes in the attributes editor, and then do it again until you have your UI looking the way you want. When you are finished designing the UI, you use the main control window to save the three files generated by Page. The first file is the .tcl file, the second is the Python GUI definition and the third file is the Python Support module. This changed a number of versions back, to allow the majority of the code to be put into the support module and not have to make many changes to the base UI file. I will walk you through the process, so don't worry. I'll go really quickly, so hold on. If you have any problems, you can run through the Page tutorial to get up to speed, then come back to this project.



We will be creating a very simple UI for our client. When it's launched, it will connect to the MQTT broker, subscribe to the Temperature and Humidity topics and when those messages are received, display the Temperature and Humidity values, as well as the time and date of the last message in label widgets. So there will be 6 total labels, three static as prompts and three active that will provide the visual data from the sensors. The three active labels have the relief, or outline, set to sunken to make them stand out from the static labels. There will also be a single button that will close our client. All of the widgets reside on a frame to group them. The topmost widget in this example is about 500 pixels by 255 pixels.

The finished UI should look something like the one shown above.

Once you have Page up and running, move the topmost widget somewhere close to the centre of the screen and resize it using the resize handles on the corners and sides. Next, using the Attribute Editor, set the Alias to "**mqttclient**" and the title to "**MQTT Client**". Now add a frame widget to hold all of our other widgets and make it look something like the image.

Now put the six label widgets organized into three rows of two labels each, again making the layout look something like our finished image above. Now here is the part that is a bit tedious. We need to set the attributes for each label. The three static labels are pretty easy. We will not set an alias for them, since they are just static. We do, however, need to set the text. Select the top left label and then in the Attribute Editor, scroll down until you see the 'text' attribute. It should read 'label' right now. Change it (click on the entry box) to '**Last Message Received**:'. Now move to the next one down, click it to select it and change it's text attribute to '**Humidity**:' and finally change the text attribute of the bottom left label to '**Temperature**:'.

Move to the top right label and select it. We have a number of attributes to set for these. First, we need to set the alias to '**lblLastMessage**'. This names the label widget for easy identification. Next scroll down a bit in the Attribute Editor until you find 'relief'. Clicking on the entry field will provide you a set of options. Select '**sunken**'. Finally, scroll down a bit more and find the text variable attribute. Put '**LastMessage**' into the entry box.

This will allow us to dynamically change the data in this label.

The next label down will have the alias set to '**lblHumidity**', the relief set to '**sunken**' and the text variable to '**HumidityValue**' (one word). Finally do the bottom right label and set the alias to '**lblTemp**', the relief to '**sunken**' and the text var to '**TempValue**'.

The last thing we need to put in our GUI is the button. Set the alias to '**btnQuit**'. Look down just a little bit in the Attribute Editor for the 'command' attribute. This is where we will put the name of the function that will be called when the button is clicked. Enter '**on_btnQuit**'. Finally set the text attribute to '**Quit**'.

Now we are ready to save our UI code. First, on the main Page window, select 'File|Save' from the menu. Select a folder to save the file into and save the file as 'mqttclient.tcl'. Once you have the .tcl file saved, find the 'Gen_Python' menu item and click it. Select the first option 'Generate Python UI'. It will show you the python UI code in a simple editor. Click on the 'Save' button. Again, select the 'Gen_Python' menu item, but this time, select the second option 'Generate Support Module'. Again, the simple code editor is shown, this time with our support code. Click save. All three code files are now in the same folder. You won't have to mess with the .tcl or the mqttclient.py file unless you need to change something in the main UI. The file we will be working with will be mqttclient_support.py. Open this in your favorite code editor.

Page has done a lot of the work for us when it comes to the GUI itself. If you were to run the 'mqttclient.py' file right now, everything should look like the sample image I showed you earlier. Our code will be providing the glue that binds all the widgets with the data coming from the broker. However, if you click on the 'Quit' button right now, nothing will happen, since we haven't written any code of our own to control what happens when the button is clicked. So you have to click on the 'X' in the top of the window.

The Code

Now we'll start dealing with our code. The first thing is to expand the import section to include the paho library, the datetime library and the locale library for printing the date/time properly. Page has already entered the sys library import for us. Below that, put in the following lines.

```
import paho.mqtt.client as mqtt

import datetime

Import locale
```

Now scroll down a bit until you see a function called 'set_Tk_var()'. This function, put in by Page sets up the global variables that we will be using to control our dynamic labels. Notice that these are all defined as 'StringVar()' which is a special type of string defined by Tkinter. Now find the function 'on_btnQuit()'. This is a CALLBACK function that is used when the Quit button is clicked. There are already two lines of code there from Page. Add the following lines below them:

```
# client.loop_stop()

destroy_window()
```

The first line is currently commented out since we haven't defined the client yet. We'll uncomment this in a few minutes but will allow you to run the UI and use the Quit button. The destroy_window() function is written for us (again by Page) to properly close and dispose of the UI.

Now in the init() function that Page set up for us, add the following four lines at the bottom of the routine.

```
lang = "en_US.utf8"  # Put
your locale here

locale.setlocale(locale.LC_AL
L, lang)

set_our_globals()

start_up()
```

The init() function is run right before the UI is shown to the user, so any code that we need to run to set up variables and widgets, or calls to startup functions should be put here. We want to make sure that our code is the last thing to run. We set 'lang' to whatever your locale settings should be. For example, in the line above, I use "en_US.utf8" since I live and think "U.S. English". If however, I were able to wrap my old brain around the date/time format in the UK, I could use "en_GB.utf8". (I tested it

and it worked but started causing me to get confused). If, on the other hand, I lived in Norway, I would use “nb_NO.utf8”.

The set_our_globals() function defines the ip address of the broker (I assume it is running on your RPi, so you will use the RPi’s address) and the two topics that we will subscribe to. If, when you programmed your DHTXX code last month you changed these topics to something else, make sure that the topic names match what your sensor program is publishing. Put this code (shown top right) right after the init() function.

Next, we will write our start_up() function (shown middle right). This is the code that gets everything going.

Notice that we instantiate the mqtt client and set two callbacks as we did in our RPi code. After that, we connect to the broker and call the subscribe_to_topics() function. Finally we start the client into a loop. Unlike a console program, we don’t call client.loop_forever(). This is because the loop_forever() function creates a blocking call which will not allow our GUI to ever update or respond to the mouse. The loop_start() call, however, does not create a blocking call, so we can use it to create the client listen loop (shown bottom right). However, we need to use the loop_stop() call before we end the program to stop the client.

In the subscribe_to_topics() function, we simply send a client.subscribe() call to the broker to make sure we get the messages.

```
def set_our_globals():
    global MQTT_SERVER, MQTT_PATH1, MQTT_PATH2
    MQTT_SERVER = '192.168.1.224'  # Enter the IP address of your broker
    MQTT_PATH1 = 'greghouse/dht22/humidity'
    MQTT_PATH2 = 'greghouse/dht22/temperature'
```

```
def start_up():
    global client
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect(MQTT_SERVER, 1883, 60)
    subscribe_to_topics()
    client.loop_start()  # Start the client loop.  Replaces loop_forever()
```

```
def subscribe_to_topics():
    global MQTT_PATH1, MQTT_PATH2, client
    client.subscribe(MQTT_PATH1,qos=1)
    client.subscribe(MQTT_PATH2,qos=1)
```

```
def on_connect(client, userdata, flags, rc):
    print('on_connect: rc={0}'.format(rc))
```

Here we use the client.subscribe(topic,qos) method to include the QOS value of 1, since the RPi is publishing the messages with a QOS of 1.

Our next function is the on_connect() callback function (shown bottom left). Once we get a connect ACK back from the broker, this routine is called. We simply print that we are connected to the terminal.

Now this is where the “magic” happens. The on_message() callback function (shown next page, top right)is run whenever a message comes in from the broker. We will handle any messages coming from the broker to us. We use an if/elif/else tree to handle the messages. First we get the time and format it into a U.S. date/time format, then first check to see if the word ‘humidity’ is in the message topic. If it is, we take the data (message.payload) and use the .set() method of the humidity label text variable to put the data into the label. Next we use a similar scheme to check to see if ‘temperature’ is in the

message topic and put that data into the label. Finally, if we get a topic that we aren't set up for (which should not happen, but can) we print the topic and message to the terminal window. Finally, we place the message date/time string into the proper label. I can only be sure that the datetime format shows as a U.S. datetime format. According to the research that I have been able to do, it should work in whatever language you use based on your locale setting. However, it might require you to use locale.setlocale(locale.LC_ALL, lang) before the tim = datetime... line, but seems to be discouraged by the majority of Python programmers.

That is all the code we need to do. Remember to uncomment the line client.loop_stop() in the on_btnClick() function so the program can shut down properly and then run your program. If your RPi is still publishing at a rate of one update every 5 seconds, you should see your data come through.

All three code files (*.tcl, *.py and *_support.py) have been placed on pastebin. The addresses

```
def on_message(client, userdata, message):
    tim = datetime.datetime.now().strftime('%x %X')
    print('Topic={0} Message={1}'.format(message.topic, message.payload))
    if 'humidity' in message.topic:
        HumidityValue.set(message.payload)
    elif 'temperature' in message.topic:
        TempValue.set(message.payload)
    else:
        print('unknown topic - {0} - {1}'.format(message.topic, message.payload))
    LastMessage.set(tim)
```

are:
mqttclient.tcl file -
https://pastebin.com/1e41SpNc

mqttclient.py -
https://pastebin.com/xiUPwaJu

mqttclient_support.py -
https://pastebin.com/byBuHPyX

I'm not sure what we'll be presenting next month, but hopefully it will be fun for all of us. Until then, ***HAVE FUN!***

Greg Walters is owner of RainyDay Solutions, LLC, a consulting company in Aurora, Colorado, and has been programming since 1972. He enjoys cooking, hiking, music, and spending time with his family.

As the saying goes, a picture paints a thousand words. Icons and images in your mind map create a link between the visual and the language centers of your brain. This promotes learning and imagination. When you see the word fruit, you don't see the letters in your mind, but you see images of what fruit are. By using icons and images in your mind map, you link to your mind's natural tendency.

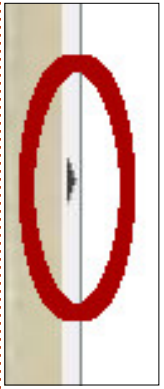
Freeplane includes a set of useful icons. Although the icons have recommended meanings, you can use them in a way that makes sense to you. Is an exclamation point a warning or emphasis? It's up to you and what it triggers in your mind. As with most things in mind mapping, the choice is yours.

Freeplane provides a way to insert your own images into your maps. To have the images display in the map, freeplane requires you save the image to the local hard drive. The recommended practice is to have them in the same folder as your map.

Icons

Freeplane provides over 90 icons in 13 categories and some specialized progress icons. You should think about the power of using icons in your maps. We use emojis, a special type of icon, every day to convey emotion in our messages. Icons can do the same thing in your maps. Use them as keys to trigger your mind to know at a glance what the node is about. You can display the icon in several ways, but the most complete options are through the menu.

The quickest access to the icons is through the icon side panel. Click on the expand arrow on the left-hand side of the Freeplane window to expand the icon sidebar. From the sidebar, you can select any icon to add it to the currently selected node(s). Using the keyboard, CTRL + F2 displays a table of all the standard icons.



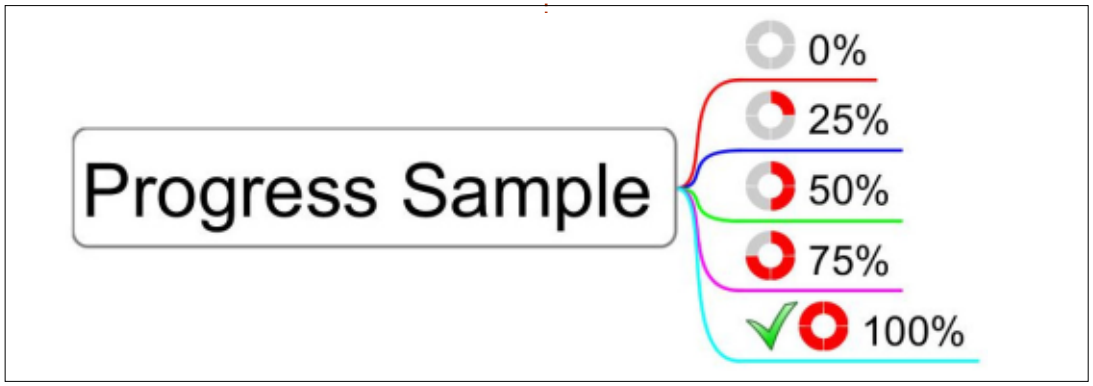
In the sidebar, table, and menus, there are three special icons, the remove icons. The first icon is the remove first icon, the second is the remove last icon, and the third is the remove all icons. In the sidebar and the table, they are the first three icons. In the menus, they are the last menu in Edit > Icons.

