

The PCLinuxOS magazine

Volume 148

May, 2019



De-Googling Yourself Part 1

*Short Topix: Julian Assange
Taken Into Custody*

*GIMP Tutorial:
More About Masks*

Casual Python, Part 4

*The Ruby Programming
Language: Writing
A Ruby Program, Part 2*

*Repo Review:
Deja Dup Backup*

FREE Linux Help Books

*Tip Top Tips: Linking Your
Orage Calendar To
Google Calendar*

*ms_meme's Nook:
If I Had PCLinuxOS*

And More Inside ...

In This Issue...

- 3 *From The Chief Editor's Desk...*
- 4 *De-Googling Yourself, Part 1*
- 8 *Screenshot Showcase*
- 9 *ms_meme's Nook: Boot Up PCLOS*
- 10 *Short Topix: Julian Assange Taken Into Custody*
- 14 *Screenshot Showcase*
- 15 *GIMP Tutorial: More About Masks*
- 17 *Tip Top Tips: Linking Your ORAGE Calendar To Google Calendar*
- 19 *Screenshot Showcase*
- 20 *Casual Python, Part 4*
- 28 *PCLinuxOS Recipe Corner: Philly Cheese & Ground Beef Casserole*
- 29 *Screenshot Showcase*
- 30 *Repo Review: Deja Dup Backup*
- 31 *PCLinuxOS Family Member Spotlight: jim2u71*
- 32 *Screenshot Showcase*
- 33 *The Ruby Programming Language: Writing A Ruby Program, Part 2*
- 40 *Screenshot Showcase*
- 41 *FREE Linux Help Books*
- 44 *PCLinuxOS Bonus Recipe Corner: Cheesy Ground Beef Manicotti*
- 45 *ms_meme's Nook: If I Had PCLinuxOS*
- 46 *PCLinuxOS Puzzled Partitions*
- 50 *More Screenshot Showcase*

The PCLinuxOS magazine

The PCLinuxOS name, logo and colors are the trademark of Texstar.

The PCLinuxOS Magazine is a monthly online publication containing PCLinuxOS-related materials. It is published primarily for members of the PCLinuxOS community. The magazine staff is comprised of volunteers from the PCLinuxOS community.

Visit us online at <http://www.pclosmag.com>

This release was made possible by the following volunteers:

Chief Editor: Paul Arnote (parnote)

Assistant Editor: Meemaw

Artwork: Sproggy, Timeth, ms_meme, Meemaw

Magazine Layout: Paul Arnote, Meemaw, ms_meme

HTML Layout: YouCanToo

Staff:

ms_meme

Meemaw

Gary L. Ratliff, Sr.

Daniel Meiß-Wilhelm

daishi

Alessandro Ebersol

CgBoy

YouCanToo

Pete Kelly

phorneker

Khadis Thok

Smileeb

Contributors:

davecs

The PCLinuxOS Magazine is released under the Creative Commons Attribution-NonCommercial-Share-Alike 3.0

Unported license. Some rights are reserved.

Copyright © 2018.



From The Chief Editor's Desk...

Apologies are in order. The magazine is later than "normal" this month. I have been dealing with some "issues" at home, which has consumed considerable time that I would have normally had for getting magazine business done.

We're fortunate where I live that the weather has been relatively mild thus far. We've only had a handful of days that were hot and humid, as is typically the case. But, on one of those days, we found out that our aging central air conditioner had failed. Central air conditioning units are designed to last 15 to 18 years. Mine is 21 years old. It's definitely time to be replaced. That's issue number one.

I contacted my trusted HVAC guy to get things rolling with replacing the failing and aging AC unit. It would require some finagling of the existing duct work, along with a new compressor unit outside, and a new condenser coil on the inside. The initial quote for replacement was \$3,400. That's not a bad price for a 3 ton central air unit.

Of course, I didn't (and don't) just have that much money just lying around. So, I had to contact my bank to procure a personal line of credit that would cover the

replacement cost. That took a little bit of time to process. Banks never seem to be in much of a hurry.

So, with the personal line of credit from my bank in hand, I contacted my HVAC guy again to see about getting started on the replacement of the central A/C unit. He then informed me that he noticed some asbestos tape used along the joints of the ductwork, and that asbestos had to be professionally removed.

This had to be done before he could do the necessary modifications on the existing ductwork that the new unit required. That asbestos remediation is going to cost another \$400. Asbestos is some NASTY, NASTY stuff, and inhaling its fibers can lead to mesothelioma, which is a form of lung cancer. No one wants or needs that! Typically, the asbestos doesn't pose a problem until and unless it is disturbed ... which my HVAC guy will have to do to make the necessary ductwork modifications. And that is issue number two.

Add in a few school functions at my son's school, plus setting aside some time to celebrate Easter with the kids, plus working full time at the hospital, plus teaching a neonatal resuscitation program skills lab (I'm an instructor), plus being Mr. Mom on the days I'm off (my wife and I work opposite days so one of us is always home with the kids), and my free time that I usually have for magazine duties filled up quickly.

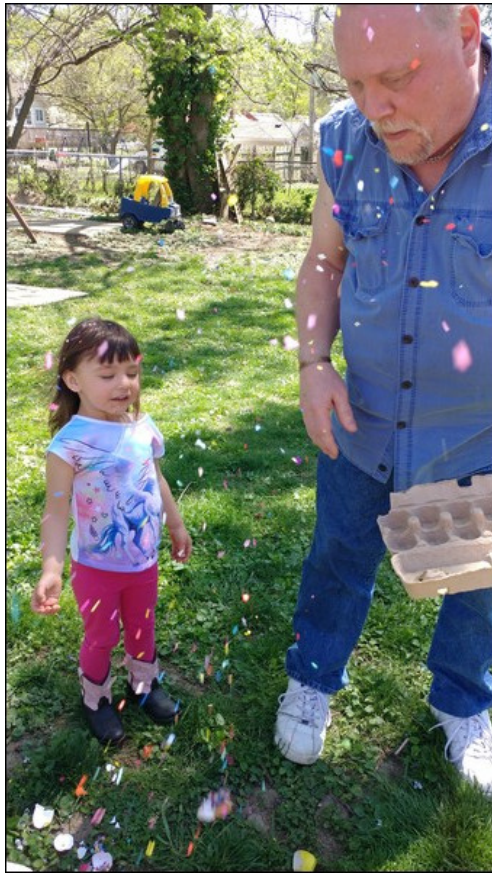
So, I am the reason that the magazine is released a bit late this month. I simply ran out of time with all of the extra things requiring my attention. In fact, at times, it all seemed a bit overwhelming.

Until next month, I bid you peace, happiness, serenity and prosperity. And here's hoping that the next month isn't quite as busy or eventful.



**The PCLinuxOS
Magazine**

**Created with
Scribus**



It's easier than $E=mc^2$
It's elemental
It's light years ahead
It's a wise choice
It's Radically Simple
It's ...

PCLinuxOS
Radically Simple

De-Googling Yourself, Part 1

by Agent Smith (Alessandro Ebersol)



I want to start this article with a joke ... Yeah, it's kind of old, but, more current than ever.

Google Pizza

- Hello! Gordon's Pizza?
- No sir, is Google's Pizza.
- So, wrong number? How pity.
- No, sir, Google bought Gordon's Pizza, now it's Google's Pizza.
- Okay then, I'll make my request, shall I?
- You want the usual?
- The usual? You know me ?
- According to our call record, in the last 12 times, you asked for sausage pizza, lots of cheese, thick crust.
- OK! That's right!
- May I suggest to you this time ricotta, arugula with dried tomatoes?

- What? I hate vegetables.
- Your cholesterol is not good, sir.
- How do you know?
- We cross the number of your fixed line with your name, through the Subscribers List.
- We have had the results of your blood tests in the last 7 years.
- Okay, but I do not want this pizza! I'm already on medication ...
- Sorry, but you have not been taking the drug regularly, from our commercial database four months ago, you just bought one box containing 30 anti-cholesterol tablets on the Drugsale Network.
- I bought more from another pharmacy.
- It does not appear on your credit card records.
- I paid with money.
- But you did not withdraw that much money, according to your bank's records.
- I have another source of money.
- This does not appear on your last income tax, unless you have purchased them from an undeclared source of income.
- WHAT THE HELL?!?
- Sorry, sir, we use this information only with the intention of helping you.
- Enough! I'm sick of Google, Facebook, Twitter, WhatsApp. I'm going to an island without internet, cable television, where there is no cell phone coverage and no one to see or spy on me.
- I understand, sir, but you will have to renew your passport first, since it expired five weeks ago.

So we start this series of articles, which will address how Google was born, its official (and unofficial) genesis, its dark links with American intelligence agencies, how Google tracks you online, and what are the alternatives for services that the company offers.

Of course, I do not want you to think there's hypocrisy here, since the article is running on a Google-owned site (Blogger). I'm also thinking of moving my blog to another platform, so do not accuse me of being a hypocrite. On a second thought, Google was one thing when it started, but over the years, either it became something else, or its true intentions became clearer. * This article was originally posted in a Blogger's blog, a service that belongs to Google, hence this explanation.

The Genesis of Google

Google was born when two Ph.D. students at Stanford University, Larry Page and Sergei Brin, came up with a search engine idea that would quickly find the most pertinent information in an ocean of data. They registered the Google.com domain name in 1997, and made it the Google company in 1998.

1997 was an important turning point for the search engine. The creators decided that there could be a change in the name of the 'googol' search engine, which has a mathematical as well as metaphorical meaning, as the term 'googol' indicates a large amount of numbers being represented.

The domain google.com was registered on September 15, 1997. They formally incorporated their company, Google, on September 4, 1998, into the garage of their friend Susan Wojcicki in Menlo

Park, California. Wojcicki eventually became an executive at Google and is now the CEO on YouTube.



The garage where Google was born

The first prototype of Google was Backrub, which in 1996, was a search engine that ran on computers at Stanford University and quickly outgrew its server, with lots of data. It worked long enough, however, to capture the attention of Sun Micro Systems co-founder Andy Bechtolsheim, and in 1998 he invested US\$ 100,000 in the yet to be formed Google company. Months later, the small garage startup was



Larry Page and Sergei Brin in 2003

chosen as the top search engine of PC Magazine in 1998. Just six years later, in 2004, the sale of Google's IPO gave it a market capitalization of over US \$23 billion.

The web crawler, BackRub, started exploring the web in March 1996, with Larry Page serving as the starting point. To convert the backlinks data it collected to a particular webpage in a rank of importance, Brin and Page developed the PageRank algorithm. When analyzing the output of the BackRub that, for a given URL, consisted of a list of backlinks ranked in order of importance, the pair realized that a PageRank-based search engine would produce better results than the existing techniques (essentially, the search engines of that time ranked the results according to the number of times the search term appeared on a page.)

Convinced that the pages with the most links to them from other highly relevant web pages should be the most relevant pages associated with the search, Page and Brin tested their thesis as part of their studies, and laid the groundwork for their search engine. The first version of Google was released in August 1996 on the Stanford website. It used almost half the bandwidth of the Stanford network.

In the words of its founder, Larry Page:

"Some rough statistics (from August 29, 1996)

Total indexed URLs: 75.2306 million

Total content transferred: 207,022 gigabytes

BackRub is written in Java and Python and works on several Sun Ultras and Intel Pentium running Linux. The primary database is maintained on a Sun Ultra II with 28GB of disk space. Scott Hassan and Alan Steremberg provided great help in a very talented implementation. Sergey Brin was also very involved and deserves many thanks."

IBM, Intel and Sun were the hardware sponsors for Sergey Brin and Larry Page. In addition, they used GNU/Linux, Java and Python as the foundation software for Google.



This is the server using only a 300 MHz Dual Pentium II, 512MB of RAM and 9GB of hard drive.

In March 1999, the company relocated its offices to 165 University Avenue in Palo Alto, home to several other notable Silicon Valley technology startups. After rapidly outgrowing two other locations, the company rented a complex of buildings in Mountain View at the 1600 Amphitheater Parkway, from Silicon Graphics (SGI) in 2003. The company has been located there ever since, and the complex has become known since as the Googleplex (a pun in the word googolplex, a number that is equal to 1 followed by a googol of zeros). In 2006, Google bought SGI's property for \$319 million.

By the end of 1998, Google had an index of about 60 million pages. The Google homepage was still marked "BETA," but an article on Salon.com already stated that Google's search results were better than those of competitors, such as Hotbot or Excite.com, and praised it for being more technologically advanced than the search engines such as Yahoo!, Excite.com, Lycos, Netscape's Netcenter, AOL.com,

Go.com and MSN.com, that back then, during the growing internet bubble, were seen as "the future of the Web," especially by stock market investors.



Google's home page in 1998

Earlier in 1999, Brin and Page decided they wanted to sell Google to Excite. They offered Excite's CEO, George Bell, to sell it (Google) for US \$1 million. He rejected the offer. Vinod Khosla, one of the major venture capitalists in Excite, convinced the pair down to \$750,000, but Bell still rejected it.

In 2002, Google began to further assist businesses with their Google Search Appliance and added the price per click to their Adwords.

In 2003, Google acquired Pyra Labs and announced Google AdSense. AdSense allows businesses to connect with large advertiser networks. Google also launched Google Grants, a non-profit edition of AdWords.

In 2004, on April 1st, Gmail was released, first an invitation-only service. Now, it has more than 425 million users. Google also acquired Picasa. In addition, Google listed itself in the stock market, offering the public 19,605,052 class A shares at US \$85 each. In December, Google established Google.org, which was dedicated to the idea that technology can change the world.



Googleplex building, Google's current headquarters

So were the first years of Google. And, this was the official story, the one we all know, the one in the books and magazines.

But ... it was not the whole story.

The history of Google, behind the history of Google ...

In the mid-1990s, the intelligence community in America began to realize that they had an opportunity. The supercomputer community was just beginning to migrate from universities to the private sector, led by investments from a place that would come to be known as Silicon Valley.

Information gathering may have been its business, but the Central Intelligence Agency (CIA) and the National Security Agency (NSA) had come to realize that their future would likely be shaped outside the government. It was at a time when the Clinton administration's military and intelligence budgets were in danger, and the private sector had vast resources at its disposal. If the intelligence community wanted to conduct mass surveillance for national security purposes, it would require cooperation between the government and emerging supercomputing companies.

In 1999, the CIA set up its own venture capital investment company, In-Q-Tel, to fund promising startups that could create useful technologies for intelligence agencies.



In-Q-Tel logo

And, here begins the obscure history of Google. In 1994, as we saw above, the two young Stanford University PhD students, Sergey Brin and Larry Page, made their discovery in the first automated web crawling and page ranking application. Brin and Page carried out their work with funding from the Digital Library Initiative, a multi-agency program of the National Science Foundation (NSF), NASA and DARPA. That is, even in the research phase, of what would become the Google search engine, there was already funding from American intelligence agencies.

In-Q-Tel, the CIA playing in the private sector.

Founded in 1999 as a way for the US to follow the rapid innovation in science and technology, In-Q-Tel has been an initial supporter of start-ups later acquired by Google (GOOG), Oracle (ORCL), IBM and Lockheed Martin (LMT).

While IQT originally served largely the CIA's needs, the company now supports many of the 17 agencies within the US intelligence community, including the National Geospatial-Intelligence Agency (NGA), the Defense Intelligence Agency (DIA) and the Department of Homeland Security Science and Technology.

The company acts in a semi-transparent way: In-Q-Tel issues a press release whenever it sponsors a new company, but does not disclose the amount of the investment nor the product on which it concentrates. It is believed that the relationship can lead to the development of off-the-shelf products specifically tailored for the CIA. A spokesman for an In-Q-Tel-funded company told Forbes that its investment focused on a specific project with a one-year deadline, refusing to provide further details.

Many companies listed on In-Q-Tel's investment site page are secret. According to the Washington Post, "Virtually any American businessman, inventor or research scientist working on ways to analyze data probably received a phone call from In-Q-Tel, or at least was surveyed by its team of technology watchers."

Here is a list of companies and projects maintained by in-Q-Tel (at least those visible to the public): <https://en.wikipedia.org/wiki/In-Q-Tel#Investments>

What influence does In-Q-Tel have on Google and its products? In 2004, Google acquired Keyhole Inc., the company that developed the technology behind Google Earth. This startup was funded by In-Q-Tel.

The technology is currently employed by US military and intelligence systems in their quest, in their own words, for the "full spectrum dominance" of the planet.

In addition, Google's link with the CIA and its private arm (IQT) extends to the sharing of at least one member of Google's staff. In 2004, In-Q-Tel's technology evaluation director, Rob Painter, moved from his former job directly to the CIA's service, to become "Senior Federal Manager" at Google.

As Robert Steele, a former CIA official, said: Google is "in bed" with the CIA.



But, it gets worse ...

Given Google's alleged concern about "human rights" and user privacy, it's worth noting that Wired magazine reported for some time that Google's friends at In-Q-Tel have invested in Visible Technologies, a software company specializing in monitoring social media.

Visible Technologies can automatically examine over one million discussion forums and posts on blogs, Flickr, YouTube, Twitter, Amazon, and so on, every day.

Visible Technology also "scores" each item online, assigning it a positive, negative, mixed or neutral state, based on parameters and terms defined by its technology operators. Information, thus summarized, can then be more effectively digitized and read by human operators.

There is also the Prism project, where the NSA collects internet communications from various American Internet companies, which was exposed by Edward Snowden in 2013.

So, we have finished the first part of this series of articles, addressing the origins of Google and their partners, both visible and hidden.

Stay with us, next month we'll address how Google captures your data, monitors your internet presence, and what to do to mitigate this intrusion.



PCLinuxOS Users Don't

Text
Phone
Web Surf
Facebook
Tweet
Instagram
Video
Take Pictures
Email
Chat

While Driving.

*Put Down Your
Phone & Arrive Alive.*

Disclaimer

1. All the contents of The PCLinuxOS Magazine are only for general information and/or use. Such contents do not constitute advice and should not be relied upon in making (or refraining from making) any decision. Any specific advice or replies to queries in any part of the magazine is/are the person opinion of such experts/consultants/persons and are not subscribed to by The PCLinuxOS Magazine.
2. The information in The PCLinuxOS Magazine is provided on an "AS IS" basis, and all warranties, expressed or implied of any kind, regarding any matter pertaining to any information, advice or replies are disclaimed and excluded.
3. The PCLinuxOS Magazine and its associates shall not be liable, at any time, for damages (including, but not limited to, without limitation, damages of any kind) arising in contract, tort or otherwise, from the use of or inability to use the magazine, or any of its contents, or from any action taken (or refrained from being taken) as a result of using the magazine or any such contents or for any failure of performance, error, omission, interruption, deletion, defect, delay in operation or transmission, computer virus, communications line failure, theft or destruction or unauthorized access to, alteration of, or use of information contained on the magazine.
4. No representations, warranties or guarantees whatsoever are made as to the accuracy, adequacy, reliability, completeness, suitability, or applicability of the information to a particular situation. All trademarks are the property of their respective owners.
5. Certain links on the magazine lead to resources located on servers maintained by third parties over whom The PCLinuxOS Magazine has no control or connection, business or otherwise. These sites are external to The PCLinuxOS Magazine and by visiting these, you are doing so of your own accord and assume all responsibility and liability for such action.

Material Submitted by Users

A majority of sections in the magazine contain materials submitted by users. The PCLinuxOS Magazine accepts no responsibility for the content, accuracy, conformity to applicable laws of such material.

Entire Agreement

These terms constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes and replaces all prior or contemporaneous understandings or agreements, written or oral, regarding such subject matter.



Like Us On Facebook!
The PCLinuxOS Magazine
PCLinuxOS Fan Club

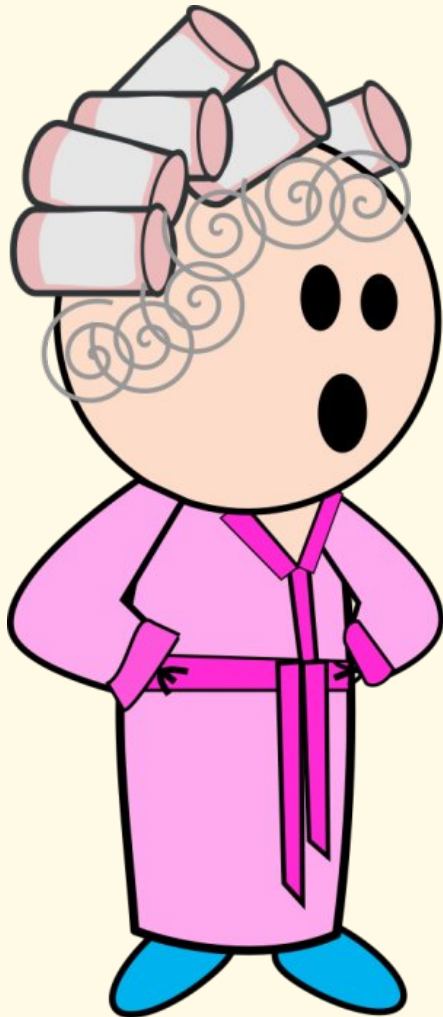


Screenshot Showcase



Posted by Aragorn, on April 6, 2019, running KDE.

ms_meme's Nook: Boot Up PCLOS



The moment I wake up before coffee I pour a cup
I boot up PCLOS
While getting my teeth in and creaming my wrinkled skin
I boot up PCLOS

Forever and ever I'll always be true
Oh how I love it
Forever and ever I'll never be blue
Oh how I love it
Forever and ever I'll boot PCLOS
To live without it heartbreak for me

I run from my coffee break and now feel wide awake
To boot up PCLOS
And when it is nighttime I dream up a new rhyme
About PCLOS

Forever and ever I will sing
Oh how you'll love it
Forever and ever some fun I'll bring
Oh how you'll love it
Forever and ever I'll boot PCLOS
To live without it heartbreak for me

MP3

OGG

Short Topix: Julian Assange Taken Into Custody

by Paul Arnote (parnote)

Don't Plug In A "Strange" USB Drive!



A friend “loans” you a USB drive. You find a USB drive, either left plugged in to a public computer, or one that has fallen from someone’s pocket. Now, we’ve all heard it before. NEVER plug a strange USB drive into your computer. You really don’t know what lurks there. But, being forever curious, that curiosity gets the best of you. Or, in hopes of finding its rightful owner, you insert it into your computer to find information that will allow you to return it to its rightful owner. Or, maybe you just want to snoop around on it and see just what’s there.

Now, Linux users are probably quite a bit less susceptible from malware and other malicious software than are Windows users, where programs and software can be easily installed and run without any user interaction at all, and often without the user even knowing. Conceivably, you could be putting all

of your data at risk, on any platform. But, do you really want to take that chance?

Apparently, the U.S. Secret Service is not immune from this, either. According to a TechCrunch [article](#), “a Chinese national, Yujing Zhang, was caught trying to sneak into President Trump’s private Florida resort Mar-a-Lago in March. The Secret Service caught her with four cell phones, a laptop, an external hard drive, a signal detector to detect hidden cameras, and a thumb drive.”

From the [Miami Herald](#) newspaper:

Secret Service agent Samuel Ivanovich, who interviewed Zhang on the day of her arrest, testified at the hearing. He stated that when another agent put Zhang’s thumb-drive into his computer, it immediately began to install files, a “very out-of-the-ordinary” event that he had never seen happen before during this kind of analysis. The agent had to immediately stop the analysis to halt any further corruption of his computer, Ivanovich said. The analysis is ongoing but still inconclusive, he testified.

Reading the Miami Herald article will just boggle your mind. According to that article, a search of the suspect’s hotel room discovered that she had **nine** USB drives, **five** SIM cards and other electronics, as well as over \$8,000 in cash.

While your initial reaction might be “what the ...,” when you stop and think about it, nothing other than the signal detector screams espionage – and even that can be explained away. When I travel, I typically have multiple USB drives, a combination of thumb drives and external hard drives. You just never know when you might need extra storage, after all. I like to backup my vacation photos so all is not lost should I have a device failure, or if my computer is stolen. I typically travel with an abundance of cash (usually

traveler’s checks, anyway, to protect against theft). And let’s face it: the U.S. **is** (or has become) a surveillance state. Heck, they will surveil you just to determine whether you have anything worth surveilling or not, and then continue to surveil you anyways, just to be sure. So it’s not too far-fetched to imagine that someone traveling to the U.S. from abroad might want to have a signal detector to help them avoid the unwanted and unwarranted intrusion into their personal affairs.

At any rate, take a lesson from Secret Service agent Samuel Ivanovich. DON’T plug “strange” USB thumb drives into your computer. As a Linux user, you’re quite well protected from most of the malware out there. But, some USB thumb drives are capable of frying or [nuking](#) a computer. And who, after all, wants something like that to spoil their vacation, or even their ordinary day? I just hope that someday, when traveling, “security” people don’t want to go through my computer gear and/or hotel room. I’m likely to be in a similar position as Yujing Zhang.