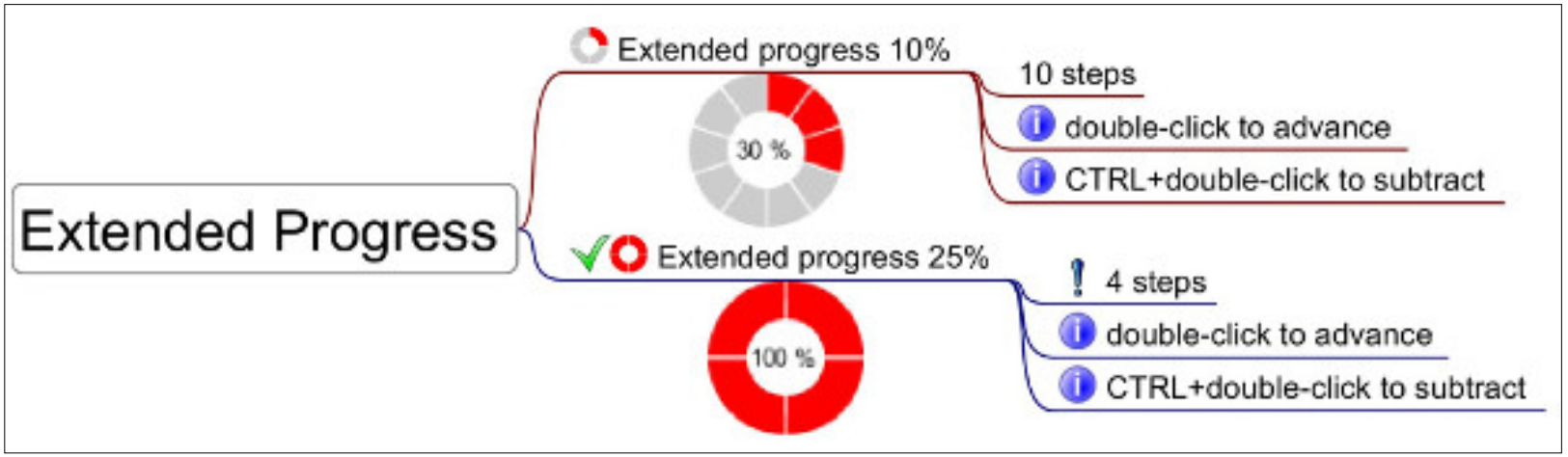
The menus under Edit > Icons gives you the most options and control of your icons.

Icons from table gives you the same table the CTRL + F2 keyboard shortcut does.

Icons by category breaks the icons into predefined categories. Keep in mind these categories are only suggestions. You can let the icons mean anything you want as long as it makes sense to you. The categories follow the same order as the icons in the sidebar and table.

The Progress icon (%) menu contains a special group of icons.





While you will find progress donuts in the icon table, the menu gives you the ability to automate your progress. These icons are great for tasks lists. I use them when outlining and writing.

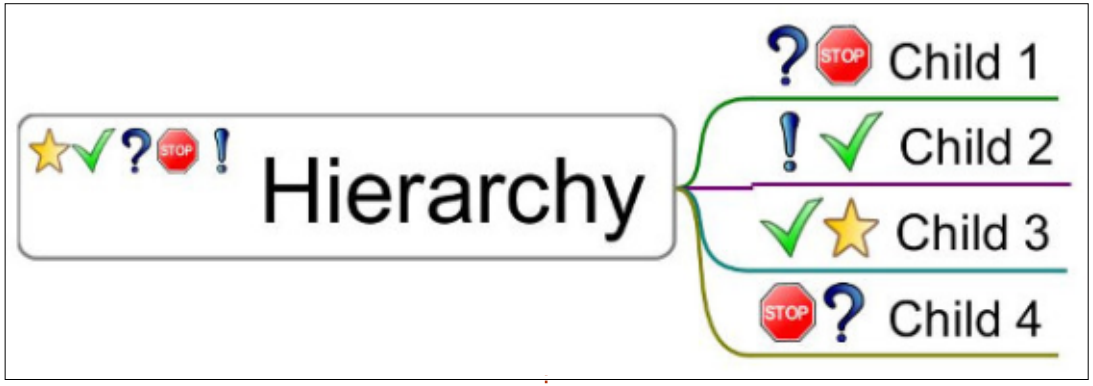
The Progress up and Progress down change the progress icon in the node. When either is first selected, the empty donut is added. If you select Progress up, the donut increases by 25% until the progress donut is full. If you select Progress down, the donut decreases by 25% until the progress donut is empty. Once the progress donut is full, a green checkmark icon is added to the node.

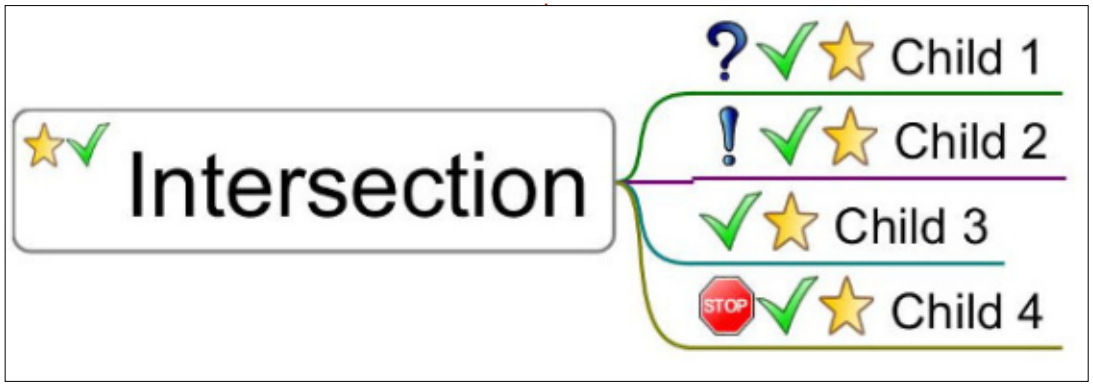
The Extended progress options add a larger progress donut below the node. The Extended progress comes in two types, 10% and 25%. The 10 % allows for a 10 step process while the 25% only 4. Selecting the 10% or 25% option adds the process donut below the node. Once created, you increase the value by double-clicking the large progress donut. To decrease the value CTRL + double-click. The green checkmark icon will show in the node when the progress donut reaches 100%. The node will also show a small progress icon in the node.

The Show icon hierarchy shows what icons are attached to any nodes below it. If the nodes underneath the node have icons, the node will display a smaller version of each of the icons. With a folded node, you can still know what icons are being used on the branch. Should you be looking for a certain icon based on your code, you can know you need to unfold the node.

The Show intersection of child icons is like Show icon hierarchy with a twist. The setting only applies to children of the node. If all the children have a certain icon, a smaller version of the icon will show in the node. This would allow you to see whether a folded node's children all contain a certain icon.

The remove icons menu gives you the Remove first icon, Remove last icon, and Remove all icons options.





Images

Freeplane lets you insert your own icons and images, too. You can insert the images as internal or external to the node. In order for the images to show in the map, you must save the images to the local disk. The accepted formats are GIF, JPEG/JPG, PNG, and SVG. In most cases, you will want to edit your image in an editor and scale it down to a useful size in your map.

Internal Image

When you insert an internal image, the image becomes the node core. The image within the node core is not scalable, so if you want a smaller size, you will need to edit it in advance. To create an internal image follow the menus Edit > Node core > Image by choice or link. Use the folders to browse to the image location and select it.

Click Open to create the image as the node core.

External Image

You can add images to the node details by drag and drop, copy and paste, and through the menus. In most cases, your need determines your method.

When you add an external image by drag and drop, the release point determines where the image is added. Release the image at the top of the node core, and the image becomes an external, scalable image in a new sibling. But, if you drop the image with the end highlighted, the image becomes an external image of a new child node.

When you use the copy and paste method, the image becomes an external image to a new child node.

To add an image as an external image of the currently selected node use the menus Edit > Node extensions > Add image. Browse to the image you want to add, select the image, and click Open.

To scale the external images hover over the edges of the image. A box will surround the image. Click and drag to scale the image to size. You can only scale the external images. Internal images are not scalable.

Icons and images create a connection between the visual and the language centers of your brain. Using icons and images in your mind maps creates association and clues about the content of a node. You can use the existing icon set to create your own coding system as well as adding other images as visual clues.

Elmer Perry is a technical support rep for an international keyless access company. He enjoys writing, woodworking, and technology. He lives in Leicester, NC with his wife.

Great Cow BASIC - Interrupts

In FCM#131 I showed you how to recognize the state of a switch and debug over the serial line. After having some troubles with the code in this article, I had to have a break and get things together. So with some delay here we are now learning about interrupts, what they are and what for they are helpful. My apologies if this caused some inconveniences.

Interrupt service routines

An interrupt does what the term describes, it interrupts a running program - or better: causes a jump in the program - and executes another piece of program code. The so called 'interrupt service routine'. The formerly interrupted program will be executed at the very same stage the interrupt has occurred. The ATTiny13a has ten interrupt vectors (or program spaces) you can place code at, which is handled, if you set the bits in the corresponding interrupt registers. For our purposes we have a closer look at the INT0 and PCINT0 interrupts. Both interrupts can wake the microcontroller from its power saving states which will be evident later on.

For the INT0 interrupt you only have one PIN available, the PIN B1 (PORTB.1). This restriction comes with some advantages, with INT0 you could set the trigger to four different states:

- if the level on PIN is low,
- any logical change,
- the falling edge or
- the rising edge of the signal (the transition from the high to the low level).

The PCINT0 interrupt can be used with all remaining PIN except Vcc and GND. The disadvantage of this interrupt is that it triggers on every transition and you have to determine which PCINT has fired. If only using one PIN this is a small caveat, but if using more than one, you would have to determine which one has 'pulled the trigger'. Another possible solution regarding switches and interrupts would be a polling routine. At certain intervals it would be checked if a switch was pressed or not. For this you would use the service routine of a timer interrupt, but for one button only, it would be kind of over-engineering. So without further ado, here are some sample codes for INT0 interrupt routines.

Vector No.	Source	Interrupt Definition
2	INT0 (PB1 only)	External Interrupt Request 0
3	PCINT0 (PB0..5)	Pin Change Interrupt Request 0

The following page (top left) shows a small pull-up version of an INT0 service routine in Great Cow BASIC.

The code to use the Pin Change Interrupt service routine instead is shown on the following page at the bottom left.

Debouncing the button presses

In addition you might not change the state of the LED everytime the interrupt occurs, because the switch could be bouncing or the signal quality is poor. Therefore you could add a simple debouncing algorithm. An example for the pull-up version with the INT0 interrupt in Great Cow BASIC is shown, page after next, top left.

And the debouncing routine with the pin change interrupt in Great Cow BASIC is:

```
#CHIP tiny13a, 1.2
#OPTION EXPLICIT
#DEFINE LED PORTB.0
#DEFINE TOGGLE PINB.0
#DEFINE SWITCH PORTB.1

DIR LED OUT
DIR SWITCH IN          'Data dir. is in, having the switch at PB1
SET SWITCH = 1         'int. pull-up active, uncomment line if you use an ext. one
SET LED = 0            'LED should be off at the start

MCUCR = 0b00000010     'INT0 reacts to the falling edge of the signal
INT0 = 1               'Enables the INT0 interrupt

ON INTERRUPT ExtInt0 CALL checkSwitch  'define interrupt service routine

'Main program:
DO
LOOP                   'intentionally left empty

'Interrupt service routine:
SUB checkSwitch
     TOGGLE =! LED
END SUB
```

```
#CHIP tiny13a, 1.2
#OPTION EXPLICIT
#DEFINE LED PORTB.0
#DEFINE TOGGLE PINB.0
#DEFINE SWITCH PORTB.1

DIR LED OUT
DIR SWITCH IN    'necessary for all input methods where the switch is at PB4
SET SWITCH = 1         'comment if you want to use an external 10 kOhm resistor
SET LED = 0

PCIE = 1         'Enables the PCINT interrupt
PCMSK.PCINT1 = 1 'Enables the PCINT1 interrupt only at PortB.1

ON INTERRUPT PinChange0 CALL checkSwitch

'Main program:
DO
LOOP             'intentionally left empty

'Interrupt service routine:
SUB checkSwitch
TOGGLE =! LED
END SUB
```

Breadboard circuitry

For the test with the internal pull-ups activated you would need an LED, a switch and the microcontroller (circuit on next page, bottom left). Connect one end of the switch to PIN 6 (PB.1) and the other to PIN 4 (GND). The anode of the LED goes to PIN 5 (PB0) and the cathode of the LED goes to PIN 4 (GND). After flashing the hex-file and connecting the power supply the LED should go on and off after pressing the switch.

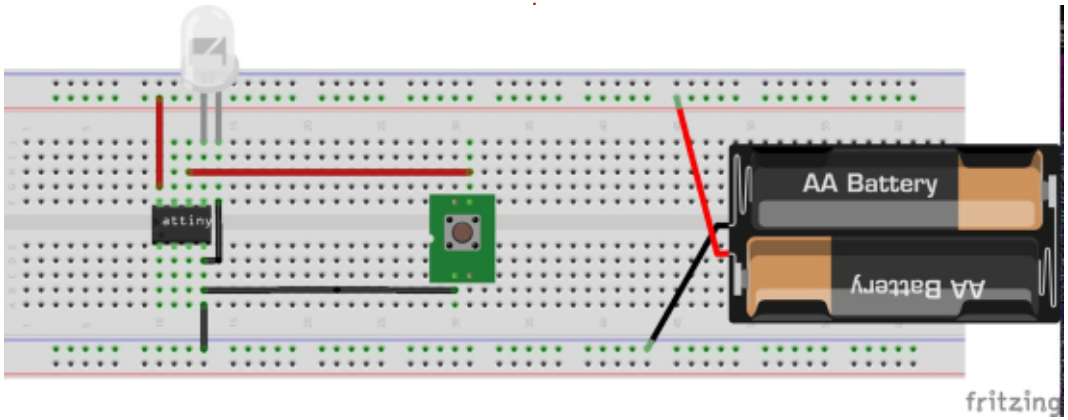
For testing purposes you would not change the hardware test circuit with above codes, with the pull-down version you would need to change code and also the breadboard circuit.

Conclusion

Interrupts permit the programmer to execute code only at a defined situation and give the advantage to give the main program the most computational power while running - except a interrupt is recognized - and do not have a need for a breaking scanning routine (e. g. check all 10 cycles if button is pressed). Or the

contents ^

```
#CHIP tiny13a, 1.2
#OPTION EXPLICIT
#DEFINE LED PORTB.0
#DEFINE TOGGLE PINB.0
#DEFINE SWITCH PORTB.1

DIM BUTTON AS BYTE
DIR LED OUT
DIR SWITCH IN          'Data dir. is in, having the switch at PB1
SET SWITCH = 1         'int. pull-up active, uncomment line if you use an ext. one
SET LED = 0            'LED should be off at the start

MCUCR = 0b00000010     'INT0 reacts to the falling edge of the signal
INT0 = 1          'Enables the INT0 interrupt

ON INTERRUPT ExtInt0 CALL checkSwitch  'define interrupt service routine
BUTTON = 0

'Main program:
DO
IF BUTTON > 3 THEN
      TOGGLE =! LED
      BUTTON = 0
END IF
LOOP

'Interrupt service routine:
SUB checkSwitch
      DO WHILE SWITCH = 0
            WAIT 1 US
            BUTTON += 1
      LOOP
END SUB
```



other way around interrupts gain the developer the possibility to create a power saving device which has the maximum computational power only when needed and does consume as little power as possible the rest of the time.

Sources

If you want to download the sources instead of copy-pasting it, you can now check it out with git or an SVN client. Have a look at https://github.com/Anobium/Great-Cow-BASIC-Demonstration-Sources/tree/master/Publication%20Solutions/Full%20Circle for more information.

References

1. pull-up and down circuitry breadboard-friendly explained, in german https://elektronik.skyline-service.de/elektronik/pullup-und-pulldown-widerstand/

Acknowledgement

I wish to thank Evan Venn (***Anobium***) from the Great Cow BASIC Team for his insights and valuable hints.

```
#CHIP tiny13a, 1.2
#OPTION EXPLICIT
#DEFINE LED PORTB.0
#DEFINE TOGGLE PINB.0
#DEFINE SWITCH PORTB.1

DIM BUTTON AS BYTE
DIR LED OUT
DIR SWITCH IN    'necessary for all input methods where the switch is at PB4
SET SWITCH = 1   'comment if you want to use an external 10 kOhm resistor
SET LED = 0

PCIE = 1               'Enables the PCINT interrupt
PCMSK.PCINT1 = 1 'Enables the PCINT1 interrupt only at PortB.1

ON INTERRUPT PinChange0 CALL checkSwitch
BUTTON = 0

'Main program:
DO
IF BUTTON > 3 THEN
     TOGGLE =! LED
     BUTTON = 0
END IF
LOOP

'Interrupt service routine:
SUB checkSwitch
     DO WHILE SWITCH = 0
          WAIT 1 US
          BUTTON += 1
     LOOP
END SUB
```

Boris holds a bachelor degree in business administration and works for an insurance company. While not working, he is a family person and enjoys playing with his kids or tinkering with his personal projects. Contact info and additional material at his site: https://www.evil-publishing.de/fcm

Inkscape - Part 74

Last time I showed you the basics of including an Inkscape image in a web page by treating it as an image object in an HTML <img> tag or via CSS, either by exporting as a PNG, or by linking directly to the original SVG image. This time I'm going to consider the other two methods of getting an Inkscape image into a web page: via the <object> tag, or through the use of inline SVG.

At first glance the HTML <object> tag isn't too dissimilar to the <img> approach. The "src" attribute is replaced with "data" and, because <object> tags can include more than just images, we have to include a "type" attribute that contains the SVG MIME type so that the tag knows what format the data will be in.

```
<!DOCTYPE html>

<html>

<head>
  <title>SVG in HTML</title>
</head>

<body>
  <h1>OBJECT tag</h1>
  <object
type="image/svg+xml"
data="circle.svg"></object>
</body>

</html>
```

SVG as an <img> is sandboxed by the browser such that any JavaScript in the file won't run and no external resources can be loaded (i.e. no web fonts). Even code in the hosting page won't be able to dig into the contents of the SVG to dynamically change the image. In fact the SVG image behaves largely the same as a rasterised version of the same picture.

With the change from <img> to <object> comes vastly more power. The SVG file is treated as a complete, self-contained, document which is allowed to execute JavaScript, load external resources, and even communicate with the host page. In future instalments of this series we'll explore some of those possibilities in more detail.

The fourth and final approach to including SVG in your web page is so-called "inline SVG". Inline, in this case, simply refers to directly intermingling the XML code of your SVG file with the HTML code of your web page. For example, here's the code for an HTML page with an inline SVG image of a red circle with a thick black stroke:

```
<!DOCTYPE html>
<html>
<head>
  <title>Inline SVG</title>
</head>

<body>
      <svg>
            <circle
            fill="red"
            stroke="black"
            stroke-width="11"
            cx="75"cy="80"
            r="60">
            </circle>
      </svg>
</body>

</html>
```

At a glance the code looks pretty straightforward. Most of it is just the sort of content you would expect of a typical web page, except that the contents in the <body> tag consist of an <svg> block, with some SVG code inside it. The SVG itself is just a single <circle> element with a few attributes to create the desired result. It's all very neat and tidy. Perhaps too neat and tidy, in fact.

This code was hand crafted to ensure that only the bare minimum appears in the SVG block. A typical SVG file from Inkscape, on the other hand, is filled with all manner of extra content, most of which isn't necessary when using it in this way. Here's a screenshot from an HTML page that contains both the hand-crafted code above, and a copy of the XML created by Inkscape for the same basic shape:

Inline code (hand-crafted)

Inline code (Inkscape)

Can you spot the difference? No, me neither. But the top circle required SVG code amounting to 107 characters, whereas the second used 2118 – nearly twenty times as much code for the same result! Surely Inkscape can do better than that?

There are two reasons for the bloated file size. The first is simply that a hand-coded SVG image always has the potential to be smaller than a computer-created equivalent. Inkscape is a general purpose tool used for all manner of tasks, so it creates general purpose files which are good enough for most circumstances, but which lack the targeted optimisations that would bring the file size down. Unless you want to shave every last byte of flab from your files it's not usually worth worrying about this first problem. But the second issue is definitely worth looking at, as it's responsible for far more of your file's expanded girth: XML namespaces.

XML stands for eXtensible Markup Language. Yes, I know. I can only guess it's because ".eml" was already taken as a file extension for email files, and "X" is way cooler than "E". But that X or, more to the point, extensibility, is at the heart of the file bloat we can see with Inkscape. One of the more sensible design decisions in XML was to allow different XML-based languages to coexist, even inside the same document. You can see the remnants of this decision in the code above, where we switch from HTML (a not-quite-but-close-to XML language) to SVG, without the browser kicking up a fuss. But in this specific case HTML has been extended to allow the inclusion of SVG elements; XML allows such inclusions in a more generic manner, without the need to extend the host language in any way.

Let's imagine that we wish to combine two XML languages in a single document. The first is our old friend SVG. The second is a new language I've just made up for garden designers – let's call it GardenML. Before you cry foul, XML is explicitly intended to allow you to create your own domain-specific languages, so although my example is a little contrived, it's not without precedent in the real world. So, back to the plot: we've got an SVG file, filled with <path> elements that describe lines in an image. But we want to include our GardenML content, which is filled with <path> elements that describe… well, paths in a garden. The sort you walk on, made up of bricks or stone slabs, and certainly not the same thing as a path in SVG.

How does an XML consuming program reconcile these different uses of the same element. How does it know when you're using an SVG <path>, and when it's a GardenML <path>? The answer is a thing called a "namespace". All XML elements have a namespace, but it's usually set as a default for the whole document, then just implicitly used for each element. Here's an SVG document which does exactly that:

```
<svg
xmlns="http://www.w3.org/2000
/svg">

      <path d="M 130,70 A
      60,60 0 1,1 130,69.9 Z"

      fill="red"
      stroke="black" stroke-
      width="11" />

</svg>
```

Notice the "xmlns" (XML NameSpace) attribute in the <svg> tag? That defines the default namespace for the document. The value is a URI – a Uniform Resource Identifier. It may look like a web address, also known as a URL (Uniform Resource Location), but it's subtly different. Whereas a URL points to the location of a resource such as a web page, a URI is just a unique identifier which doesn't have to have a corresponding resource at the specified address (though they often do). In other words, a URL is a path to a real document, whilst a URI is a unique string used to make sure that each namespace is distinct from all the others. A URI may also be a URL, but doesn't have to be.

When a browser sees that the document's default namespace is "http://www.w3.org/2000/svg" it knows that any <path> elements are SVG paths, not GardenML paths. But suppose we want to create a file that will both display in the browser and can be interpreted by something that understands GardenML. In that case we need to define more than one namespace in the <svg> element, give all but the default one a handy shortcut name, then prefix the elements and attributes with the shortcut as appropriate.

```
<svg

xmlns="http://www.w3.org/2000
/svg"

xmlns:garden="http://fullcirc
lemagazine.org/GardenML">

      <path d="M 130,70 A
      60,60 0 1,1 130,69.9 Z"

            fill="red"
            stroke="black"
            stroke-width="11"
            />

      <garden:path
      start="back door"
      end="shed"
      type="gravel" />

</svg>
```

This time the default namespace is still SVG, so the <path> element still renders in a web browser. The second <path> element, however, has a prefix of "garden", identifying it as being from the GardenML namespace that's defined in the <svg> tag. The browser doesn't try to render the second path because it doesn't know what to do with elements in that namespace – but equally it doesn't complain about them either.

By default attributes are in the same namespace as their element – so in the previous example the

```
<svg
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:cc="http://creativecommons.org/ns#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:svg="http://www.w3.org/2000/svg"
   xmlns="http://www.w3.org/2000/svg"
   xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-0.dtd"
   xmlns:inkscape="http://www.inkscape.org/namespaces/inkscape"
   width="40mm"
   height="40mm"
   viewBox="0 0 40 40"
   version="1.1"
   id="svg8"
   inkscape:version="0.92.2 (unknown)"
   sodipodi:docname="circle.svg">
```

"d" attribute is in the SVG namespace, whereas the "start" attribute is in the GardenML namespace. But you can prefix individual attributes as well, should you need to. In this example we've got an SVG path to which I've added some custom attributes of my own:

```
<path d="M 130,70 A 60,60 0
1,1 130,69.9 Z"

      fill="red"
      stroke="black" stroke-
      width="11"
      garden:start="gate"
      garden:end="front door"
      garden:type="paved"
/>
```

Now all of this might seem a little esoteric, but there are two reasons for explaining it. The first is that namespaces are integral to XML documents, so when we get round to manipulating Inkscape files using JavaScript a little later in this series you'll be glad of a good grounding in the topic. The second is that it explains why our Inkscape generated files are so much larger than a hand-crafted version. Here's the opening <svg> element of a typical Inkscape file:

Notice all the different namespaces being defined, and a couple of them being used on the last two attributes. The "svg" namespace you now know about, but what of the others? The "inkscape" namespace is used to store extra attributes that hold Inkscape-specific data. Without these, Inkscape would be limited to the features defined in the SVG spec, and wouldn't be able to provide extra capabilities such as Live Path Effects. The "sodipodi" namespace serves a similar purpose – Inkscape was forked from the Sodipodi program many years ago, but its history lives on in attributes that date from before the split.

The remaining namespaces are used for the metadata about your document that you can enter into the Document Properties dialog. There are several of them because they each refer to a different XML language. Inkscape could have just used it's own namespace for all of them, but by referring to other well-known languages it improves the ability of the metadata to be automatically parsed and understood by indexing programs or other XML tools. It does bloat

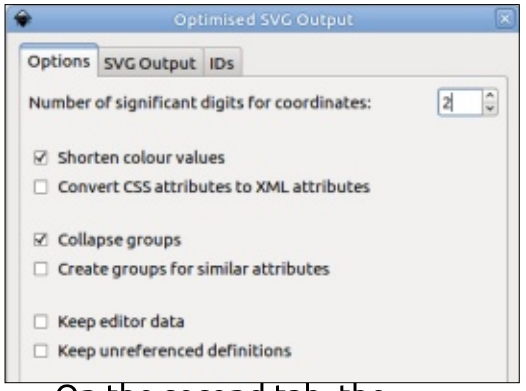
the size of the file quite considerably, though.

An obvious way to reduce the file size, therefore, is to remove the additional namespaces, elements and attributes. Doing this obviously compromises the SVG file in various ways, from removing the editability of Inkscape-specific features, to loss of metadata. Inkscape does, however, provide a couple of options for doing this for you. They're both alternative formats in the File > Save As… dialog, but in practice I recommend still saving a normal Inkscape SVG file, then creating your slimline version using File > Save A Copy…. This approach avoids the problem of forgetting to save in the full-fat format when you've made an edit, and losing data in the process.

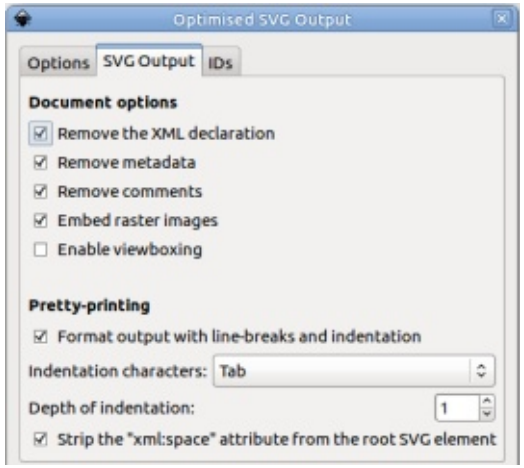
Your first choice is to save as "Plain SVG". This strips out the proprietary Inkscape and Sodipodi namespaces, and their associated elements and attributes. It still leaves the other namespaces intact, so the file will still contain any metadata you entered into the Document Properties dialog. This option is ideal for use when linking to an SVG file via the <img> or <object> tabs, as it removes the data that the browser doesn't understand, but leaves any copyright or license information that you may have added. With my simple example file, saving as Plain SVG reduced the file size from 2.1kB to 1.3kB.