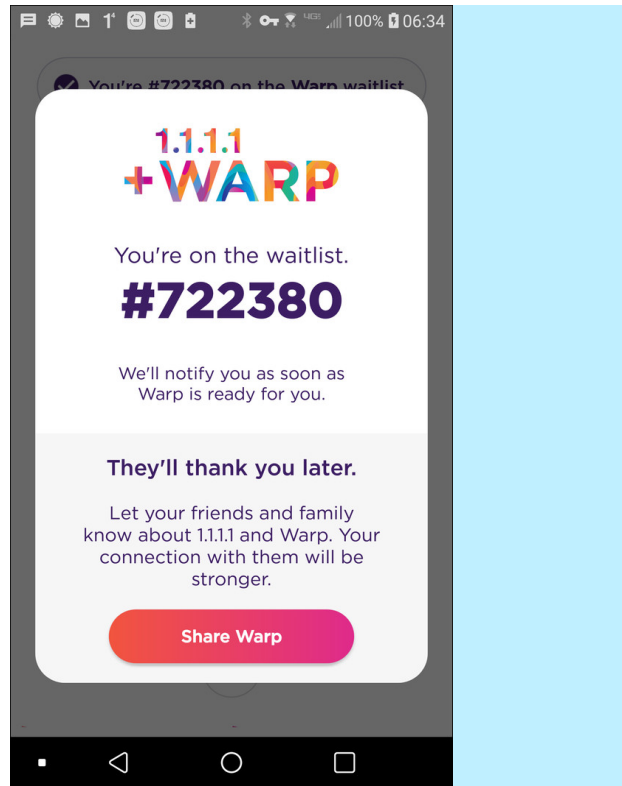
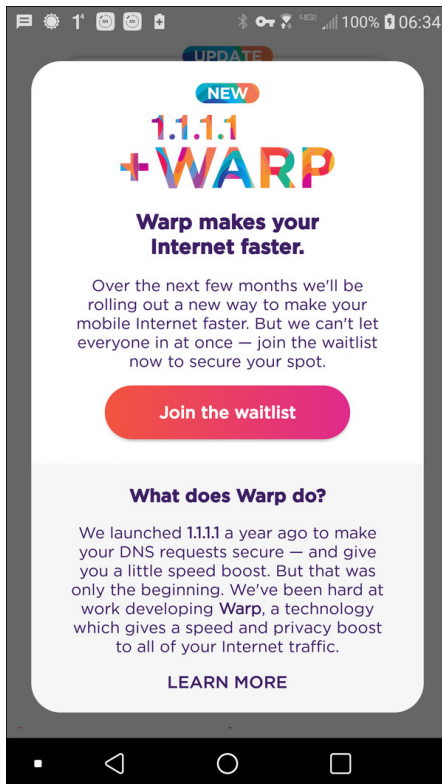
Reliable, Trustworthy, Free VPN For Your Wireless Devices



In the May 2018 Short Topix [column](#), we told you about Cloudflare's new privacy-enhanced DNS. Announced on April 1, a lot of people (including myself) thought it was a cruel April Fool's joke. But no, it was real, and I've been using it ever since.

Then, on November 11, 2018 (11/11), Cloudflare released their first ever mobile app to bring the enhanced speed and security of the 1.1.1.1 DNS resolver to mobile devices. It is available through both the [Apple](#) App Store and the [Android](#) Google Play Store.

This past April 1, Cloudflare announced the 1.1.1.1 + Warp program, which brings a bona fide, quality VPN to mobile devices. Warp will be free to use, but users wanting a bit more advanced version will be able to "upgrade" to Warp+, which will be available for a modest monthly fee.



As you might imagine, there are a lot of people clamoring for exactly this kind of app, specifically designed to work with mobile devices as they transition from cellular to wifi connections and back, and which works without placing a heavy load on the mobile device's battery. Through the 1.1.1.1 app, you can join the Warp waitlist. Just click on the "Join the waitlist" button in the app. Gauging by my position on the waitlist, there are 722,379 other people who will gain access to Warp before I get the opportunity.

Warp is designed to be easy to use. It is designed for people who don't even know what a VPN is, much less how to go about setting one up. Desktop users shouldn't be afraid, because Cloudflare is also working on Warp for desktop computers. You can read more about all of it on the Cloudflare [blog](#).

Julian Assange Arrested, Taken Into Custody



Source: SkyNews

[Julian Assange](#), an Australian journalist and the founder of WikiLeaks, was arrested and taken into custody April 11, 2019, after Ecuador withdrew its grant of diplomatic asylum to him. Assange had been staying in the Ecuador embassy in London since 2012, after Ecuador granted him diplomatic asylum to avoid being arrested under a warrant for jumping bail while trying to fight extradition to the U.S.

WikiLeaks was founded by Assange and others in 2006, to bring transparency to secretive and covert activities by politicians, governments, and government agencies around the world by publishing documents related to those secretive and covert activities. Assange and WikiLeaks achieved a high profile status after publishing secret documents provided by former U.S. Army soldier Chelsea Manning (formerly Bradley Manning). These documents detailed U.S. activities in the Afghanistan and Iraqi wars, along with U.S. diplomatic cables. Later, WikiLeaks published a large number of emails from former U.S. Secretary of State and twice failed presidential candidate Hillary Clinton, as well as damaging internal communications from the Democratic National Committee, all during the 2016 U.S. presidential race.

To many, Assange is a hero. To others, he is a criminal for publishing top secret or confidential

documents. But, is he any different than Daniel Ellsberg, who exposed atrocities of the Vietnam War, published in the New York Times? Is he any different than Bob Woodward and Carl Bernstein, who reported on and sparked the official Watergate investigation, as reported in the Washington Post? There are many other parallels to other whistleblowers that could be made. There are many who argue that governments, to effectively represent their people and to combat corruption, must be as transparent as possible, and secretive behaviors of those governments and its officials should be minimized.

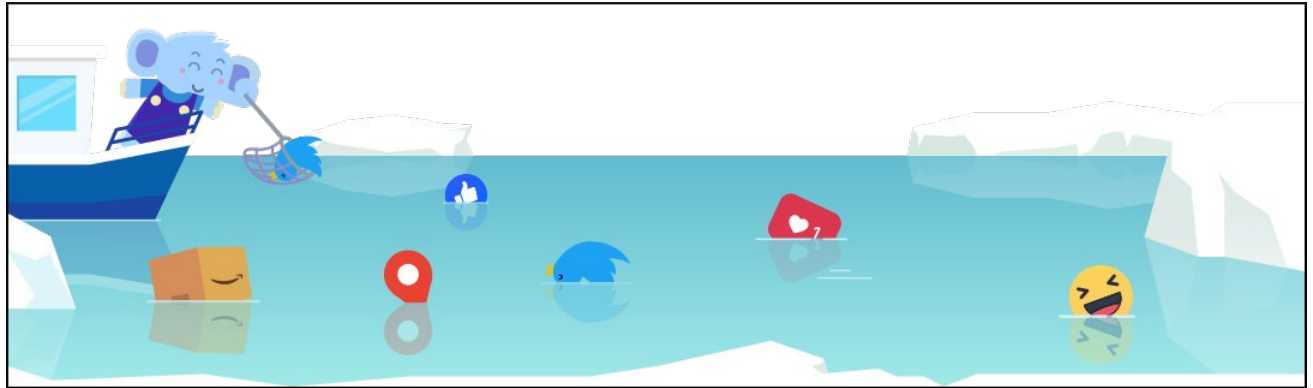
Assange's fate is currently in the hands of a UK magistrate, who will determine if Assange's extradition to the U.S. will be allowed. Concerns have been voiced about Assange's continued well-being, and human rights concerns should he be extradited to the U.S. Assange has already been indicted by a U.S. federal grand jury secretly for his supposed role in the release of the documents provided by Manning. [Here](#) is a summary (PDF) of the charges that Assange has been indicted for, filed in a U.S. District Court in Alexandria, VA.

Undoubtedly, there remains a lot of this story to still be written. I suspect we'll find this playing out over the next several years.

Jumbo: Privacy Assistant For Facebook, Twitter, & More

[Jumbo](#) is a new privacy assistant that will help maintain your privacy on Facebook, Twitter, Google, and Amazon Alexa. Soon, it will be adding Instagram and Tinder to the list of apps that it helps maintain your privacy.

Any Facebook user knows that controlling your privacy settings is a daunting experience (and just another reason among many why I refuse to have a Facebook account). From what I'm told, there isn't just one area to check to make your privacy settings.



Instead, they are spread out in several different areas. Jumbo acts as your agent and goes in and makes the changes in a wholesale manner. You simply tell Jumbo if you want weak, medium or strong privacy protections. It then does the rest.

For Twitter users, Jumbo lets you set how long before your tweets are deleted. You determine if Jumbo deletes your tweets that are older than a day, week, month (recommended), or three months. All deleted tweets are currently saved to Jumbo's "Memories" tab. The plans are to be able to download all of your "Memories" to either Dropbox or iCloud. At no time does Jumbo "see" your data, as all of your information is stored to your mobile device, locally. There is no server that it is sent to or synced with. Until the Dropbox and iCloud features come about, there currently is no other way to export your "Memories" from the app. If you're a prolific Twitter user, it may take a few rounds with Jumbo to delete all of your old tweets, since Twitter's API limits the number of old tweets you can delete to 3,200 every few days.

On Google, Jumbo will delete your search history. For Alexa, it deletes voice recordings stored by Amazon. The ability to delete your old Instagram photos and videos, along with your old Tinder matches and chats are reported to be coming next.

You can learn more about Jumbo by following the first link (which leads you to the apps home page),

or by reading the TechCrunch article, [here](#). Currently, the app is only available via the Apple App Store, but an Android version is reported to be coming soon.

Brexit Equals Uncertainty For UK Tech Firms



Back in 2016, by narrow margins, British voters chose to exit the EU. The referendum became known as "Brexit." It wasn't a straight-up vote across the UK, either. Northern Ireland and Scotland voted to remain with the EU, while England and Wales voted in favor of leaving the EU. Prime Minister Theresa May delivered the order for the exit from the EU on March 29, 2017. The referendum allowed for a two year period for Britain to leave the EU, and the whole thing was to be finalized on March 29, 2019. Without getting into the why's and wherefore's, that

date is now set for the end of October 2019 (as best I can tell).

In the aftermath, tech firms in the UK are having difficulty attracting talent to fill the available tech jobs, thanks to the uncertainty surrounding Brexit. According to a TechRepublic [article](#), the uncertainty stems from not knowing what form Brexit will take. More uncertainty stems from not knowing whether or not there will be a 21 month transition period after Brexit, which represents a period of time when the UK's relationship with the EU will stay largely the same as it is. That transition period is set forth in the Withdrawal Agreement negotiated by Theresa May's government and the EU.

Parliament, however, has overwhelmingly rejected the Withdrawal Agreement three times. This has caused Theresa May to open negotiations with the Labour Party, who is largely in opposition to Brexit, to come up with a Brexit strategy that is agreeable to both the Labour Party and the EU.

Another uncertainty for the UK tech industry is how EU data will be handled by the UK post-Brexit, and the ability to freely exchange data between the EU and UK post-Brexit. One thing that is striking fear into the UK tech industry is a no-deal Brexit, where the UK leaves the EU at the end of October without an approved formal Withdrawal Agreement. Nearly 70% of tech companies claim that a no-deal Brexit would have negative ramifications for their businesses, should it happen.

To say that the entire Brexit proceedings, as well as trying to imagine a post-Brexit world, are a tangled web full of about faces and dead ends would be a gross understatement. There is a lot to ponder and consider, and having a Withdrawal Agreement in place to ease the transition certainly seems to make the most sense.



Roundup At The PCLinuxOS Corral



OVER AT [SLASH GEAR](#), they definitely make a strong case for abandoning Windows for Linux. Given Windows 10's lengthy problems with updates, Windows users could start to view the transition to Linux with greater interest. So what about Linux did they use to make a strong case for making the switch to Linux? Well, having a curated software repository, a smoother update process for programs and the OS, a choice of highly customizable desktops, being WAY more lightweight than Windows, and less data extraction. They also point out that Linux is more secure, and is nearly always FREE. If you're already a Linux user, then these are things you most likely already know. But, it will make good ammunition for getting your reticent Windows friends to finally give Linux a try.

Linux Journal's very first issue featured an interview between LJ's first Publisher, Robert Young (who went on to co-found Red Hat among other things), and Linus Torvalds (author of the Linux kernel). After 25 years, they thought it'd be interesting to get the two of them together again.

You can read that first interview from 1994 [here](#). So, what's the secret to Linux's long term success? According to Torvalds, it's the open source development model. You can read the newest interview [here](#), along with all of the clarity that 25 years of experience provides on a wide range of topics.

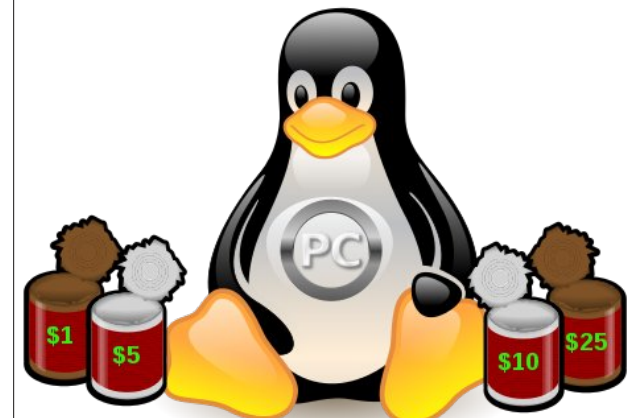


Donate To PCLinuxOS

*Community Supported.
No Billionaires/Millionaires.
No Corporate Backing Or Funding.*

*Click [here](#) to make a one-time donation
through Google Checkout.*

*Or, click one of the amounts down below
to make a monthly, recurring donation.*





Visit Us On IRC

- Launch your favorite IRC Chat Client software (xchat, pidgin, kopete, etc.)
- Go to freenode.net
- Type "/join #pclosmag" (without the quotes)

Does your computer run slow?