The other choice is to save as "Optimised SVG". This presents an additional dialog that lets you choose from a wide range of optimisations that can potentially reduce the file size. Be warned that this option can lead to quite extensive changes in the structure and content of your SVG file, so always test the resultant file to make sure you haven't optimised away something important. Be particularly wary of reducing the number of significant digits too far, as this will affect the fidelity of your image.

When it comes to the additional namespace-related data, there are a couple of key fields to pay attention to. On the first tab, the "Keep editor data" checkbox determines whether the Inkscape and Sodipodi namespaces are preserved.



On the second tab, the "Remove metadata" option will lose all the other namespaces, together with any metadata you may have added to the Document Properties.



Also on the second tab, pay attention to the Pretty-printing options. Turning this off can reduce the size further, but if you're trying to create a file to put inline in an HTML page, a little formatting can make it a lot easier to work with.

So how well does Optimised SVG stack up against hand-written code? It doesn't reach the 107 characters of my carefully crafted version, but at 277 characters of fairly readable SVG it doesn't do too bad a job. For most cases where you want to inline your image into an HTML page it will be good enough, with far less scope for errors than trying to write everything by hand.

Phew! We've covered quite a lot this month, from <object> tags to XML namespaces. If all you want to do is to include a static image in a web page, stick with the methods described last time. But as we move on to more advanced topics, such as incorporating code into our files, the details of this article will become more relevant.

Mark uses Inkscape to create three webcomics, 'The Greys', 'Monsters, Inked' and 'Elvie', which can all be found at http://www.peppertop.com/

Statistical Analysis

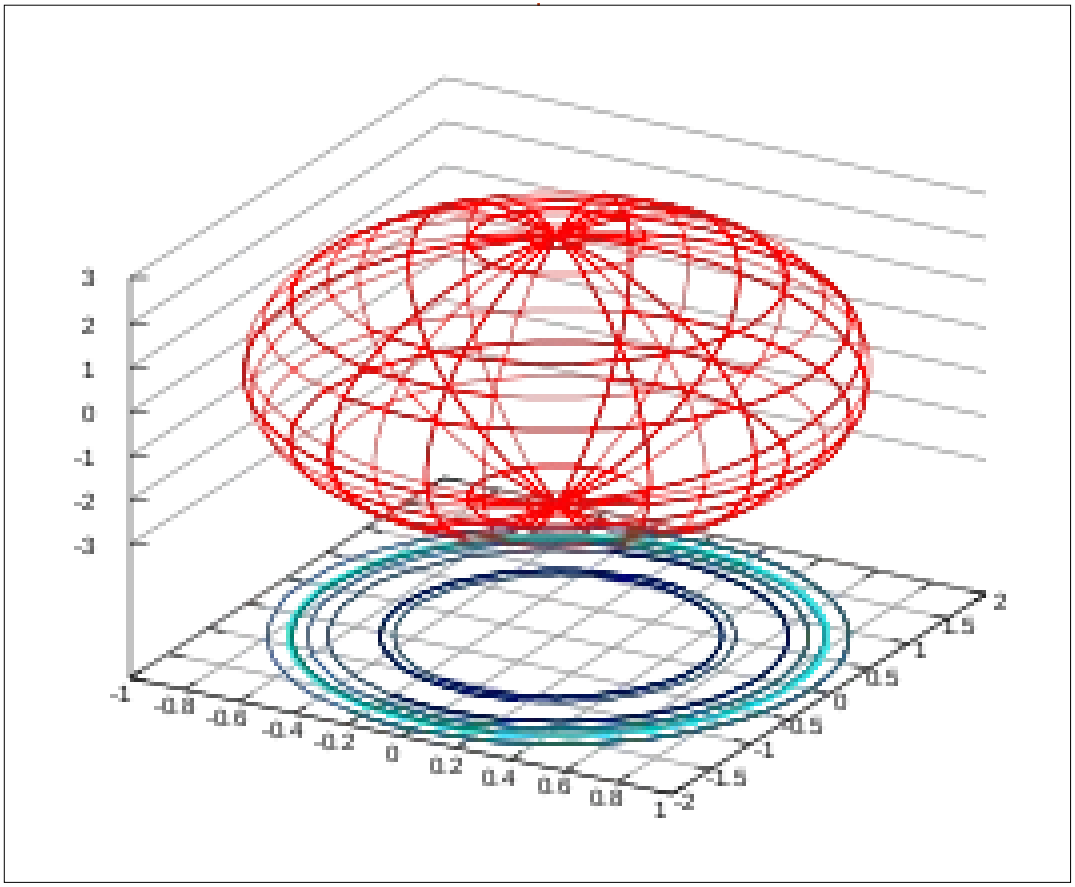
The core of any clinical research: is this hypothesis statistically significant? I am not a biostatistician, but I am often asked for basic statistics such as standard deviation, mean, and intraclass correlation coefficient. I have a Perl batch program that fulfills these demands. However, I am limited by the batch program. I only can generate simple tables, and no graphs.

There are commercial resources like Mathematica and Matlab that can develop a deeper analysis for research data sets. Yet yearly licenses and vendor contacts are required for these products. There are two other programs available: R and Gnuplot for Linux users.

R is the continuation of a math programming code called S. It is a full language used by statisticians. It is hailed as one of the easiest and most robust languages to learn and implement for work streams. Numerous people are learning this language. Further information can be found here: https://www.r-project.org/

Gnuplot is a command line tool that can generate tables and graphs. Gnuplot does not qualify for a totally free and open software. It is free to use, but in the same vein it is similar to Vanderbilt's REDCap. It is a closed software tool, but any researcher can use it.

When I first used Linux Mint, I attempted to use a front end graphical program of Gnuplot. Yet it was not very reliable nor stable. I recently became owner to a book titled Gnuplot in Action: Understanding Data with Graphs. I am hoping that my line of research in motion capture and pressure can be turned into graphs instead of tables. I review low back pain in the lumbar region and other pelvic landmarks such as the sacrum and pubic tubercles.



The motion data is generated as a comma separated value file. The pressure is a tab separated value file. I often blend these two files into one "mega file" that I process into generate a database. I will attempt to utilize the book, and see if i can apply it to my research line.

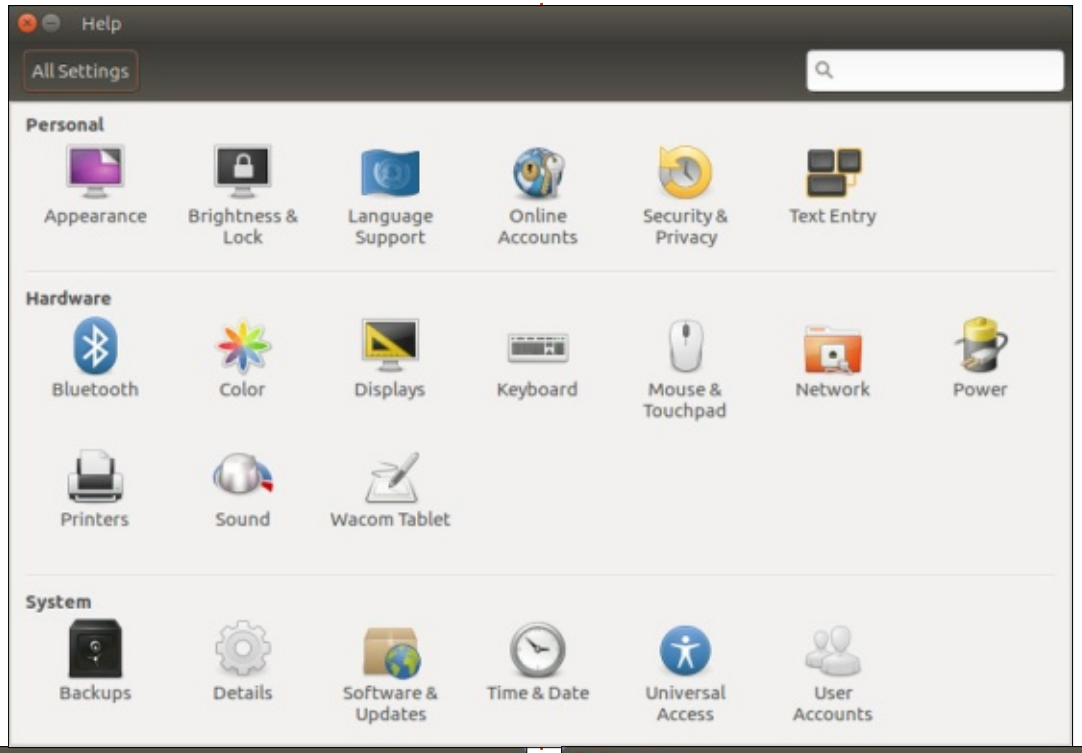
SJ Webb is a researcher coordinator. When he is not working, he enjoys time with his wife and kids. He thanks Mike Ferarri for his mentorship.

System Settings Pt.2

Last month, we looked at Ubuntu's System Settings and we'll finish that this time, with basics on how to customize the Unity desktop. Click the gear icon in the upper right:

In the menu that comes up, go to System Settings, then Keyboard, and we'll pick up where we left off last month.

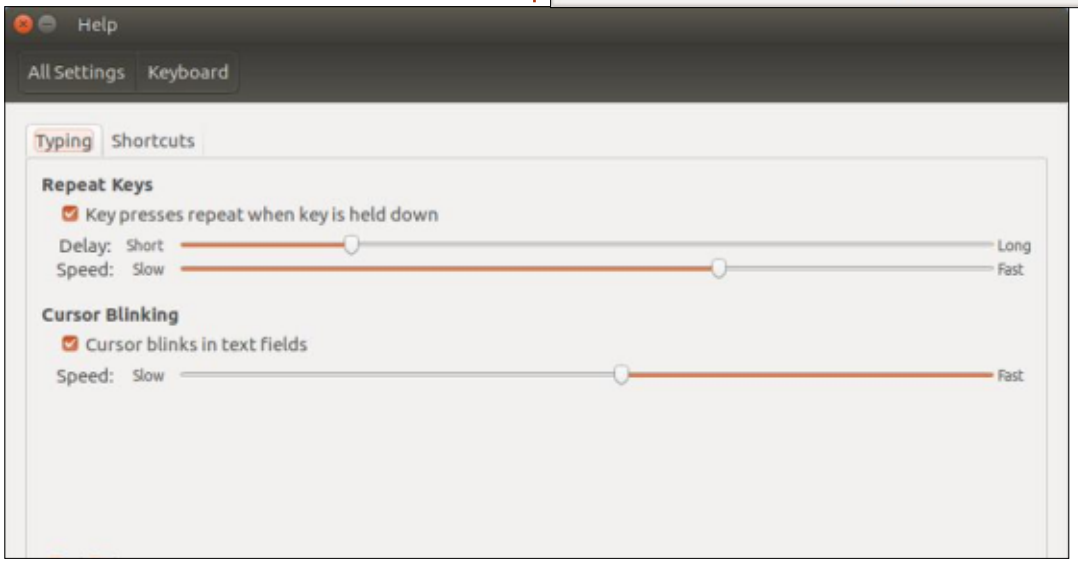
Most users won't really need to tweak the Keyboard settings, although you can modify keyboard shortcuts and key repeat timing here, if you're so inclined (below).

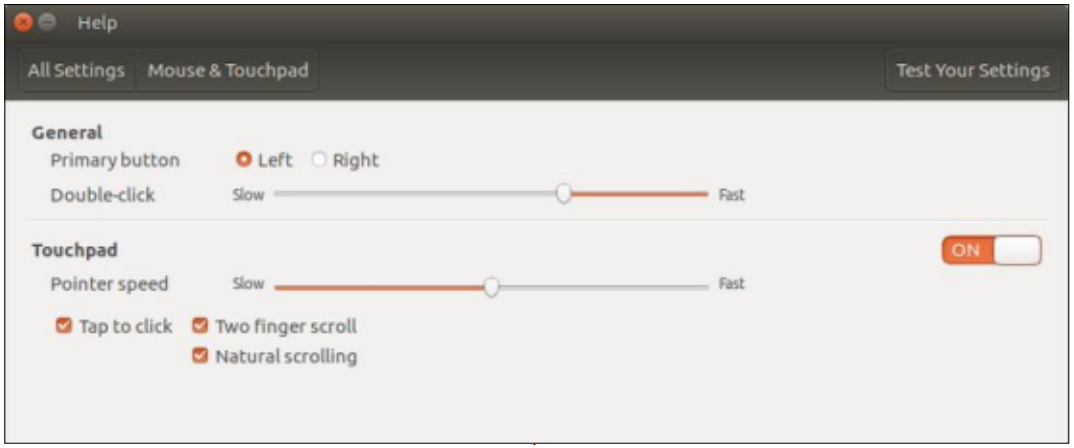


Here is the Keyboard shortcuts screen (bottom right).

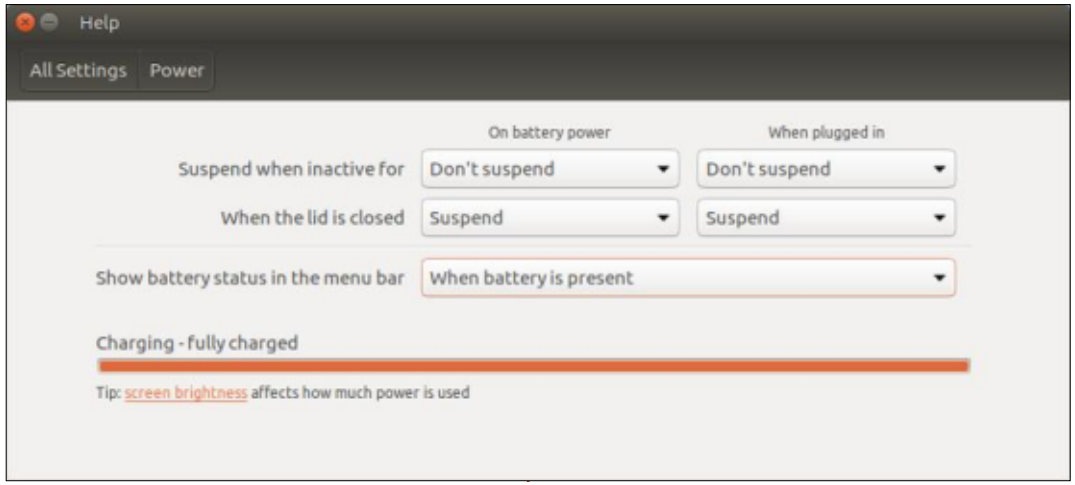
Mouse and Touchpad (following page, top left) will allow you to change double-click speed, pointer speed, and other mouse-related settings, so it's worth looking at as part of the fundamental ergonomic experience of working on the Unity desktop:

Since your average personal Linux user isn't really going to manually configure networking, we'll skip that and look at Power settings (following page, top right).





This is also one of the first things I personally adjust when setting up a new system, as I think the defaults are ridiculously short and impractical. I also typically don't want my system ever suspending on its own when it's on AC power, I'll always suspend manually in that case. You may feel differently.



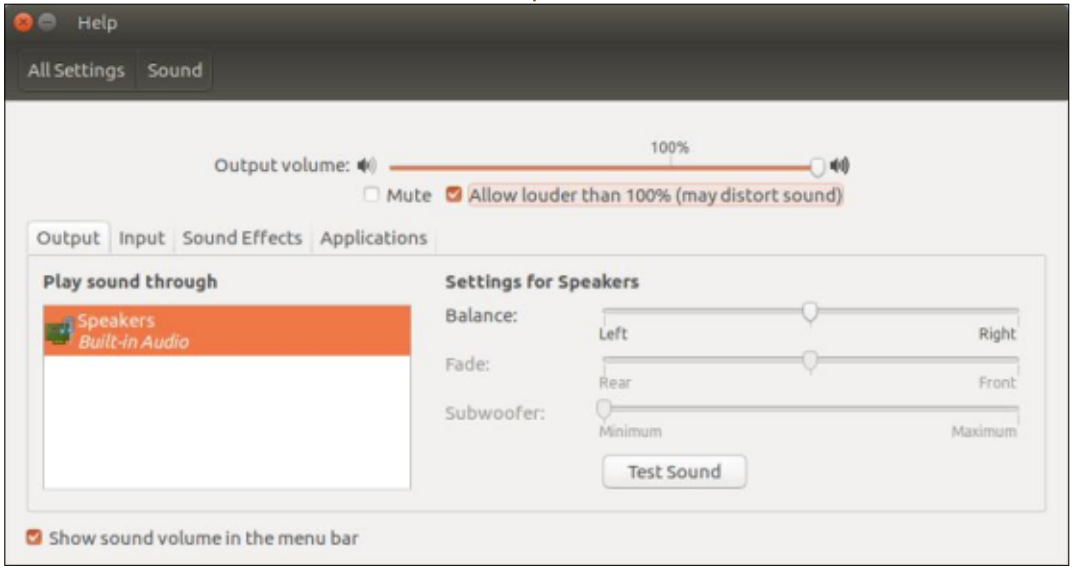
Printers shows what printers are set up on your system , obviously enough. It will vary quite a bit depending on what specific hardware you have, so we'll leave that aside for now and look at Sounds (bottom left)

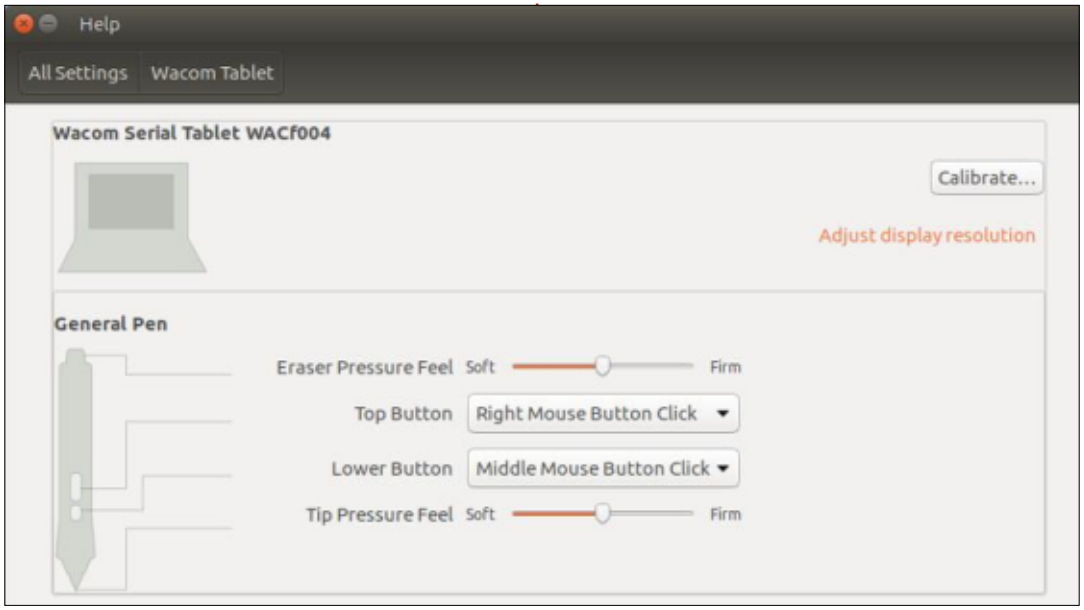
You can adjust volume, speakers, balance, and other aspects of your system's sound characteristics here.

If you have a Wacom tablet or pen-enabled screen, like my Fujitsu Lifebook, you will have a Wacom setting screen where you can make adjustments to its behavior (below)

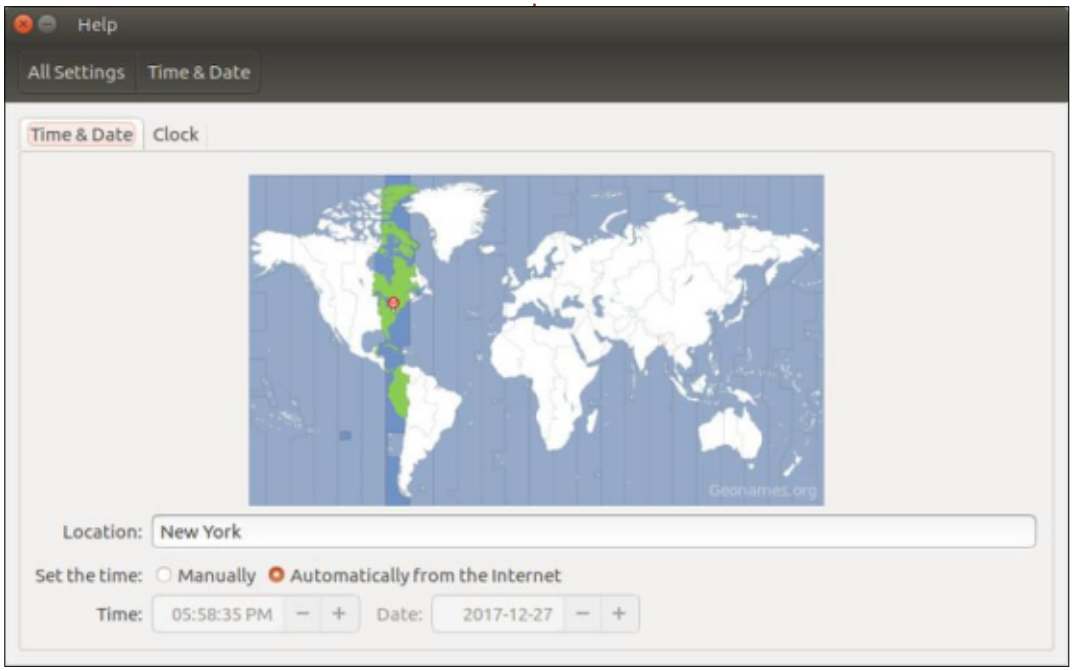
The Details setting (following page, top left) shows you information on your system.

You can change Default Applications (following page, top

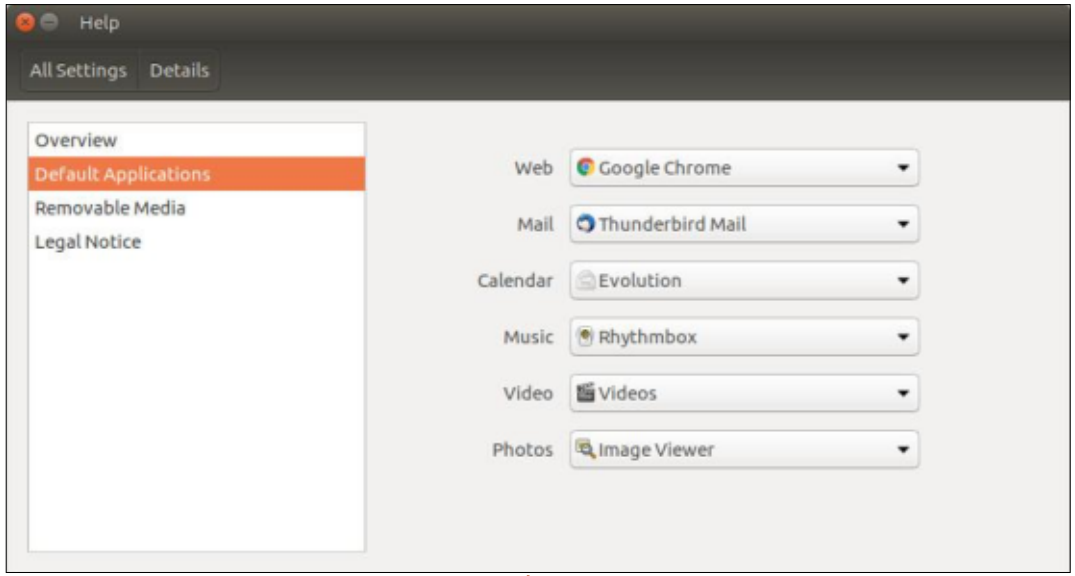




right), check your system version, memory, disk size, processor, graphics card, and manage your Removable Media here.



Default Applications is definitely worth evaluating, as you may want to change from the chosen defaults, and may even



want to do this on multiple occasions over time:

Chrome may not be your favored web browser, you might want to use another e-mail client instead of Thunderbird, and Rhythmbox might not be your favorite music player. Video and Photo management can also be changed, along with your default Calendar. All these preferences are likely to change over time for most users. In fact, I decided to change my own Calendar and Video options while writing this article.

We're going to look at Software and Updates in substantial detail next month, so we'll finish up for this month with Time and Date (bottom left)

You can set your time zone, time and date, time setting configuration, or adjust your system's clock settings.

One other aspect we'll touch on this month is managing desktop shortcuts (or, technically, Symbolic Links). The Unity desktop in previous iterations would not let you put shortcuts on the desktop, which I found annoyingly limiting. Desktop shortcuts can be a great time saver for applications you don't use enough to put on the Launcher, but still use with some frequency.

To put a new shortcut on the

desktop (or the Launcher), go to the Dash and find the application, then click and drag it to an open area on the desktop, or to the desired location on the Launcher, then release the mouse button to place it. If it's on the desktop, the Dash will likely cover the area where you want to place it as a final location, but just drag and drop to an open space, minimize or close the Dash and then drag and drop the new shortcut to wherever you want it.

Setting up desktop shortcuts is very simple once you know what to do, but not necessarily all that intuitive if you don't already know the procedure.

You can also organize desktop shortcuts into folders by right clicking the desktop and selecting New Folder, and then dragging existing shortcuts into the new folder. I find this a much better way to organize and access my applications than relying on the Dash, and well worth the extra effort to set up.

Next time: Updating applications and Linux itself.

I invite feedback on easier/better ways to do things. Any such submissions in response to articles or content will be considered the property of Full Circle Magazine for publication purposes, without remuneration, unless the writer/commenter specifies otherwise. That said, commentary and feedback are heartily encouraged and appreciated, at acer11kubuntu@gmail.com.

Richard 'Flash' Adams spent about 20 years in corporate IT. He lives in rural northwest Georgia, USA, with his adopted 'son', a cockatiel named Baby.

Full Circle 2018 Survey

It's that time of the year again where we ask what you think of FCM, Ubuntu, and Linux.

Some questions are a requirement, some you can skip over if not applicable.

Your answers will help shape Full Circle, so please use your constructive criticism. If you don't tell us what you think, or what we're doing wrong, then we won't know.

Survey URL:
http://bit.ly/fcm2018

Full Circle Weekly News

Join our new hosts Wayne and Joe as they present you with a short podcast (<10min) with just the news. No chit-chat. No time wasting. Just the latest FOSS/Linux/ Ubuntu news.

RSS:
http://fullcirclemagazine.org/feed/podcast

MY STORY

Written by Paolo Pelloni

Part 3

Last month we ended on a happy note, with all the information we needed to attempt to solve the issue at stake. What was it? I wanted to be able to enter sort information in Rhythmbox in an easier way than to do it manually track by track. Which is boring when all the tracks on the same album share the the same data, plus it was frustrating that at times they even got lost (maybe a bug, maybe only my case). We figure out that an xml file contains the data we want to manipulate and python offers an xml library ready to do it. So what is next?

I had to decide if I wanted to write a command line interface program (CLI) or a graphical user interface (GUI). The former are the text only ones you launch from the terminal, like grep or mp3diags, the latter ones are with windows and icons like audacity or Gnome text editor (gedit).

Surely enough a GUI program is nicer and looks like the preferable choice. As I said in the introduction my programming knowledge needed a brush up and I had no experience in linux / gnome graphic environment. Add to that it would be my first python program I was very sceptical on going down that road.