Are you tired of all the "Blue Screens of Death" computer crashes?



Are viruses, adware, malware & spyware slowing you down?

Get your PC back to good health TODAY!

Get



Download your copy today! FREE!



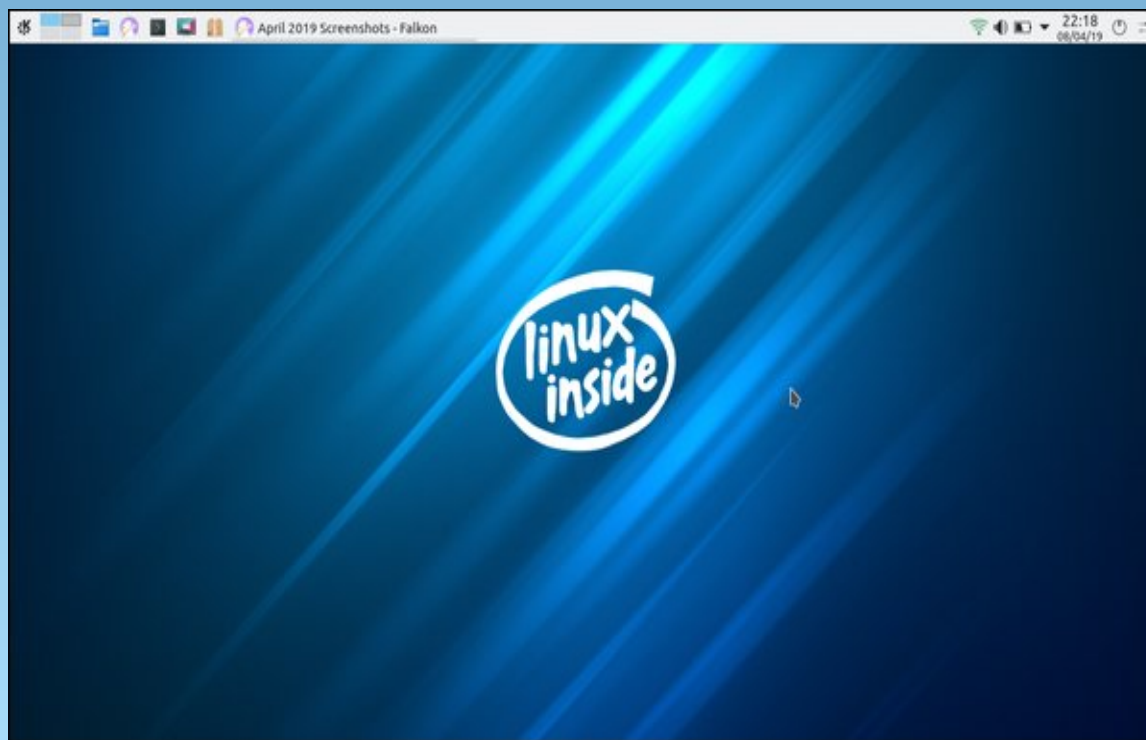
**Linux Training
Courses & Classes**

BeginLinux.com

CHIMPBOX

Big Things, Little Packages. Measuring only 7.5 X 8.5 X 2 inches
<http://chimpbox.us>

Screenshot Showcase



Posted by hurricane, on April 8, 2019, running KDE.

GIMP Tutorial: More About Masks

by Meemaw

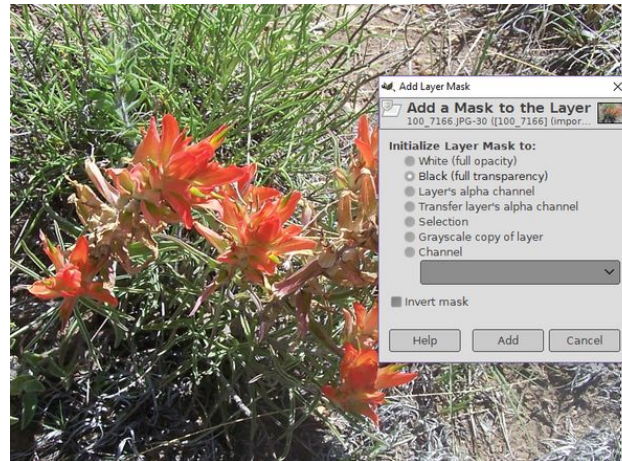
I was talking to some of the guys in PCLOS-Talk the other day, and they were talking about GIMP, and how they were wanting to understand masks. They wanted to learn more about how they worked, and told me that one of my future tutorials needed to cover that.

In my research, I discovered that I had already done a tutorial about masks in the [August, 2013](#) issue, but that's been a while, so I will revisit the subject here.

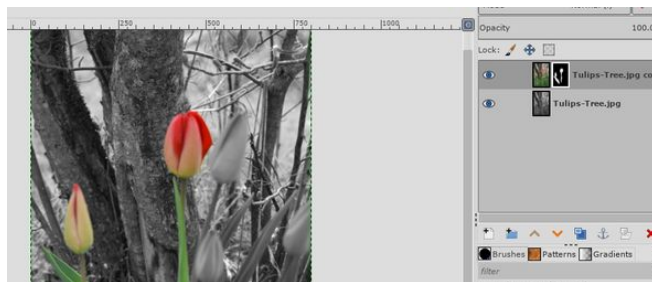
As I said in the first article, a mask is a layer you can add to your project to affect certain parts of it. However, instead of being another layer, the mask is attached to a layer and affects the pixels on that layer. It's used to block or show the pixels on the layer it is part of, so your project is changed in some way.

According to the online book, [Grokking the GIMP](#), *layer masks are special layers that are only 8 bits deep and that represent the alpha channel of an image layer*. Now I'm not quite sure I understand it when stated that way, but it is a special layer that can be manipulated to change the image.

To add a layer mask, open the image you want to use. In the layers dialog, right click on the layer where you want the mask. A menu will open, and click on **Apply Layer Mask**. You'll get the window shown at center top:



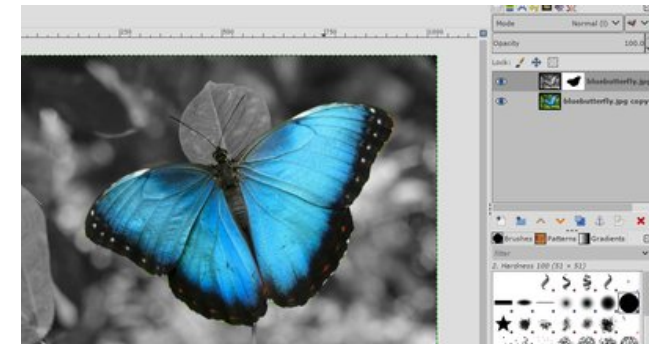
the same layer as the mask. In one of the projects, I changed the mask itself to a gradient (grey to black, so the top left of the photo was partially colored and the bottom right was black & white like the desaturated copy below), then painted the flower with white to highlight it. Here is another example of the first kind, using the tulips in my yard.



In each instance I used a black layer mask and painted it with white, essentially covering up the layer then exposing selected areas of it.

A white mask does the opposite: it makes the layer visible. Painting it with black hides that part of the layer, revealing the layer below. On the following

photo, I put the desaturated layer on top and used a white mask, painting it with black to expose the color photo below.



The dramatic effect above can be done with either a black or white layer mask. You just have to place the desaturated layer in a different spot.

I can also use this tool to quickly erase the background from a photo that has just the picture I want to use elsewhere. We went to an auto auction a few years ago (just to look at the vehicles) and I snapped this picture:





If I want to use only this truck in another project, I will want to get rid of the background. Open it in GIMP, and apply a white layer mask to this layer. Notice in the image below that I have also selected it with the Lasso tool. Go to **Select > Invert** to change the selection to the background. The white blob in the top left corner is what happens when I start painting with the eraser, but if I use the paintbrush with black, the background disappears (top right corner). When it is exported, only the truck shows. An extra added attraction is that if your hand slips and you run over the truck, it doesn't get erased, because it isn't in the selection.



You could also use another tip: if your object can be outlined relatively easily with one of the selection tools, you can then bucket fill your selection with black instead of painting it in with the brush. I use the Lasso or Intelligent Scissors most, but you can also use the Paths tool. It is much faster and more accurate than painting! I used that on the butterfly photo, but the truck and the flower photos would be a bit more difficult (because there are more detailed edges). As you can see above, I didn't outline the truck very well. If I was going to use that in another project I would have to be much more careful. However, when I get the truck outlined correctly, I can invert the selection, then bucket fill with black. If you press Delete, the background will disappear, but replaced with a white background (I started with a jpg file, which doesn't use transparency). Then you could export it as a png.

When the online book was written there were only 3 choices in the Layer Mask window. Now there are 7, and I'm still learning. The book says that the layer mask will only work if the photo has an alpha channel, and there is a choice to add one if needed. We'll revisit this again, but I need to learn more first.

Support PCLinuxOS! Get Your Official

PCLinuxOS
Merchandise Today!

PCLinuxOS



Tip Top Tips: Linking Your Orage Calendar To Google Calendar

Editor's Note: *Tip Top Tips* is a semi-monthly column in *The PCLinuxOS Magazine*. Periodically, we will feature – and possibly even expand upon – one tip from the PCLinuxOS forum. The magazine will not accept independent tip submissions specifically intended for inclusion in the *Tip Top Tips* column. Rather, if you have a tip, share it in the PCLinuxOS forum's "Tips & Tricks" section. Your tip just may be selected for publication in *The PCLinuxOS Magazine*.

This month's **tip** comes from **davecs**. Instead of the Tips & Tricks section of the forum, this tip appeared in the Xfce section of the PCLinuxOS Forum. Orage is the Xfce calendar program.

I posted about this a few years back, but that solution doesn't work any more. I've done some research, and found another way to do this. It seems a little complicated, but just follow everything as I've set out, and you should be OK!

BEGIN>

In order to sync Orage to your Google Calendar, you need add a program called "**python3-pip**" from Synaptic. This enables you to install some useful python-based programs direct from the internet.

Then you'll need to open a terminal as root, and install two further programs using pip. All you have to do is type in the following two lines:

```
pip3 install vdirsyncer
pip3 install requests_oauthlib
```

Then you need to log in to the Google API dashboard:

<https://console.cloud.google.com/apis/dashboard>

Once there click on "+ **ENABLE APIS AND SERVICES**" near the top of the screen.



This takes you to a search screen, and in the search box, enter "**caldav**". You will get a box to click on, "**CalDAV API**" which you should select. You will need to **Create Credentials**. In return, you will get two long IDs, **client_id** and **client_secret**.

Open your File Manager, and Show Hidden Files, click into **.config** and create a folder within that called **vdirsyncer** — and inside that create a text file called **config**.

You should create a folder under **~/local/share/orage/** which is your gmail address (shown as **youraddress@gmail.com** in the file that follows).

config should look like this. You will use your own **gmail address**, **client_id** and **client_secret**, but everything else should be typed exactly as I have:

```
[general]
status_path = "~/vdirsyncer/status/"
```

```
[pair mygoogle_calendar]
a = "mygoogle_calendar_local"
b = "mygoogle_calendar_remote"
collections = ["from a", "from b"]
```

```
[storage mygoogle_calendar_local]
```

```
type = "singlefile"
path = "~/local/share/orage/youraddress@gmail.com/%s.ics"
```

```
[storage mygoogle_calendar_remote]
type = "google_calendar"
token_file = "~/vdirsyncer/google_token"
client_id =
"1235467890abcdefghijklmnopqrstuvwxyz.apps.googleusercontent.com"
client_secret = "1234567890abcdefghijklmnopqrstuvwxyz"
```

Now, make a folder in your home folder (not under **.config**) called **./vdirsyncer**, inside that, create another folder called **"status"**, and inside that, another folder called **"calendar"**.

Tip Top Tips: Linking Your Orage Calendar To Google Calendar

That's the preparation done. Now for the moment of truth. You need to open a terminal, and run the following command as user:

vdirsyncer discover

A browser window should pop up to complete the authorisation, and that will create an access token, another long string of characters, which you need to copy to your clipboard. Meanwhile, the terminal goes into a "waiting" state. If you click on the terminal and hit a key, you will be prompted for the access token, which you should paste in. Keep an eye on the text that outputs to the terminal, in order to know that everything has gone ok.

Once this is done, you need to make the first synchronisation. You do this by typing the next line into the terminal:

vdirsyncer sync

When the sync finishes, you should have one or more files in the folder `"/home/your-user-name/.local/share/orage/youraddress@gmail.com/"` which you can now link to orage.

To do this, you have to click on your panel orage time and date to bring up the calendar. You link the calendars by clicking on **File>Exchange Data**, and the tab called **Foreign Files**. Look for them in the the folder I just mentioned.

There may be a few with very long random names, which you will link as **READ ONLY**, except one that is almost human-readable, shorter and similar to your email address, who you should link as **READ WRITE** (but I am ready to be corrected because I'm not sure if that's right. It seems to work). At the moment I am getting down-syncing from Google OK but not up-syncing to Google, so if anyone can correct this for me? Still that's better than nothing, provided you use your browser or phone to add new entries to your calendar.