I have also seen that some linux applications are CLI based and then they also have GUI version, which looked like it was using the command based program to do the job. For example some software management application, like the graphical updater, are invoking a terminal command (apt) and then rendering the output in a more friendly way. Thus my decision was made. I was going to write a CLI based program and eventually I'd write a GUI interface on top of it.

I called it fixrhy on the assumption that it was fixing something that rhythmbox wasn't doing the way I wanted it. I know it's arguable but hey, my application, my name!

Ready to start my first ever python program I launched the Gnome text editor which I figured was more than enough for the task at hand.

The logic of the program is very simple. As input it requires the name of the artist, the name of the album, the sorting name of the artist and the sorting name of the album. After writing the basic code I added a fifth parameter which I'll talk about later. The output would be just some confirmation messages, while the real action is actually manipulating the xml file that rhythmbox uses to store its database. The full code of the final version is available here: https://pastebin.com/zBuhmi1y and I would like to go through it not to teach python or programming but to share the logic and its development. I will stop only on significant parts of the program.

Line 14 is:

```
#!/usr/bin/python
```

Technically this is needed to execute the code as a Python program. To me it is the statement that I wanted to put in practice something I learned on Full Circle pages, and knowing that if I need help there'll be a community out there ready to help. I could have used C (or C++) that I already knew, but that would have taken away a bit of the fun. Learning something new it is always challenging and it offers more opportunity of reaching out for others and live in the community. Case in point, I eventually wrote to Greg Walter and Ronnie Tucker (I'll come back to it later on).

Line 18 is:

```
import xml.etree.ElementTree as ET
```

This is required to be able to use the ElementTree library to parse XML files. I could have parsed it myself, but it would have been more work for no good reason. If tomorrow the XML specifications change or evolve you can also count on the library to update along. This means that there are great chances that your code will still work as it is. For that reason note that if you google "python xml" that ElementTree comes at the top in the official

Python documentation. That gave me the peace of mind that I was working with a fully supported library.

Line 22 is:

```
sys.setdefaultencoding( "UTF-8" )
```

If I recall correctly this became necessary after struggling with accented letters (as in Beyoncé) that are quite common in names.

Next I have to check if I have enough information to go on. Remember we said we need 4 pieces of data, 2 to identify an album (artist and album name) and 2 to set their respective sort values. The fifth parameter is option (to override existing sorting information). We do it at line 25:

```
if len(sys.argv) < 5:
```

Note that we check for 5 since the first value is always the script name (check https://docs.python.org/2/library/sys.html). Lines 26 to 30 print out a little help. I wrote them to be consistent with typical Linux programs, although in the beginning this script was mainly for me. I tried to think about others and follow standards everyone could relate to. Moving along you'll see that in lines 32 to 35 we save the input data and 37-40 are to check if there is the extra flag "force".

Line 42 and 43 looks like debugging lines:

```
print "Looking for", lk_alb, "by",lk_art

print "Sorting as", sr_alb, "by", sr_art
```

and indeed they were at the beginning. I kept them there because again I thought that tomorrow I (or someone else) could write a GUI program on top of it, and that could help if being parsed.

From 45 to 50 we set everything up, loading the Rhytmbox database into a tree (47) and declaring a couple of counting variables.

The real processing starts at line 52 where go through every entry in the tree. Here is just a simple sequence of checks, is it a song? Is it the right artist? Is it the right album?

```
  if entry.attrib == {'type': 'song'}:
    if entry.find('artist').text == lk_art:
      if entry.find('album').text == lk_alb:
```

If any of them is false we can discard that given entry since we don't need to amend it.

From line 58 we check if the artist sort information is already there. The reason being that if the field is missing we need to create it new (60), if it is already there we have to change it (or leave it unchanged). Former case we could not modify something not existing, latter case we should not create something that is already there.

The rest of the code builds on that, and there is a similar part of the album sort information. The only two points I want to make here are that I put in the code the parameter to choose if to modify existing data or not. Personally I would always change it, but I wanted to write a program that eventually could be used by others, maybe with different needs. Also note that since I couldn't change the element if existing I destroy and create a new one (68-70). Most of the print commands are there for debug originally and then to be used by GUI as I said before about line 42/43.

Finally before saving we make a backup copy (always important!):

```
shutil.copy2(filename,backup)
```

and the tree that has been so long in the memory goes in the file.

```
tree.write(filename, xml_declaration=True)
```

I am quite happy with my first Python program ever. It does what it says on the tin but there is room for much improvements. Amongst them surely pass the file position as a parameter (like this you should run the script in the folder with the Rhythmbox DB file). The parsing maybe not the most efficient and fast. The naming could be done automatically, for example a clever program could learn that "The Doors" should be "Doors, The" without entering it every single time. And many more, including having a GUI. And that's where we are heading to...

Write For Full Circle Magazine

Guidelines

The single rule for an article is that **it must somehow be linked to Ubuntu or one of the many derivatives of Ubuntu (Kubuntu, Xubuntu, Lubuntu, etc).**

Rules

• There is no word limit for articles, but be advised that long articles may be split across several issues.

• For advice, please refer to the **Official Full Circle Style Guide**: http://bit.ly/fcmwriting

• Write your article in whichever software you choose, I would recommend LibreOffice, but most importantly - **PLEASE SPELL AND GRAMMAR CHECK IT!**

• In your article, please indicate where you would like a particular image to be placed by indicating the image name in a new paragraph or by embedding the image in the ODT (Open Office) document.

• Images should be JPG, no wider than 800 pixels, and use low compression.

• Do not use tables or any type of **bold** or *italic* formatting.

If you are writing a review, please follow these guidelines :

When you are ready to submit your article please email it to: articles@fullcirclemagazine.org

Translations

If you would like to translate Full Circle into your native language please send an email to ronnie@fullcirclemagazine.org and we will either put you in touch with an existing team, or give you access to the raw text to translate from. With a completed PDF, you will be able to upload your file to the main Full Circle site.

Reviews

Games/Applications

When reviewing games/applications please state clearly:

- title of the game
- who makes the game
- is it free, or a paid download?
- where to get it from (give download/homepage URL)
- is it Linux native, or did you use Wine?
- your marks out of five
- a summary with positive and negative points

Hardware

When reviewing hardware please state clearly:

- make and model of the hardware
- what category would you put this hardware into?
- any glitches that you may have had while using the hardware?
- easy to get the hardware working in Linux?
- did you have to use Windows drivers?
- marks out of five
- a summary with positive and negative points

You don't need to be an expert to write an article - write about the games, applications and hardware that you use every day.

Zorin OS

I've been a Linux fan for close to two decades and have used Ubuntu derivatives for a good chunk of that time. Some of you may even remember I wrote a few reviews here and there for this magazine some time ago.

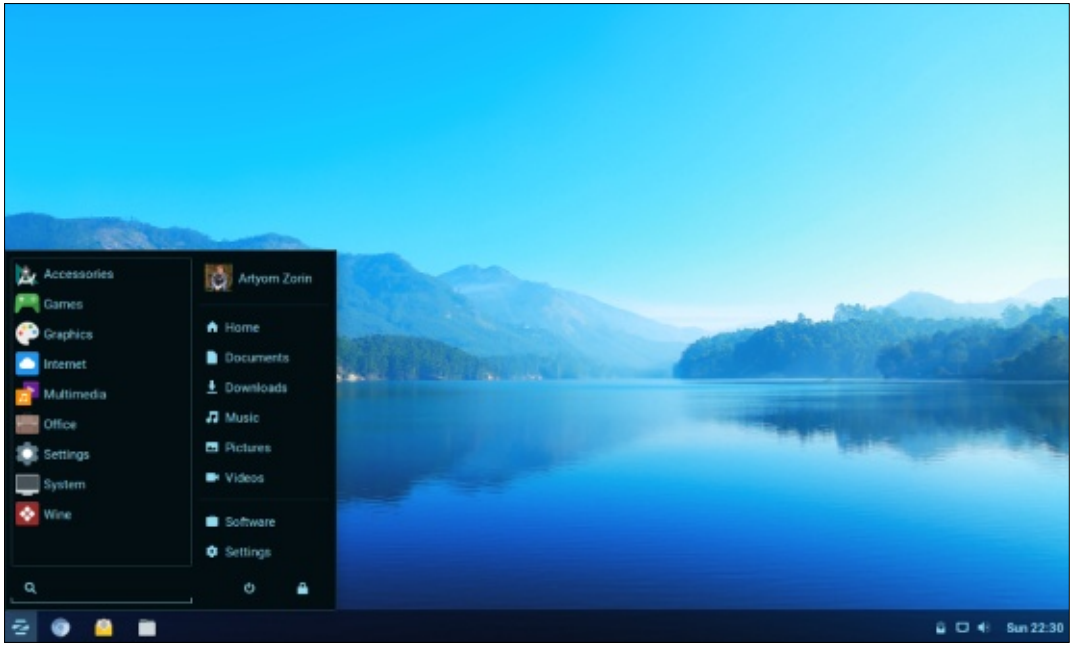
In my OS trials I've come across some oddities such as ArtistX (take everything in the store and ram it on a DVD and then wonder why most of it didn't work), PearOS (made it look so much like Apple's OS X they were seen by some as a Hackintosh substitute), Ultimate Edition (a lot of pizzazz mixed with a bit of gee-wizardry and dark undertones to create a hodgepodge that's hard to describe), and UberStudent (a top notch educational offering that should have worked but just never gained much of a following).

My major problem then became gratification, or lack thereof. Everybody was so content to be like Ubuntu they just stopped trying and merely added programs galore without addressing shortcomings or improving much of anything.

Then came Linux Mint and I didn't look back, especially when Ubuntu went to that awful Unity desktop and started losing market share. It comes with everything I need and little I don't. Even has a neat update feature for dummies like me that forget to check every once in a while.

However, wanderlust got the best of me again so I decided to look at other offerings to play with. ArtistX went bye-bye a few years ago, PearOS (was bought out, changed their name to Pearl OS or Pearl Linux – even they aren't sure, and became another Ubuntu clone), Ultimate Edition defected to KDE but is still as dark as ever, and UberStudent wasn't uber enough and all but stopped updating nearly 3 years ago (and then the website vaporized, too).

Yet others, like base Ubuntu and derivatives that start with K or L, were just too plain. I want an OS that's different, and I'm not talking Ubuntu with flashy wallpapers.



In my research I came across Zorin OS, an Irish offering I spotted a couple of years ago and tried. Its major claim to fame was a switch that allowed users to change the desktop menu and scheme from Ubuntu based to Windows or Apple lookalikes.

Yippee (all lack of enthusiasm intended).

Otherwise, it wasn't anything much.

I guess they got the hint and decided to beef up the "try me" incentives by pulling the Linux version of the automotive shell game (you know, Acura is a gussied-up Honda, Infiniti is Nissan and Toyota cranks out the Lexus line – all at shockingly higher prices than the base models they come from).

Like your Linux websites short and to the point? Prefer not to have them resembling the NYC Yellow Pages?

You've come to the right place at www.zorinos.com. Nothing but the facts here, ma'am. Not even a good gallery, in my opinion.

In short, you get your choices, what they include, how much they cost and what you'll need in the way of computing power.

As for the basics, you'll need a computer running:

- Minimum single core processor with a 1GHz processor.
- At least 1GB RAM.
- 10GB free HDD space for all versions but Ultimate and it requires 20GB.
- Resolution of at least 800x600.

Lite versions of Zorin drop these requirements to a 700 MHz processor, 512 RAM, 8GB free HDD space and 680x400 resolution.

In short, about any PC sold by your local big box store within the past decade; however, my guess is you'll want more horsepower under the hood if you're running their beefier offerings such as Business or Ultimate.

As for versions you'll find the following (all in 32 or 64 bit offerings):

Lite. Made for older computers, this is free to download and comes in at 1.4GB. As the name implies, it's Ubuntu basic in what it offers.

Core. At 1.5GB it includes a few more programs but is still free and can run well on newer computers.

Business Lite. Contains all the Core programs plus a few more geared to the business owner with older computers. Runs €15 per computer, or slightly less than $19USD when I wrote this. It comes at around 2.4GB.

Business. Contains everything in Business Lite plus a few other programs thrown in. Slightly over 2.5GB and costs the same per computer as Business Lite.

Education. Geared, obviously, to the younger student crowd (many of the apps seem to cater to the pre-high schoolers), this version comes with math and science programs. Free to download and comes in around 2.5GB.

Education Lite. You guessed, a smaller version of above running about 2.4GB. Still free.

Ultimate. As the name implies, this is the granddaddy of them all but the 3.6GB download also includes tech support in the price of €19 or a little less than $24USD.

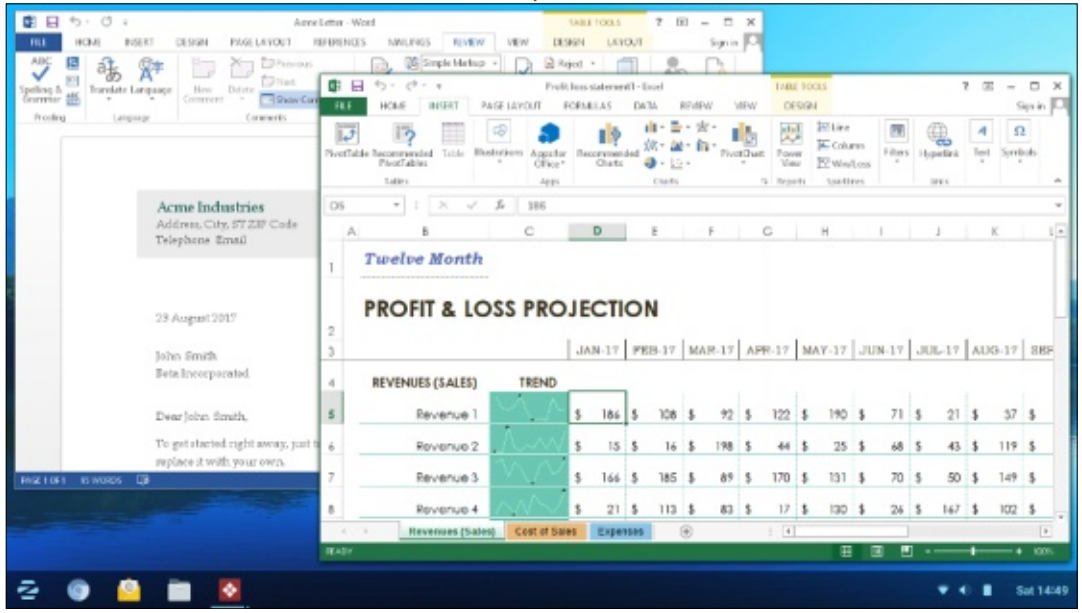
For those on a budget, like me, you can opt to go to www.osdisc.com where a DVD copy will set you back about $8 with shipping (the pay a cut to Zorin); however, beware of Softpedia.com where they advertise it for €9.99 but then link you to the Zorin page where it jumps up to €19. It turns out Zorin doubled the price but somebody at Softpedia didn't get the memo.