Setting up a cron job seems fairly easy. You have to open a terminal as user, and type in the command **crontab -e**. An editor called **vi** will run. If you're not familiar with **vi**, in order to start entering text, you type **"i"**.

Type in a line like:

```
26 * * * * /usr/local/bin/vdirsynchroner sync
```

followed by **ENTER**.

To save it, type **ESC** (which takes you back to command mode), followed by **:wq** which writes your file then quits. The line entered above will cause the job to run

on the 26th minute of every hour. If you have more than one scheduled task running it's best that they don't all start at the same time, but you can change 26 if you like.

If you want to get more control over timing and frequency, I suggest you read up some help on **crontab**.



PCLinuxOS *Users Don't*

Text
Phone
Web Surf
Facebook
Tweet
Instagram
Video
Take Pictures
Email
Chat

While Driving.

***Put Down Your
Phone & Arrive Alive.***



PCLOS-Talk
Instant Messaging Server

Sign up **TODAY!** <http://pclostalk.pclosusers.com>



Screenshot Showcase



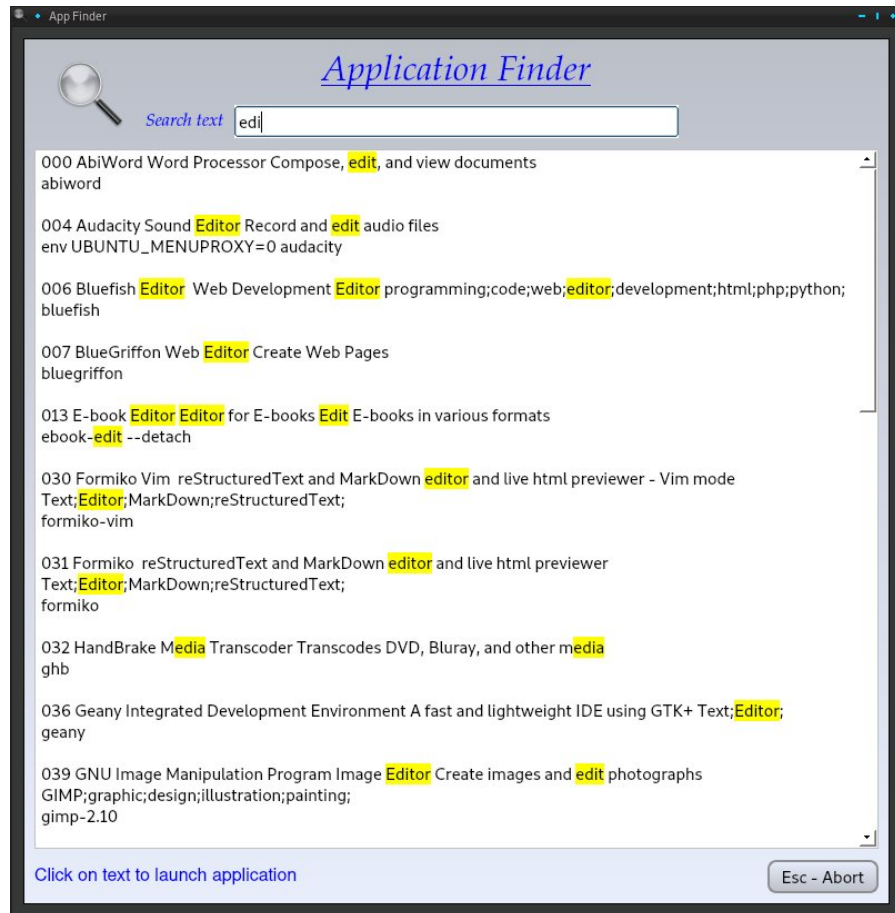
Posted by mutse, on April 24, 2019, running Mate.

Casual Python, Part 4

by critter

Appsearch

This application is more complex than the previous examples. It will take more time to explain, and even more time to understand, as there a lot of new ideas coming all at once. When you can understand (most of) this example, you will have a good understanding of basic python. Some of the code in here is admittedly not basic stuff, so if it is a bit too much to 'get your head around,' just accept that it works for now. Eventually, you will grasp it.



I have a lot of applications installed on my system. Some I don't use very often, but still find useful. When I need one of these rarely used applications, I often cannot remember the name and sometimes, when searching through the menu, the name doesn't readily reflect the purpose or function of the application. The next utility that I am going to build goes some way to solving this problem by providing real-time searching on any word and single click launching of the selected application. It will be based on the template application and looks like the snapshot above.

Suppose that I need to use a HTML editor. I know that I have at least one installed, but cannot remember the name.

As I begin to type 'editor' in the box, the search is refined, at first showing every possibility that contains the letter 'e', then 'ed'. By the time that I have typed in 'edi,' I have found two possibilities – Bluefish and Blue Griffon (scrolling down reveals no more than the two shown in the screenshot). Clicking anywhere in the respective text closes the appsearch utility and launches the chosen HTML editor. This is much quicker than searching the menu.

How it works

Most Linux desktops adhere to a standard defined by freedesktop.org. When an application is installed, a file with the name of the application and the extension '.desktop' is placed in the /usr/share/applications directory. This file describes the applications name and features, usually in many languages, and consists of several sections so the entire file can be quite long. For the purposes of this application, I am going to restrict its' use to the default English language and to five sections:

- | | |
|---------------|---|
| • Name | the application's name. |
| • Genericname | this might be something like editor |
| • Comment | some description of the application |
| • Keywords | words that might be used to search for the application |
| • Exec | what you would type at the command line to launch the application |

For the bluegriffon.desktop file this reduces to:

```
Name=BlueGriffon
GenericName=Web Editor
Comment=Create Web Pages
Exec=bluegriffon
```

There is no 'Keywords' entry in this file. This is the information that was displayed by the search in the snapshot.

The application will read each of the '.desktop' files in /usrshare/applications and create a list of this information for each of the files. It will then search for whatever is currently in the text search box in these lists and display matches, highlighting the matched part. The highlighting is helpful as in the example above 'edi' is found in Media but we are looking for editors, so this line can be ignored. The number on the left is added by the application, and is an index into the generated lists. Clicking on the text for bluegriffon sends the index 007 to the application, and the Exec field of the list tells how to launch the selected application.

Creating the new application

Create a new directory called qt5_appsearch and copy the following files from the qt5_template directory into this new directory renaming them as follows:

```
qt5_template.py      =>    qt5_appsearch.py
template.ui          =>    appsearch.ui
update_res.sh        =>    this stays the same
```

Also place a suitable icon in there.

Edit update_res.sh and change both occurrences of template to appsearch.

Next, we have to re-work the user interface file so open designer. Select open, and navigate to our new directory. Open qt5_appsearch.ui.

You will now see the template user interface. Change the form to look like the example above, using the same methods as before. You may need to right click on the background and select layout - break layout. Make sure that your QTextEdit widget is a QTextEdit, not a QPlainTextEdit (this is necessary to show the highlighting). The only widgets that we shall be interacting with directly are the line edit, textedit and button, so make sure that their object names are exactly:

lineEdit, **textEdit** and **btn_Quit**

as this is how the code recognizes them. When you are happy with the

appearance, check it out with a control-r preview, and then save it. Run update_res.sh to generate the python code for the user interface.

Designing the code

I have already said that the application depends on data found in freedesktop.org standard files, and as I am using the bluegriffon editor as an example. Here is the bluegriffon.desktop file in full with the lines of interest highlighted:

```
[Desktop Entry]
Name=BlueGriffon
GenericName=Web Editor
GenericName[hu]=Webszerkesztő
X-GNOME-FullName=BlueGriffon Web Editor
X-GNOME-FullName[hu]=BlueGriffon webszerkesztő
Comment=Create Web Pages
Comment[es]=Crea páginas web
Comment[hu]=Weboldalak készítése
Comment[it]=Creare pagine Web
Categories=Development;WebDevelopment;
Exec=bluegriffon
Icon=bluegriffon
Terminal=false
MimeType=text/html;application/xhtml+xml;
Type=Application
```

To use the top down design method it is necessary to determine the 'top' which here is to launch an application.

Launch the application

Before we can do that we have to choose the application.

Choose an application Launch the application

To be able to make our choice we need to display a list of applications and their descriptions.

Display a list of applications and their descriptions Choose an application Launch the application

In order that we may generate a list of suitable applications we have to get some requirements from the user.

get search criteria from the user

Display a list of applications and their descriptions
 Choose an application
 Launch the application

The very first thing we need is a list of all available applications and their descriptions.

Create a database of available applications and their descriptions
 Interrogate the user for search criteria
 Display a list of applications and their descriptions
 Choose an application
 Launch the application

That then is our design.

The structure of the code will be:

some imports

a class definition which contains 8 method definitions

```
__init__
buildDb
addCount
showResults
findText
MousePressed
KeyPressEvent
exitApplication
```

finally our standard code that starts; if `__name__ == '__main__':`

The imports:

```
import sys
import os
import glob
import subprocess
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
import appsearch_ui
```

The first item in the design is to create a database of application information for all available applications.

This can be broken down into steps as above but with more detail:

change to the directory where the .desktop files are stored

```
read in a list of applicable filenames
create an empty list to store the retrieved data
read in each file
    create a list of 5 empty strings to store the data
    check each line of the file for relevance
    if relevant store the content in the list
append the completed list to the data list
index the database
```

Each application may have up to five items of data: Name, GenericName, Comment, Keywords and Exec. We can easily store these in a python list of strings. However, not all applications have all sections completed (Bluegriffon has no Keywords section), so we should start with a list of five empty strings. In the final code, it looks like this, with comments.

```
def buildDb(self):
    ''' collect data from freedesktop.org standard entries
        located in /usr/share/applications
        store the data as a list of text lists'''

    os.chdir('/usr/share/applications')
    path = '*.desktop'
    files = sorted(glob.glob(path), key=str.upper)
    self.data = []
    for desktop_file in files:
        with open(desktop_file) as f:
            # supply an empty string for missing fields
            field_list = ['', '', '', '', '']
            for line in f:
                if line[0:5] == 'Name=':
                    appname = line.split('=')[1].strip().rstrip('\n')
                    field_list[0] = appname
                if line[0:12] == 'GenericName=':
                    apptype = line.split('=')[1].strip().rstrip('\n')
                    field_list[1] = apptype
                if line[0:8] == 'Comment=':
                    comment = line.split('=')[1].strip().rstrip('\n')
                    field_list[2] = comment
                if line[0:9] == 'Keywords=':
                    keywords = line.split('=')[1].strip().rstrip('\n')
                    field_list[3] = keywords
                if line[0:5] == 'Exec=':
                    execstring = line.split('=', 1)[1].rstrip('\n')
                    for c in '%f', '%F', '%u', '%U':
                        execstring = execstring.replace(c, '')
                    field_list[4] = execstring
            elf.data.append(field_list) # add the record
    self.addCount()
    return self.data
```

This follows the detailed breakdown quite closely. A list of files is created, which is then sorted, Uppercase first, and stored in a list. Another empty list named data is created, ready to store the final extracted information.

The for loop runs through each file, which is then opened for reading (the default), and a new list of five empty strings is created to store the information about each application. Another for loop then steps through each line of the current file, and uses string slicing to compare the start of the line to a required pattern. If found, the line is split at the '=' sign, the 'end of line character' (\n) removed, and the rest of the line stored in the first element of the field_list list.

If there is no match, then the line is compared to the next heading of interest. This is repeated for each of the five fields of interest. In the 'Exec' string, there are often some characters that would confuse our code, and these, if present, are replaced by blanks. The completed field_list is appended to the data list.

Finally, when all of the files have been processed and added to the data list of lists, another routine (method) is called to index the database. This indexing code has been separated into another method, as the buildDb routine was already quite complex and smaller 'chunks' of code are easier to manage and may also be re-used in other applications.

The addCount method which does the indexing is next, the comments explain it:

```
def addCount(self):
    ''' index the database of lists '''

    count = 0                # Intialize the count
    for app in self.data:    # loop through the lists
        number = str(count).zfill(3) # pad with zeros to length 3
        app.append(number)    # add the number to the
                             # applications data
        count += 1           # add 1 to the count each time
                             # through the loop
```

Each application gets a number. Bluegriffon's entry now looks like this:

```
['BlueGriffon', 'Web Editor', 'Create Web Pages', '', 'bluegriffon', '007']
```

Then we have the showResults method:

```
def showResults(self):
    ''' update the results pane '''

    self.textEdit.clear()
    for line in self.apptext:
        t = self.lineEdit.text()
```

```
if t in line[0].lower() \
    or t in line[1].lower() \
    or t in line[2].lower() \
    or t in line[3].lower():
    found = line[5] + ' ' + \
        line[0] + ' ' + \
        line[1] + ' ' + \
        line[2] + ' ' + \
        line[3] + ' ' + \
        '\n' + \
        line[4] + '\n'
    self.textEdit.append(found)
```

self.findText()

Remove any text in the textEdit widget.

For each line of data in the apptext database, we look for the typed in text, which we are calling 't' here. The if construct compares 't' to the lowercase version of each element of the current line. If a match is found, then we build a string called 'found', starting with the index number and adding each element of the matched line, separated by spaces, which we add to the textEdit. Finally, from here we call the findText method, which does the highlighting for us, and has been separated out into a new definition.

```
def findText(self):
    ''' find all matching text and highlight to the end
        of the first word of the matching text'''

    cursor = self.textEdit.textCursor()
    Format = QTextCharFormat()
    Format.setBackground(QBrush(QColor("yellow")))
    Format.setForeground(QBrush(QColor("black")))
    pattern = self.lineEdit.text()
    regex = QRegExp(pattern)
    pos = 0
    self.matches = []
    index = regex.indexIn(self.textEdit.toPlainText(), pos)
    while (index != -1):
        cursor.setPosition(index)
        self.matches.append(index)
        cursor.movePosition(QTextCursor.EndOfWord, 1)
        cursor.mergeCharFormat(Format)
        pos = index + regex.matchedLength()
        index = regex.indexIn(self.textEdit.toPlainText().lower(),
pos)

    self.find_index = 0
    cursor = self.textEdit.textCursor()
    self.lineEdit.setFocus()
    cursor.setPosition(self.matches[self.find_index])
```

```
self.textEdit.setTextCursor(cursor) # go to first find
```

This method is quite complex, so I am not going to explain it all. This is a method that highlights all text in a textEdit widget that matches text in a lineEdit widget. We can re-use this anytime we need to highlight matching text without understanding it.

The mousePressed method is very similar to the one we used in the docz application.

```
def MousePressed(self, Event):
    ''' mouse button handler '''

    textCursor = self.textEdit.cursorForPosition(Event.pos())
    textCursor.select(QTextCursor.BlockUnderCursor)
    self.textEdit.setTextCursor(textCursor)
    word = textCursor.selectedText()
    word = word[1:]
    command = (self.data[int(word[:3])][4])
    if ' ' in command:
        subprocess.Popen(command, shell=True)
    Else:
        subprocess.Popen(command)
self.exitApplication()
```

There is one difference. If the command includes a space, then the Popen method needs the shell=True option to interpret it. This is what the if/else construct is for.

The last two methods, keyPressEvent and exitApplication are the same as previous ones.

In the final part of the code we create an instance of the class

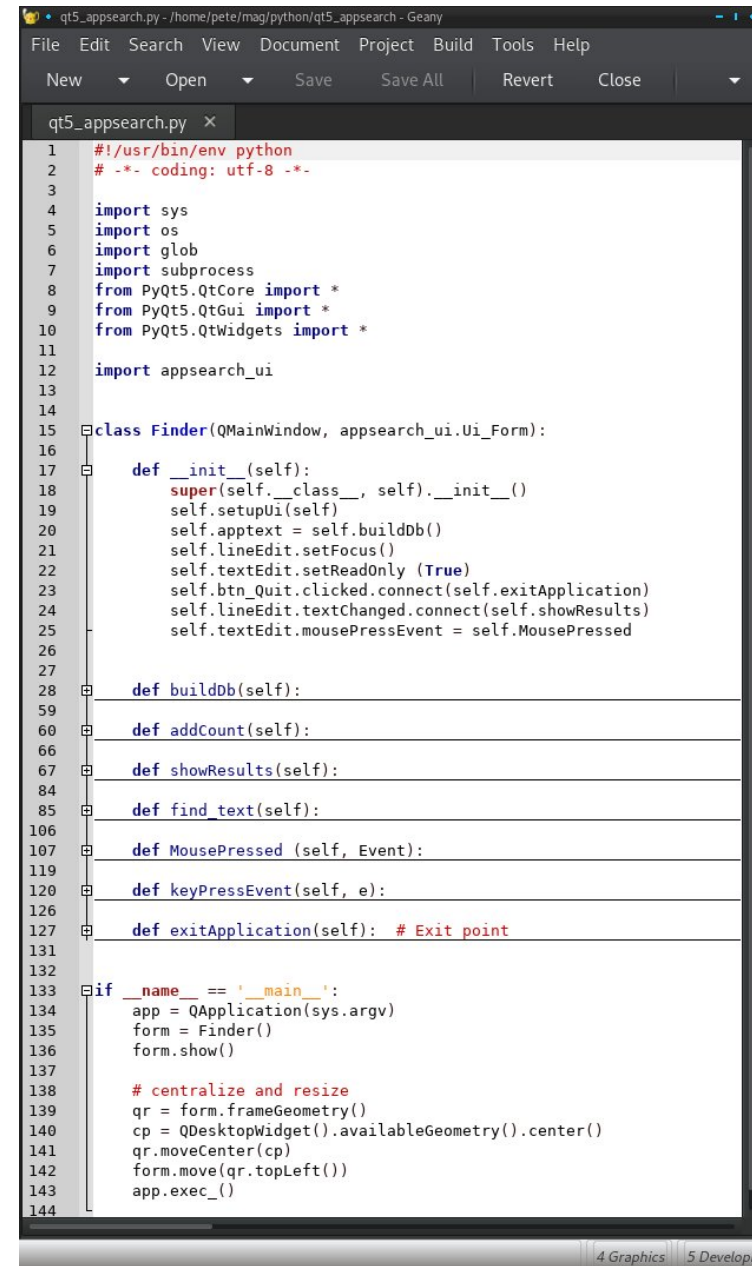
```
if __name__ == '__main__':
    app = QApplication(sys.argv)
    form = Finder()
    form.show()
```

And to center the form on screen we could add this:

```
qr = form.frameGeometry()
cp = QDesktopWidget().availableGeometry().center()
qr.moveCenter(cp)
form.move(qr.topLeft())
app.exec_()
```

And move `form.show()` to the line below it.

The final code in geany, with the code I have already shown in detail, collapsed looks like this:



```
qt5_appsearch.py x
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  import sys
5  import os
6  import glob
7  import subprocess
8  from PyQt5.QtCore import *
9  from PyQt5.QtGui import *
10 from PyQt5.QtWidgets import *
11
12 import appsearch_ui
13
14
15 class Finder(QMainWindow, appsearch_ui.Ui_Form):
16
17     def __init__(self):
18         super(self.__class__, self).__init__()
19         self.setupUi(self)
20         self.apptext = self.buildDb()
21         self.lineEdit.setFocus()
22         self.textEdit.setReadOnly(True)
23         self.btn_Quit.clicked.connect(self.exitApplication)
24         self.lineEdit.textChanged.connect(self.showResults)
25         self.textEdit.mousePressEvent = self.MousePressed
26
27
28     def buildDb(self):
29
30
31     def addCount(self):
32
33
34     def showResults(self):
35
36
37     def find_text(self):
38
39
40     def MousePressed (self, Event):
41
42
43     def keyPressEvent(self, e):
44
45
46     def exitApplication(self): # Exit point
47
48
49
50
51 if __name__ == '__main__':
52     app = QApplication(sys.argv)
53     form = Finder()
54     form.show()
55
56     # centralize and resize
57     qr = form.frameGeometry()
58     cp = QDesktopWidget().availableGeometry().center()
59     qr.moveCenter(cp)
60     form.move(qr.topLeft())
61     app.exec_()
```

Variables

As you know by now, variables are the names we use to reference our objects. So what is a valid variable name?

- A variable should start with an underscore or letter
- The remainder of the variable may consist of letters, numbers and underscores
- All uppercase is conventionally used for constants
- Avoid lowercase o (oh), l (ell) and uppercase I (eye). These can be confused with other characters

For other conventions see pep8.

The range function

Python provides a range function that produces a range of values.

`range(start, stop, step)`.

The start, stop and step arguments mean the same as for list indices and the stop value is **not** included. The function returns an iterable range object not a list of numbers.

```
For i in range(5):
    print(i)
0
1
2
3
4
For i in range(2, 15, 3):
    print(i)
2
5
8
11
14
```

Tuples

Lists are extremely useful, and will be found in lots of python code. Their main disadvantage is that they are mutable, and so can be changed from under you. When you need an immutable list, use a tuple. They are not quite as flexible as lists, but you can rely on their contents being as expected.

A tuple is an ordered, immutable sequence of comma separated items. Parentheses are often used surrounding the items to avoid ambiguity. A tuple with a single value must have a trailing comma: `single_tuple = ('a',)` Without the comma python will create an instance of the native data type: `str`, `int` etc. An empty pair of parentheses will give an empty tuple. All of these are valid tuples.

```
t1 = ('a', 'b', 7, 'c')
t2 = 'a', 'b', 7, 'c'
t3 = ('a',)
t4 = ()
```

Tuple indices and ranges are started in square brackets `[]` as in lists.

Slicing a tuple creates a new tuple, but tuples cannot be changed in any way.

Tuples are faster than lists.

An empty tuple is boolean `False`, and all other tuples are `True` regardless of value.

You can convert between tuples and lists using the `list()` and `tuple()` functions.

Multiple assignment can be achieved using tuples. This is known as unpacking.

```
In [1]: volumes = ('/dev/sda1', '/dev/sda2', '/dev/sdb1')
```

```
In [2]: root, home, data = volumes
```

```
In [3]: root
Out[3]: '/dev/sda1'
```

```
In [4]: home
Out[4]: '/dev/sda2'
```

```
In [5]: data
Out[5]: '/dev/sdb1'
```

Tuples may also be used to return multiple values from a function or method.

```
In [6]: def powers(x):
...:     square = x * x
...:     cube = x ** 3
...:     return square, cube
...:
```

```
In [7]: powers(4)
Out[7]: (16, 64)
```

Tuple concatenation and repetition are also supported as for lists. Refer to that section for a description.

The following list methods are **not** supported by tuples:

append, clear, copy, extend, insert, pop, remove, reverse & sort.

Dictionaries

A dictionary is an unordered collection of key-value pairs, often compared to associative arrays in other languages. Each key must be unique and have a value. A dictionary is a mapping, not a sequence. An empty dictionary is False, while all other dictionaries are True.

Note that as a dictionary is an unordered collection, the key order is not maintained so that numerical index as with lists is not possible. However, in a for loop for example, each key,value pair will be visited.

The keys in a dictionary must be unique and of an immutable type. The values may be of any type.

Unlike lists, which need to use the append method to grow in size, a dictionary may be freely added to and may be changed in-place. A dictionary may be created by literal assignment, dynamic assignment, keyword argument format or by assigning a list of key,value tuples.

```
a_dict = {1: 'a', 2: 'b', 3: 'c', 4: 'd'}
# literal assignment
print(a_dict[2]) # >>> 'b'
```

```
b_dict = {}
b_dict[10] = 'j'
b_dict[11] = 'k'
b_dict[12] = 'l'
b_dict[13] = 'm'
# Dynamic assignment
print(b_dict[12]) # >>> 'l'
```

```
c_dict = dict(t=20, u=21, v=22, w=23)
# keyword argument format
# keys must be strings.
# no space around '='
# quotes must not be used around keyword
# i.e. it can not be an expression
# but quotes must be used to access the value
```

```
print(c_dict['t']) # >>> 20
```

```
d_dict = dict([(1, 'uno'),
               (2, 'due'),
               (3, 'tre'),
               (4, 'quattro')])
# list of key,value tuples
```

```
print(d_dict[4]) # >>> quattro
```

Values may be referenced by key but not keys by value which return None.

Modifying a dictionary

A dictionary may be added to or modified in-place.

```
a_dict[5] = 'e' # adds key-value pair 5:'e'
```

```
print(a_dict) # >>> {
# 1: 'a',
# 2: 'b',
# 3: 'c',
# 4: 'd',
# 5: 'e'}
```

```
a_dict[5] = 'E' # changes the value of key 5
print(a_dict) # >>> {
# 1: 'a',
# 2: 'b',
# 3: 'c',
# 4: 'd',
# 5: 'E'}
```

Length of a dictionary

The following dictionary has a length of two. There are 2 keys, 'deutsch' and 'francais', and each key is mapped to a list.

```
e_dict = {
'deutsch': ['eins', 'zwei', 'drei', 'vier', 'fünf'],
'francais': ['un', 'deux', 'trois', 'quatre', 'cinq']}
```

```
len(e_dict)          # >>> 2
e_dict['français'] # >>> ['un', 'deux', 'trois', 'quatre', 'cinq']
e_dict['français'][2] # >>> 'trois'
```

Deleting an element

```
del a_dict[3]
print(a_dict)          # >>> {1: 'a', 2: 'b', 4: 'd', 5: 'E'}
a_dict.pop(1)          # >>> 'a'
print(a_dict)          # >>> {2: 'b', 4: 'd', 5: 'E'}
```

Membership

```
2 in a_dict            # True
3 in a_dict            # False
```

List out the items, keys or values

```
a_dict.keys()          # >>> dict_keys([2, 4, 5])
a_dict.values()         # >>> dict_values(['b', 'd', 'E'])
a_dict.items()          # >>> dict_items([(2, 'b'), (4, 'd'), (5, 'E')])
```



Like Us On Facebook!
The PCLinuxOS Magazine
PCLinuxOS Fan Club



PCLOS-Talk
Instant Messaging Server

Sign up TODAY! <http://pclostalk.pclosusers.com>



CHIMPBOX

The chimpbox packs a punch. Zero noise, small footprint and low power usage.
<http://chimpbox.us>

The PCLinuxOS Magazine Special Editions!