As for me, I merely borrowed a DVD from another user.

I opted to try mine on all old 80GB HDD I had left over from my Windows 7 days and installation wasn't anything painful or slow. In fact, from start to finish took the usual 20 minutes or so and I imagine that time would have been cut substantially had I been using a USB drive to SSD instead.

Once installed, it does a nice job of booting up within 20 seconds, surprising considering I'm using it on my 7-year-old Dell e6430 workhorse and the HDD is an "slow" spinner at 5,400 RPM.

I was pleasantly surprised to find everything worked out of the box; however, there was a



disturbing lack of wireless notification. No icon and no nag message stating networks were in the area.

Turns out it's there, just playing hide-and-go-seek. You must click the taskbar area near the clock and it'll bring up the wireless sub-menu.

Gee, and this is a beginner's introduction to Linux over Windows? Newbies want and expect an icon. Come to think of it, so do I.

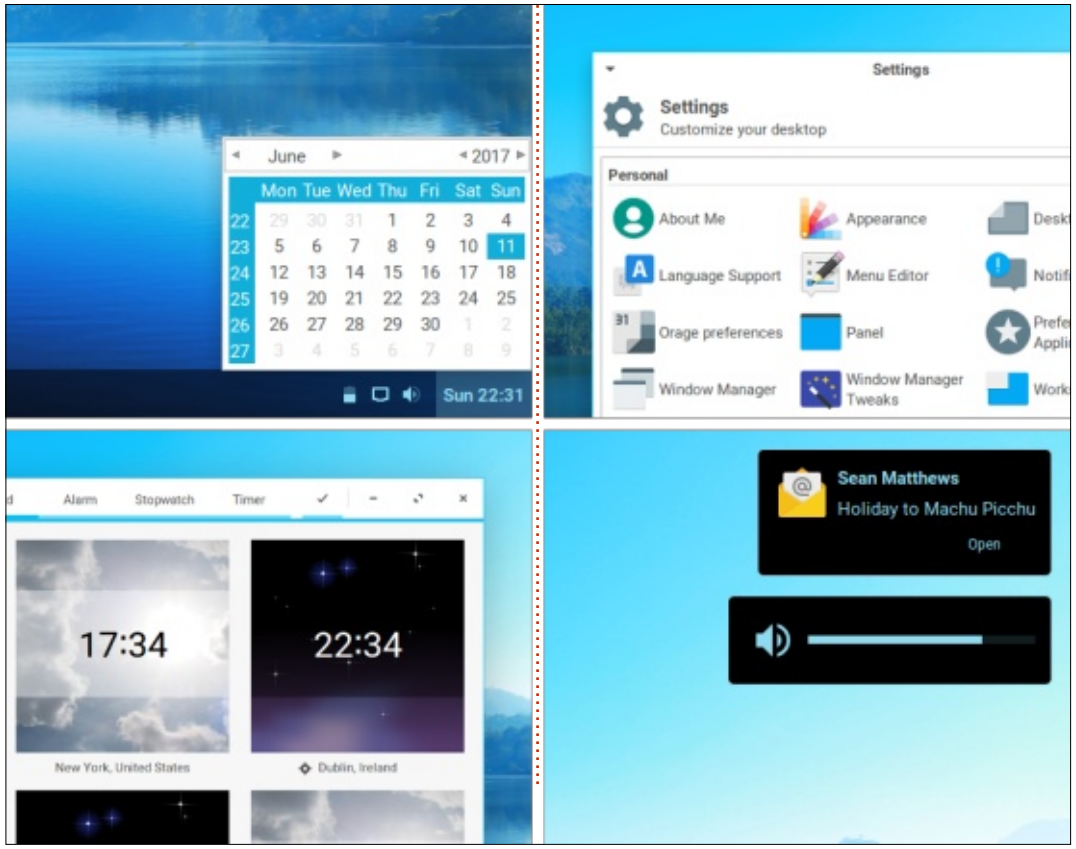
For those transitioning from early Windows (XP through 7) to Linux, you might appreciate Zorin's attempt to make users feel at home. The desktop exudes minimalism and functionalism at the same time. Plus, there are options under settings for those desiring a little more visual zest (at least in paid versions). Click on the desktop switcher and choose between Mac OS, Gnome, Unity or Windows 7.

As you might have figured on your own, the Gnome versions put the menu option in a corner, Unity crams everything to the left, Mac OS gives you the bottom, center mounted, icon bar and Windows 7 gives you the Windows menu in the corner. By default, it's usually Gnome.

Then there's the design I can't figure out. You get an icon bar to the right and yet another to the left. When I clicked on it absolutely nothing came up. Guess it's a work in progress or you're supposed to do something I'm unaware of.

There is another fault here, though. The switcher doesn't tell you what the designs are meant to represent. You had better know what Mac OS, or any of the others, look like or you'll be stabbing in the dark.

And this brings up a point an observer mentioned. Since all new Windows computers come with 10 and 7 effectively ended a few years ago, why would anybody transitioning really want the 7 design unless that's the last one they have fond memories of?



For those looking for a little visual pizzazz, there is an option to have a live desktop; however, Zorin offers just one design and you'll have to go fishing for the rest (if you go to their website, it's the one swirling about there). If you do opt for it, be prepared to have a RAM usage spike.

As for programs you get the usual inclusions such as LibreOffice (which I upgraded to 6) but you'll also find:

In the Audio/Visual department: Mixxx DJ, Blender 3D, GIMP Image Editor, LMMS (digital audio), Kdenlive Video editor, MyPaint, Ardour Audio Workstation, Audacity Sound Editor, Inkscape (Vector Graphics), Builder (for compiling desktop apps), Kodi Media Center, VLC Media Center, Photos (now with better online cooperation with Facebook and Google Photos), Cheese, Brasero and a few others.

For the office crowd you'll find FreeCAD, HomeBank (a nice Quicken sub), VYM (mind mapping), Planner (project planning), Calendar (decent Google interactive substitute),

Referencer (I've used this in the past to log references and produce bibliographies), PDFMod, Xournal (note taker and personal daily log), CellWriter Dia, LibreCAD and GnuCash. I purposely separated the last two because they are effectively repeats of CAD and financial programs mentioned earlier.

For the gaming set: Super Tux Kart, Xonotic (arena shooter), Neverball (one of my personal favorites), Neverputt, Pingus, Frets on Fire, Gweled and Warzone 2100. I probably missed a couple but you get the idea.

And for those looking for the internet experience, you'll find Chromium (not Chrome, though), Maps (a Google Earth lookalike also known as Gnome Maps), and Weather (the new Gnome 3 version, I take it)

Finally, for those who can't quite wean themselves off the Windows practice, you'll find Wine and Play on Linux.

In general, daily operation I found Zorin to be okay but with a few quirks.

First, some taskbar icons seem to have an agenda of their own. The wireless icon was adept at disappearing and reappearing at will along with others like Dropbox. Mind you, the wireless still worked and I'll assume so did Dropbox, but the icons weren't there.

Second, the software store is a hit or miss proposition with the latter being truer than the former. Often it would open and attempt to install programs of my choice and just stall. Not only that, it would then lock up my system requiring a force quit or restart.

Even when it did work the result appeared laborious with downloads getting off to a grand start and then trickling down to 1990s modem standards or just dying. After a while it just became frustrating and I'd go back to the command line to install what I wanted.

However, I do have to give praise to the store design. It's intelligently designed with a layout a few other Linux offerings could take a clue from. Even a newbie could figure it out.

Third, and finally, when it worked Zorin was indeed speedy with programs generally opening in a snap; however, it was prone to bogging down at the most inopportune times. For example, browser pages would occasionally not open and instead post a page about not being there only to open a couple seconds later.

I even thought this might be a Chromium issue and installed Opera, a browser I've found speedier than Chrome. Nope, that didn't help. Same issue would crop up on a random basis and not under the same circumstances.

Oh, if you're wondering, it's not RAM overload, either. I'm pushing 6GB DDR3 so an open website shouldn't bother the overall operation.

Otherwise, Zorin is just another Ubuntu derivative loaded with programs you might want and some you won't (such as the repeats you may have noticed in the listings above).

In fact, here's an idea – download Linux Mint with its predictable, yet reliable, daily operations. Then add any of the programs I listed.

Not only do you get a stable system, you just saved yourself a few bucks!

And, therefore, I can give Zorin OS Ultimate no more than 3 out of 5 stars. For the price requested it should work much better than it does.

If you are interested, I'd advise you start with one of the free versions first and work your way up.

LETTERS

If you would like to submit a letter for publication, compliment or complaint, please email it to: letters@fullcirclemagazine.org. PLEASE NOTE: some letters may be edited for space.

32-BIT TNG?

Like many, I am still running a couple of 32 bit machines. They work fine for what I need them to do (browsing, email, and the like, no gaming, no intensive graphics). They also make good machines for people unable to afford the latest and greatest. There is no reason (yet at least) to stop using them. I want to prepare for 2020/2023 when it seems there will be no or few 32 bit Linux OSs around and limited if any support/updates.

With the recent announcements of suspension of 32 bit Linux distros, what is a person like me to do??? Is there anyone around that could write an article or two on how to prepare for this event. Perhaps reviewing/suggesting existing 32 bit OSs that may survive and discuss what will happen to programs (sorry applications) as well. Might be an interesting read. Sorry I am not knowledgeable enough to write such a piece.

John

Ronnie says: *While I can't vouch for most of the distros in this post from Fossbytes (https://fossbytes.com/best-lightweight-linux-distros/) it mentions some really good 32-bit friendly distros.*

FULL CIRCLE 2018 SURVEY

It's that time of the year again where we ask what you think of FCM, Ubuntu, and Linux.

Some questions are a requirement, some you can skip over if not applicable.

Your answers will help shape Full Circle, so please use your constructive criticism. If you don't tell us what you think, or what we're doing wrong, then we won't know.

Survey URL:
http://bit.ly/fcm2018

Join us on:

goo.gl/FRTMl

facebook.com/fullcirclemagazine

twitter.com/#!/fullcirclemag

linkedin.com/company/full-circle-magazine

ubuntuforums.org/forumdisplay.php?f=270

FULL CIRCLE NEEDS YOU!

Without reader input **Full Circle** would be an empty PDF file (which I don't think many people would find particularly interesting). We are always looking for articles, reviews, anything! Even small things like letters and desktop screens help fill the magazine.

See the article **Writing for Full Circle** in this issue to read our basic guidelines.

Have a look at the last page of any issue to get the details of where to send your contributions.

Q&A

Compiled by Gord Campbell

If you have a Linux question, email it to: misc@fullcirclemagazine.org, and Gord will answer them in a future issue. Please include as much information as you can about your query.

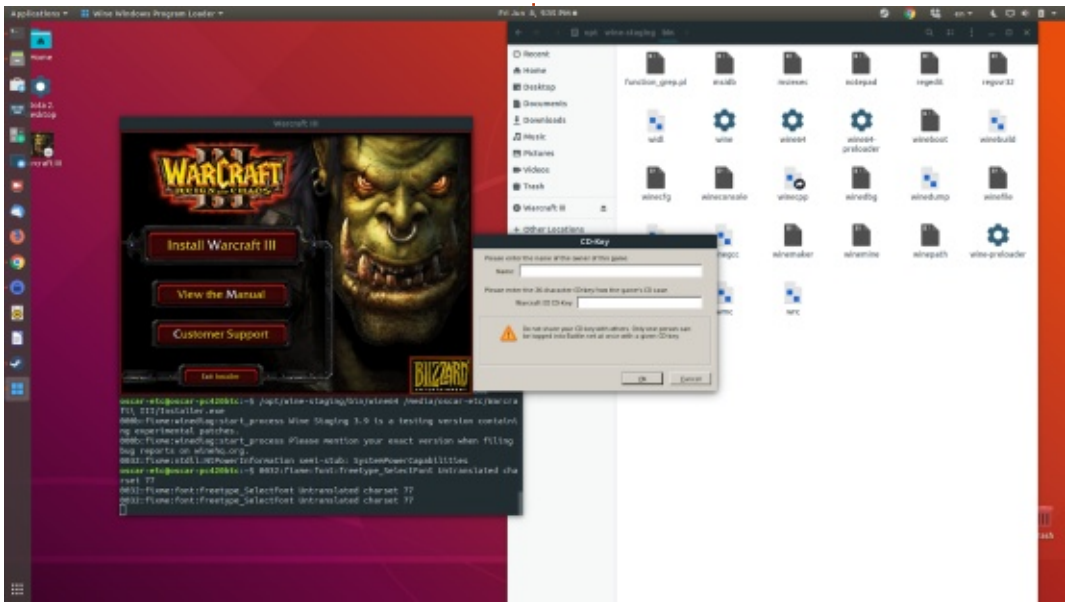
Unfortunately, Gord is leaving us (due to health reasons) and this may mean the end of Q&A!