Get Your Free Copies Today!

PCLinuxOS Recipe Corner



*from the kitchen of
youcantoo*

Philly Cheese & Ground Beef Casserole

Ingredients

1 1/2 lb lean (at least 80%) ground beef
8 oz sliced mushrooms
1 teaspoon salt
1/2 teaspoon pepper
8 slices (1 oz each) provolone cheese
2 tablespoons butter or margarine
2 large onions, halved and thinly sliced into wedges
2 medium red bell peppers, cut into strips
2 cloves garlic, finely chopped
1 can (16.3 oz) Pillsbury™ Homestyle original biscuits

Directions

1. Heat oven to 350°F. Spray 13x9-inch (3-quart) baking dish with cooking spray.
2. In 12-inch skillet, cook beef, mushrooms, salt and pepper over medium-high heat 7 to 9 minutes, stirring frequently, until beef is thoroughly cooked; drain. Place in baking dish. Arrange cheese over beef mixture, overlapping slices if needed.
3. In same skillet, melt butter over medium-high heat. Add onions and bell peppers. Cook over medium-high heat 3 to 5 minutes, stirring frequently,

until peppers are crisp-tender. Stir in garlic; cook 1 to 2 minutes longer. Spoon over cheese in baking dish.

4. Separate dough into 8 biscuits. On lightly floured surface, pat biscuits into 5-inch circles. Arrange biscuits over vegetable mixture.

5. Bake 35 to 40 minutes or until biscuits are golden brown on top.



**The PCLinuxOS
Magazine**

**Created with
Scribus**



PCLinuxOS.nl
Netherlands



DONATE
TODAY



Screenshot Showcase



PCLinuxOS Magazine

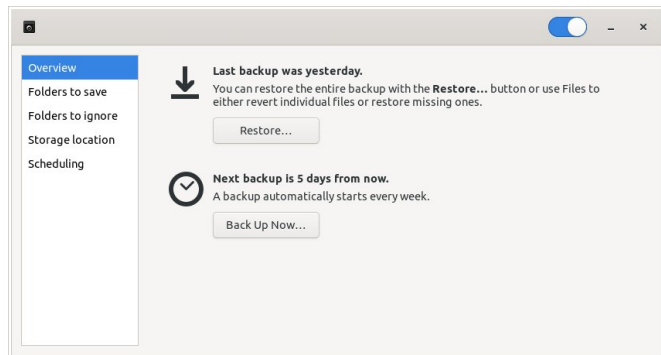


Repo Review: Déjà Dup Backup

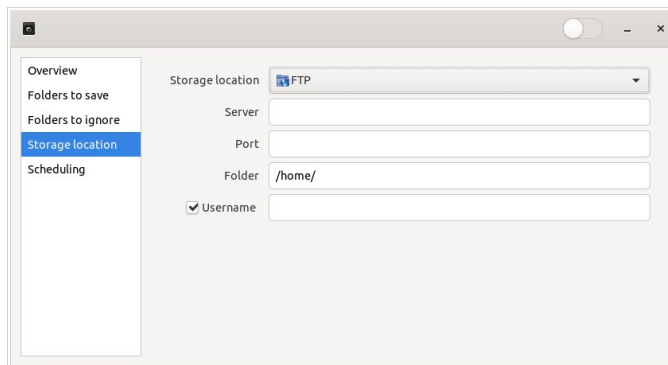
by CgBoy

Déjà Dup is an easy to use GUI frontend for the command-line Duplicity backup tool. It offers support for doing automatic, incremental, and optionally encrypted backups to cloud services, remote servers, or local folders.

Déjà Dup has a simple, well designed interface. The main Overview tab is where you can manually start and restore backups from, and it'll tell you when the last backup was performed, and when the next one is scheduled for.

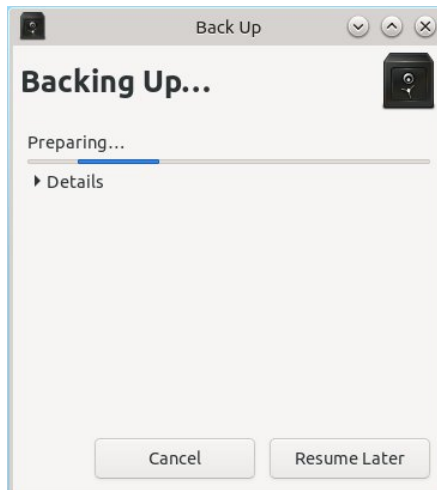


From the next two tabs you can select which folders you want to be backed up, and which ones to ignore. From the Storage location tab you can set where the backups will be saved. You can choose a cloud service (Amazon S3, or Rackspace Cloud Files), a remote server (FTP, SSH, WebDAV, or Windows Share), or a local folder. If you have an external storage device connected, such as an external hard drive or a flash drive, it will also appear as a storage location. Unencrypted backups are saved as diffstar.gz files, and encrypted backups are diffstar.gpg.

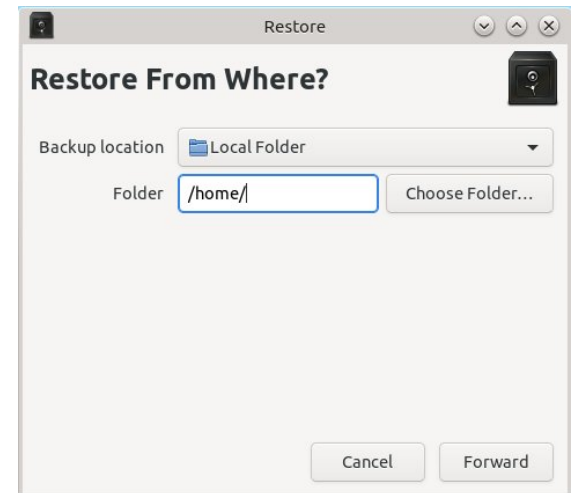


The switch in the top right corner of the window enables the automatic scheduled backups. From the Scheduling tab, you can set Déjà Dup to perform backups every day, or every week, and for how long to keep the old backups around before deleting them. Déjà Dup uses its own scheduling service, rather than cron.

Backups can also be started at any time from the Overview tab. When you first start a backup, you will be asked if you want to enable password encryption or not.



Restoring backups is easy, and as I said earlier, can be done from the Overview tab. You will have to select the location the backups were saved to, the date to restore from, and finally, where to restore the backups to.



Summary

Déjà Dup is not the most advanced backup application available for PCLinuxOS, but it's really very simple to set up and use, and I haven't had any problems with it. It would be nice if you could have a bit more control over what time the scheduled backups start, though, rather than just daily or weekly.



PCLinuxOS Family Member Spotlight: jim2u71

As told by YouCanToo

What is your name/username?

Harold Williams, username "jim2u71"

How old are you?

I will be 76 in August 2019.

Are you married, single?

Married for 51 years to my wonderful life partner

How about Kids, Grandkids (names and ages)?

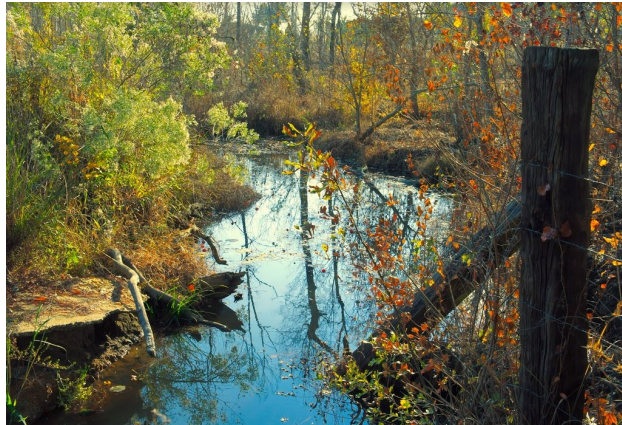
We have two children that are grown and one grandchild.

Do you have pets, what is your favorite?

My wife has a cat that she dotes on. Every time another cat comes around to fight she runs out and screams to run the other cat off.

Are you retired, still working and if working, what do you do?

I am retired for several years. I have taught school, sold paint, worked offshore in the oil production industry for 19 years, worked in the cell phone industry (can you hear me now?), and computerized a couple of loggers' offices. Now, I mostly volunteer at the local drug/alcohol rehab center and help Christians on Mission construct wheelchair ramps for those in need that can't afford to hire it done. I am the local handyman for the rehab center. It houses up to 19 female clients in a donation only type facility.



Where do you call home? What is it like? IE: weather, scenery

Home is in rural south Alabama. Hot and humid in the summer and gets below freezing a few days each winter. In 2018 the first frost didn't occur until November. Lots of timber and living in the country beats living in the city anytime. A lot of people in this area work in the timber industry.

Where did you go to school and what is your education level?

I have a BS degree in Education from the University of Southern Mississippi in Hattiesburg, MS.

What kind of things you like doing? hobbies, travel, fishing, camping?

My wife and I volunteer with Alabama Disaster Relief when needed and with another construction mission trip once a year. We have done this for several years





now. My personal hobbies are golf and computers. Last year I built a computer for the first time, just to see if I could. A few bumps but I got it done. My daughter uses it now. Oh, I almost forgot, I keep up with the golf handicaps for several of the guys that play golf at our local course.

Why and when did you start using Linux?

I first looked at trying to use Linux in 2007. The first one that I really liked was Mepis. It later got so unstable that I went distro hopping and haven't used anything on a day-to-day basis except PCLOS for several years now. Mostly I use KDE but the Xfce version is quite a bit quicker, it seems.

What specific equipment do you currently use with PCLOS?

Intel Core i5 Sandy Bridge desktop that is several years old now. I recently upgraded the ram to 12 GB and installed a 500 GB SSD. I multiboot PCLOS KDE and Xfce, along with Windows 7. I also have an Dell Inspiron 1545 laptop that only has PCLOS KDE and Xfce installed.

Do you feel that your use of Linux influences the reactions you receive from your computer peers or family? If so, how?

Most people don't know or care about what Linux is. They only want to see their familiar Windows when they boot up their computer. If something doesn't

come up like it normally does then they have no idea what to do nor do they care to learn. Just give them their one email account and Facebook or Instagram and they will be happy as long as it works.

What would you like to see happen within PCLOS that would make it a better place. What are your feelings?

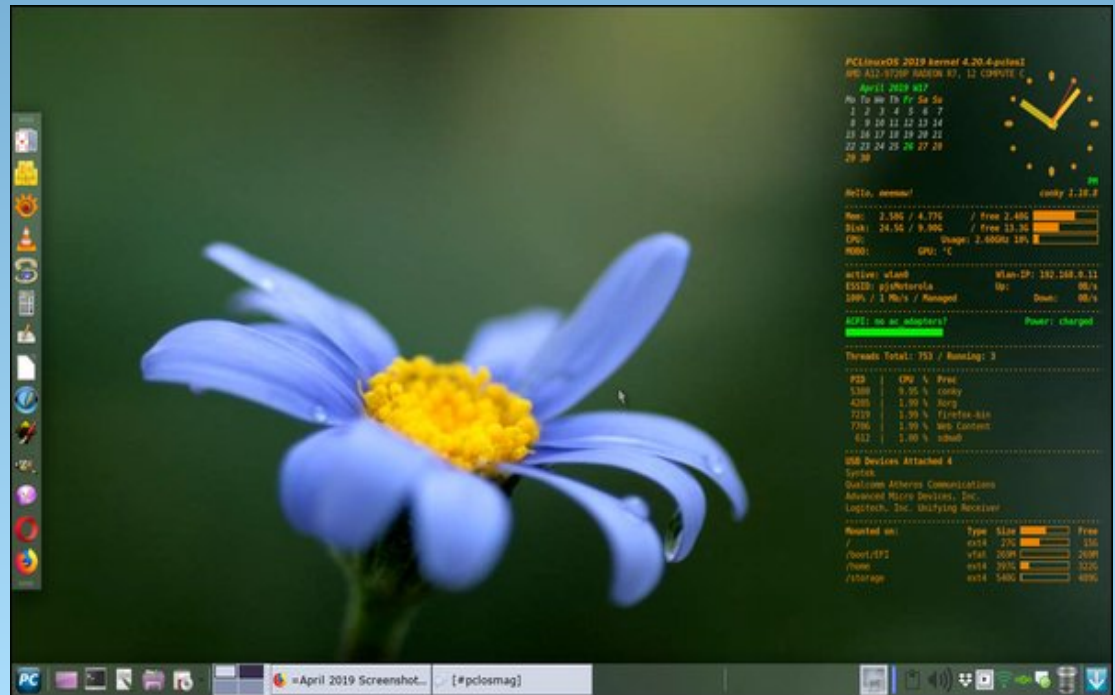
PCLOS is great. I can't imagine the work that is put into keeping this distro going. Keep up the good work! Sometimes it seems that some of the older admins on the forum lose patience with me when I post something stupid. But all in all it is a great

forum. Thanks to those that keep it going.