If you'd like to take over, and fill his extra large shoes, drop an email to: ronnie@fullcirclemagazine.org and say where you'd like to take this column. Changes? Keep it the same? Let me know. But, obviously, you need to be able to do it every month.

Warcraft III

In many ways I feel I've been spoiled in that my entire experience with Linux has been very graciously blessed with an overly-abundant plethora of video games. The whole time I've been using Linux there have been more than enough video games with which to divert my worries, challenge my reflexes, relieve my stress and inspire me to keep creating the same art that has kept my heart beating since birth. I started using Ubuntu back in 2010 and ever since then, there's been an ever-growing flow of Linux-native games released. Because of such blessing, I've never had the need to rely on simulators. However, things weren't always that way. In fact, even today, the amount of games available on Linux is perhaps less than 1% of the number of games available on Windows. A lot of those games can actually be played on Linux via Wine (Wine Is Not an Emulator). So, I've decided to finally give it a try at playing a video game through an emulator. Recently, a friend of mine suggested that I try to play World of Warcraft and review it, which I'll probably do in the near future but for now I've gone ahead and installed Warcraft III: Reign of Chaos and to my delight, I've had some pretty rewarding results. Warcraft III is a fantasy, real-time-strategy game. Intended to be played on a computer, Warcraft III can only be played with mouse/keyboard. So, let's get on with it. Without going into too much detail (there are numerous how-to-guides available on the internet) I'll briefly go over a few basic steps needed to install and play Warcraft III in a modern version of Ubuntu & its derivatives (the steps I've followed should work with most releases after Ubuntu 14.04).

Originally released in 2002, Warcraft III is the third game from the Warcraft game series. The Frozen Throne, an expansion pack was released the next year, in 2003. You can buy Warcraft III directly from Blizzard's battle.net shop for $9.99 if you buy only the game without the expansion pack. This game has been such a big hit over the years that Blizzard is still actively updating it. The latest patch (version 1.29) for Warcraft III was actually released in April 2018 which is pretty impressive for a 16 year old game. I was lucky enough to find the Warcraft III Battle Chest in Amazon for the low price of $6.99 in new condition. The Battle Chest includes Warcraft III: Reign of Chaos CD, Warcraft III Expansion: The Frozen Throne CD and two bonus strategy guides from Brady Games. I was able to install the game from these CDs.



There are plenty of guides on how to get Warcraft III running on Linux. However, with the new release of Ubuntu 18.04, there are a few hurdles which I had to jump in order to first install and then to run the game. I'll briefly go over these crucial steps which no single guide will include. Let's first talk about problems you may run into if you're using the latest version of Ubuntu and a possible solution which may work for you. The first problem I encountered dealt with installing wine. According to most of the up-to-date content I found regarding the installation of Warcraft III on Linux, a regular wine version will not suffice.

Instead, it is recommended to install wine-staging which is the newest version of wine, or as it's called on the winehq.org website, the Staging branch. Unfortunately, at this time there isn't yet an official release key for winehq on Ubuntu 18.04 so instead I had to do a bit of a runaround. I basically tricked my OS into using the Ubuntu 17.10 version of the winehq release key.

So, the steps I followed are these:

```
$ sudo dpkg --add-architecture i386

$ wget -qO- https://dl.winehq.org/wine-builds/Release.key | sudo apt-key add -

$ sudo apt-add-repository 'deb http://dl.winehq.org/wine-builds/ubuntu/ artful main'
```

It was at this step where I first ran into problems because the release key that we're adding is not yet an official release for Ubuntu 18.04 but it's very likely that by press time this is no longer an issue. So, let's review: to install Warcraft III on Linux it's recommended to use wine-staging instead of default wine.

Unfortunately, for Ubuntu 18.04 there is no official repository key for wine-staging so you may need to add it to /etc/apt/sources.list

At the following line, change it from bionic to artful and that should do the trick:

```
deb https://dl.winehq.org/wine-builds/ubuntu/ bionic main
```

It should now look like this:

```
deb https://dl.winehq.org/wine-builds/ubuntu/ artful main
```

OK, now we're ready to continue without interruption as follows:

```
sudo apt-get update apt-get install --install-recommends winehq-staging
```

Most of the steps I followed came from the following video: https://www.youtube.com/watch?time_continue=3&v=uPPyBkq_fNg.

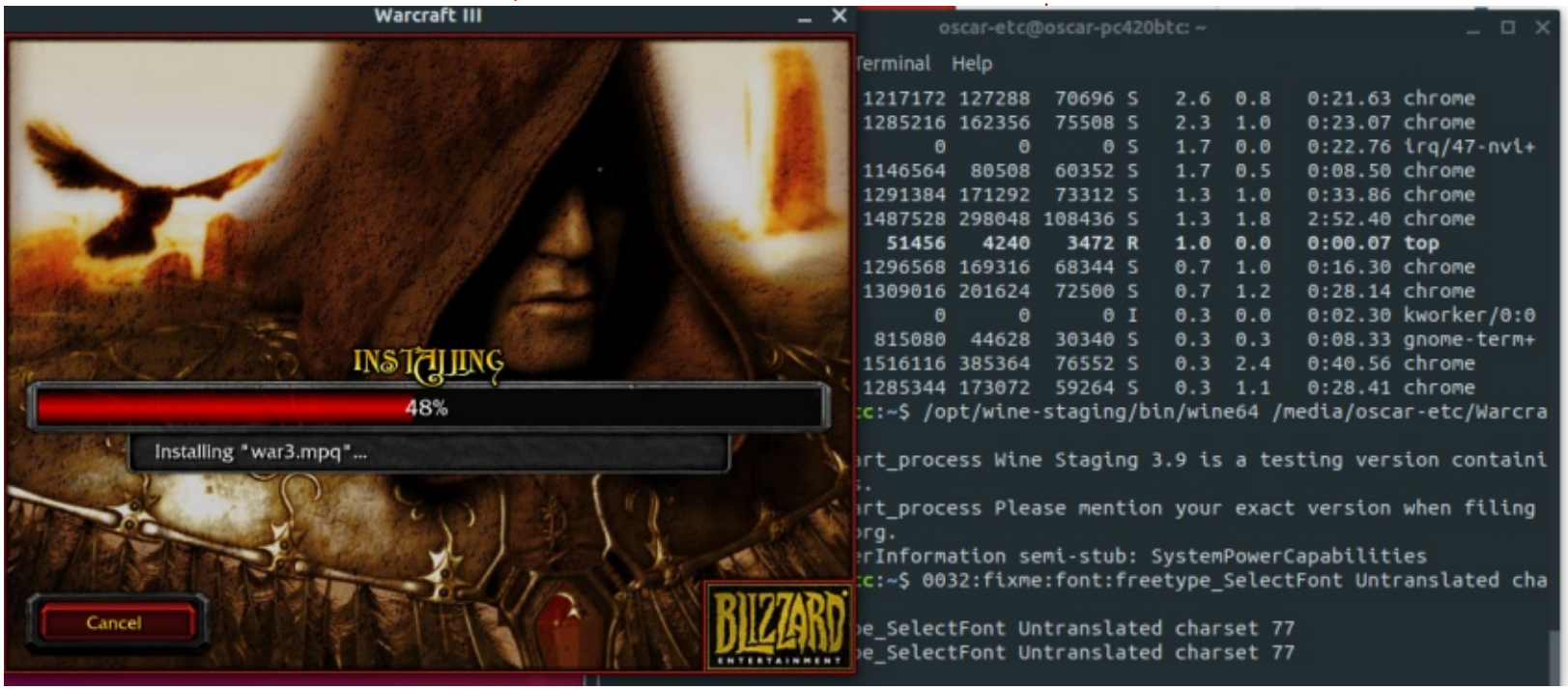
Now that we've got wine installed we can continue with the installation of the game. This part should be easy. Just pay attention as to the location of the installation files for Warcraft III, especially the Warcraft.exe file. In my case, since I was installing straight from disc, the location for me was:

```
/media/username/WarcraftFolder/Warcraft.exe
```

The steps to follow are these:

```
/opt/wine-staging/bin/wine64 /path/where/you/downloaded/the/wc3/installer
```

Notice the space after wine64 and before /path and it should go without saying that the path where you downloaded the wc3 installer should specify the real location of where the installer is.



So this leads us to the final part of the installation which should be pretty automatic, if you have any questions at this point is when you can watch the video.

You might have to register the game by entering the CD key provided. Then continue with the installation by clicking OK on all of the screens that follow. Always read before clicking OK just in case something else unexpected comes up.

Once I had the game installed I kept running into problems when trying to run the game for the first time. The main problem was that although I could hear the game running, I couldn't see it. Instead, all I could see was my desktop and whatever other windows I had open on such desktop. Then I found out that the game works best in linux/wine with opengl in native Full Screen so to launch it I had to run the following in a terminal:

```
$ wine
~/.wine/drive_c/Program\
Files\ \(x86\)/Warcraft\
III/War3.exe -opengl
-nativefullscr
```

Having run that command I was finally able to successfully launch the game and I've had zero problems with it ever since.

Next month we'll finish this with the actual review of the game

Oscar graduated from CSUN, is a musician, game enthusiast and has been working with Bitcoin and other alt-coins. You can follow him at: https://twitter.com/resonant7hand or email him at: 7bluehand@gmail.com

PATRONS

Monthly Patrons

2016 - Present:
Bill Berninghausen
Jack McMahon
Linda P
Remke Schuurmans
Norman Phillips
Tom Rausner
Charles Battersby
Tom Bell
Oscar Rivera
Alex Crabtree
Ray Spain
Richard Underwood
Charles Anderson
Ricardo Coalla
Chris Giltnane
William von Hagen
Mark Shuttleworth
Juan Ortiz
Joe Gulizia
Kevin Raulins
Doug Bruce
Pekka Niemi
Rob Fitzgerald
Brian M Murray
Roy Milner
Brian Bogdan
Scott Mack
Dennis Mack
John Helmers
JT
Elizabeth K. Joseph
Vincent Jobard
Joao Cantinho Lopes
John Andrews

2017 - Present:
Matt Hopper
Jay Pee
Brian Kelly
J.J. van Kampen

2018 - Present:
John Helmers
Kevin O'Brien
Kevin Raulins
Carl Andersen
Charles Stewart
Dave Nelson
Brian Bogdan
Dennis Shimer

Single Donations

2018:
Yvo Geens
Graig Pearen
Carlo Puglisi
James A Carnrite
John Holman
P G Schmitt
Robert Cannon
Thomas A Lawell
Ronald Le Blanc
Luis Eduardo Herman
Glenn Heaton
Peter Swentzel
Alain Mallette
Christophe Caron
Linda Prinsen
Ronald Eike
Anthony Cooper
Louis W Adams Jr
Joseph Tong
Robert G. Wells
Robert Kaspar
Thomas Gambier
Peter Fitzsimons
Terry O'Neill

The current site was created thanks to **Lucas Westermann** (Mr. Command & Conquer) who took on the task of completely rebuilding the site, and scripts, from scratch, in his own time.

The Patreon page is to help pay the domain and hosting fees. The yearly target was quickly reached thanks to those listed on this page. The money also helps with the new mailing list that I set up.

Several people have asked for a PayPal (single donation) option, so I've added a button to the right side of the website

A big thank you to all those who've used Patreon and the PayPal button. It's a HUGE help.

https://www.patreon.com/fullcirclemagazine

https://paypal.me/ronnietucker

https://donorbox.org/recurring-monthly-donation

HOW TO CONTRIBUTE

FULL CIRCLE NEEDS YOU!

A magazine isn't a magazine without articles and Full Circle is no exception. We need your opinions, desktops, stories, how-to's, reviews, and anything else you want to tell your fellow *buntu users. Send your articles to: articles@fullcirclemagazine.org

We are always looking for new articles to include in Full Circle. For help and advice please see the **Official Full Circle Style Guide**: http://bit.ly/fcmwriting

Send your **comments** or Linux experiences to: letters@fullcirclemagazine.org
Hardware/software **reviews** should be sent to: reviews@fullcirclemagazine.org
Questions for Q&A should go to: questions@fullcirclemagazine.org
Desktop screens should be emailed to: misc@fullcirclemagazine.org
... or you can visit our **site** via: fullcirclemagazine.org

FCM#135

Deadline:
Sunday 08th July 2018.
Release:
Friday 27th July 2018.

Full Circle Team

Editor - Ronnie Tucker
ronnie@fullcirclemagazine.org

Webmaster - Lucas Westermann
admin@fullcirclemagazine.org

Editing & Proofreading
Mike Kennedy, Gord Campbell, Robert Orsino, Josh Hertel, Bert Jerred, Jim Dyer and Emily Gonyer

Our thanks go to Canonical, the many translation teams around the world and **Thorsten Wilms** for the FCM logo.

Getting Full Circle Magazine:

For the Full Circle Weekly News:

You can keep up to date with the Weekly News using the RSS feed: http://fullcirclemagazine.org/feed/podcast

Or, if you're out and about, you can get the Weekly News via Stitcher Radio (Android/iOS/web): http://www.stitcher.com/s?fid=85347&refid=stpr

and via TuneIn at: http://tunein.com/radio/Full-Circle-Weekly-News-p855064/

EPUB Format - Most editions have a link to the epub file on that issue's download page. If you have any problems with the epub file, email: mobile@fullcirclemagazine.org

Issuu - You can read Full Circle online via Issuu: http://issuu.com/fullcirclemagazine. Please share and rate FCM as it helps to spread the word about FCM and Ubuntu.

Magzster - You can also read Full Circle online via Magzster: http://www.magzter.com/publishers/Full-Circle. Please share and rate FCM as it helps to spread the word about FCM and Ubuntu Linux.