PCLinuxOS Family Member Spotlight is an exclusive, monthly column by YouCanToo, featuring PCLinuxOS forum member. This column will allow "the rest of us" to get to know our forum family members better, and will give those featured an opportunity to share their PCLinuxOS story with the rest of the world.

If you would like to be featured in PCLinuxOS Family Member Spotlight, please send a private message to youcantoo, parnote or Meemaw in the PCLinuxOS forum expressing your interest.

Screenshot Showcase



Posted by Meemaw, on April 26, 2019, running Xfce.

The Ruby Programming Language: Writing A Ruby Program, Part 2

by phorneker

In the last issue, we discussed the basic structure of a Ruby program, integers and floating point numbers, logical and mathematical operators, strings, and the if/then/else statement.

Control Statements

Let us look at the if/then/else fragment from last month.

```
if (number != 0.0) then
  validnumber = true
else
  if (name == "0.0") || (name == "0") then
    validnumber = true
  else
    validnumber = false
  end
end
end
```

The `elsif` keyword is a shortened form of the `else if` keywords in a control statement. Hence, we could have written this fragment of code as:

```
if (number != 0.0) then
  validnumber = true
elsif (name == "0.0") || (name == "0") then
  validnumber = true
else
  validnumber = false
end
end
```

Ruby provides a `unless` statement that executes code if the condition associated with the `unless` statement is `false`. The fragment of code here would then be written as:

```
unless (number = 0.0)
  validnumber = true
else
  if (name == "0.0") || (name == "0") then
    validnumber = true
  end
end
```

```
else
  validnumber = false
end
end
```

Note that the keyword **then** is optional for the **unless** statement, yet it is *required* for the **if** statement. This makes sense from a readability perspective.

One interesting note on the **if** and **unless** statements is that they can be used as modifiers to other statements, such as **puts** and **print**.

Consider this block of code:

```
if validnumber then
  puts "You have typed in the number #{number}"
else
  puts "Sorry, what you typed in is not a valid number"
end
```

We could write this fragment of code as follows:

```
puts "You have typed in the number #{number}" if validnumber
puts "Sorry, what you typed in is not a valid number" unless
  validnumber
```

...and it would be easy to see the logic behind this fragment of coding, not to mention that the same thing was accomplished in **only two lines of code** rather than five in the fragment before this.

For this fragment of code, **only one of these statements** will be executed, and which one depends on whether **validnumber** is **true** or **false**.

But, what if **validnumber** is set to **nil**? In Ruby, the **boolean** data type for variables is not truly binary. The existence of **nil** provides a third condition to test for. Technically, **nil** is neither **true** nor **false**. The above fragment will work correctly if **nil** is assigned to **validnumber** as the **unless** statement tests to see if the condition specified is **false**.

For the current version of Ruby, **nil** and **false** are considered to be the same for the **unless** statement. *However, this may change in future versions of Ruby.*

Another Way to Code If/Then/Else Statements

Let us take another look at the previous fragment of code:

```
unless (number = 0.0)
  validnumber = true
else
  if (name == "0.0") || (name == "0") then
    validnumber = true
  else
    validnumber = false
  end
end
end
```

As with most any statement, method or variable, the **if/then/else** statement can return a boolean value that can be assigned to a variable. Hence, this fragment of code can be written as follows:

```
unless (number = 0.0)
  validnumber = true
else
  validnumber = if (name == "0.0") || (name == "0")
end
end
```

...and while we are at it, we can take **this** one step further.

```
validnumber = if (number != 0.0) || (name == "0.0") || (name == "0")
```

We have managed to accomplish the same task in this **one** line of code whereas previously, we did it in **nine** lines of code in the original fragment.

Now, let us revisit the program from the first article (the one called **numbertest.rb**):

```
#!/usr/bin/ruby
```

```
print "Please type in a number "
name = gets
number = name.to_f
if (number != 0.0) then
  validnumber = true
else
  if (name == "0.0") || (name == "0") then
    validnumber = true
  else
    validnumber = false
  end
end
```

```
end
if validnumber then
  puts "You have typed in the number #{number}"
else
  puts "Sorry, what you typed in is not a valid number"
end
```

Given what we now know, we can rewrite the program as follows:

```
#!/usr/bin/ruby
```

```
print "Please type in a number "
name = gets
number = name.to_f
validnumber = if (number != 0.0) || (name == "0.0") || (name == "0")
puts "You have typed in the number #{number}" if validnumber
puts "Sorry, what you typed in is not a valid number" unless validnumber
```

This is one example of the productivity that can be achieved with the Ruby programming language, though the former is far more readable if you are used to programming in languages like Pascal.

Here, we managed to reduce the size of the source code example from **19 lines** to a mere **8 lines**, and still achieve the same result.

Case Statements in Ruby

The **case** statement is an extension of the nested **if/then/else** statements. The structure of a case statement is as follows:

```
case <parameter>
when <value1>

  <block of code to be executed when parameter is value1>

when <value2>

  <block of code to be executed when parameter is value2>

.
.
.

else
```

<block of code to be executed when all else fails>

end

For any **case** statement, there is the **end** statement to tell Ruby that this is the end of the **case/when/else** statement, and *at least one* **when** block and *exactly one* **else** block is required, with the **else** block placed at the end of the **case/when/else** statement just prior to the **end** statement. The **else** block contains code to be executed **when** all conditions contained in all **when** statements are false (i.e. when all else fails).

Conditions, shown here with the <valuex> labels can be a single value, or a range of values.

Let us say, we want to define a range from 90 to 100. In Ruby, this range is coded as

90 .. 100

The spaces between the two dots and each of the numbers are required to tell Ruby that we are defining a range of values.

On to the example.

One good use of the **case** statement would be to create a function that returns a letter grade representing a test score (from 0 to 100). Assuming the standard grading scale used for many years, i.e. that passing starts at 60, we would then define ranges as follows:

90 – 100	A
80 – 89	B
70 – 79	C
60 – 69	D
all others	F

This is a rather simplified form of a typical grading scale. We could define this further.

98 – 100	A+
93 – 97	A
90 – 92	A-
87 – 89	B+
83 – 86	B
80 – 82	B-

77 – 79	C+
73 – 76	C
70 – 72	C-
67 – 69	D+
63 – 66	D
60 – 62	D-
all others	F

I know that in practice, some schools consider 70 to be passing instead of 60, but for purposes of demonstrating the **case** statement, this grading scale will be used.

As we can see here, the ranges here are clearly defined. For each **when** statement, we could execute any code we wanted. Here, we will simply print out the appropriate **grade** for the grade value passed to the **case** statement.

```
grade = <value assigned>
case grade
when 98 .. 100
  puts "A+"
when 93 .. 97
  puts "A"
when 90 .. 92
  puts "A-"
when 87 .. 89
  puts "B+"
when 83 .. 86
  puts "B"
when 80 .. 82
  puts "B-"
when 77 .. 79
  puts "C+"
when 73 .. 76
  puts "C"
when 70 .. 72
  puts "C-"
when 67 .. 69
  puts "D+"
when 63 .. 66
  puts "D"
when 60 .. 62
  puts "D-"
else
  puts "F"
end
```

More operators

The **when** portion of the **case/when/else** statement is equivalent to **<value> === <range>**

which tests **value** to see if it is contained within **range**, and will return **true** if the contents of **value** are included in the set of values represented by **range**.

In contrast, **<value> == <range>** simply tests to see if the contents of **value** are exactly the same as the contents of **range**, which by definition will always return **false** as individual values will never be the same as a range of values, *even if the range contains only one value*.

On the other hand, **<value> = <range>** simply assigns the contents of **range** to **value** *only if value happens to be the name of a variable*. If **value** happens to be a constant or a data type incompatible with the type being assigned, an error message would result, as such an assignment would not make sense.

Likewise, **<value> !== <range>**, as one would expect, tests to see if the contents of **value** are **not** included in the set of values represented by **range**.

Looping in Ruby

In C, C++, and even BASIC, we have the **for** statement to execute a block of code inside a loop. Ruby has this function, too. The **for** statement requires a range of numbers and a variable for parameters.

What sets Ruby apart here is that the range **does not have to be all numbers**. Ranges specified in Ruby *can be of mixed data types*, a feature not found in traditional programming languages.

The **for** statement in Ruby is coded as follows:

```
for <variable> in (<range>)
<block of code placed here>
end
```

The things that are mandatory are **variable** (which does not have to be used in the loop, but most likely should depending on the code being executed), and the **range**, which can be almost anything you could imagine, as long as it makes up a list, including a list of ranges.

For example:

```
for color in (black, blue, green, cyan, red, magenta, yellow,
white)
puts(color)
end
```

will display the names of the basic colors in an RGB palette.

(You can emulate this in Pascal by defining a custom data type in the variable declaration sections.)

The **while** statement is very much the same in Ruby as it is in C and C++, i.e.

```
while <condition>
<body of code placed here>
end
```

C, however, requires the keyword **do**, whereas **do** is *optional* in Ruby. It is there for those of us who are used to writing in C.

For the **while** statement to be executed at least once, the value of **condition** must be **true** at some point. If the value of **condition** is always **false**, the code inside the loop will **never execute**.

Likewise, if the value of **condition** is always **true**, then you have just coded an **infinite loop**.

The **while** statement executes code inside the loop when the value of **condition** is **true**, whereas the **until** statement executes code inside the loop when the value of **condition** is **false**.

```
until <condition>
<body of code placed here>
end
```

which is basically the equivalent to

```
while !<condition>
<body of code placed here>
end
```

Did you know that **while** and **until** loops can be coded a second way?

```
begin
<body of looped code placed here>
end until <condition>
```

is the same as

```
until <condition>
<body of code placed here>
end
```

...and the same follows for the **while** statements

```
begin
<body of looped code placed here>
end while <condition>
```

is the same as

```
while <condition>
<body of code placed here>
end
```

Give Looping a Break

Ruby code inside the **while** and **until** statements have the potential to become what we call **infinite loops**, or blocks of code that continue to execute endlessly (or until something causes the program to crash, such as a **kill** command from the superuser).

Ruby offers a generic **loop** statement that does just that and does not require any parameters to boot.

```
loop
<body of code placed here>
end
```

The break statement provides a way for the program to exit from such a loop.

The infinite loop is useful in some instances, such as implementing the main loop in a command line interpreter.

It is a better programming practice to use the while and until statements to create the loop, then use a flag to indicate when to exit that loop, rather than relying on a generic loop statement and then placing a break statement to end the loop.

The If/Then Statement Redux

Ruby provides yet another form of the if/then statement. This one does not

include the keywords if or then, but instead consists of two operators, namely “?” And “:”, and can be used in place of a variable, or assigned to a variable.

If you have programmed in C or C++, this form should be familiar to you. Let us go back to a previous code fragment.

```
if validnumber then
puts "You have typed in the number #{number}"
else
puts "Sorry, what you typed in is not a valid number"
end
```

This can be written as follows:

```
puts validnumber ? "You have typed in the number #{number}" :
"Sorry, what you typed in is not a valid number"
```

(Note: This is all one line of code.)

The first argument after the “?” is what is displayed if the value of **validnumber** is true, otherwise the second argument, the one after the “:” is displayed.

Program Flow

As with any interpreter, Ruby programs normally start at the beginning of the source file and flow down to the end of the source file. Unlike C and C++, there is no **main** function or statement. Unless functions and methods have been defined in the source code, what would be considered the **main** function is the first line of Ruby code **not contained in a separate block**.

Consider the code contained in our **numbertest.rb** (after simplification from what we discussed earlier)

```
#!/usr/bin/ruby

print "Please type in a number "
name = gets
number = name.to_f
validnumber = if (number != 0.0) || (name == "0.0") || (name == "0")
puts "Sorry, what you typed in is not a valid number" unless
validnumber
puts "You have typed the number #{number}" if validnumber
end
```

The Ruby Programming Language: Writing A Ruby Program, Part 2

This program starts at the **print** statement and ends where it says (appropriately) **end**.

If we were to define a function (or a method) in this source code file, the program would still begin at the first **print** statement as the code between **print** and **end** has not been enclosed inside a block.

What Exactly Are Blocks?

Before we continue, we need to know what exactly is a **block** with regards to Ruby.

Think of a **block** as a fragment of Ruby code enclosed between two statements, one marking where execution should start, and one marked **end** where execution should end.

A generic block in Ruby is coded as follows:
begin

<code to be executed here>

end

Simple enough, isn't it? Most likely, there is a control statement such as a **for**, **if/then/else**, **while**, **until**, or **unless** that replaces the **begin** in this statement. (We have already seen examples of this in this article.)

Blocks are also found in **when** statements contained in **case** statements. In the case of the **when** statement, execution of that block of code continues until another **when** statement is encountered, an **else** statement is encountered, or when it comes to the **end** statement. In any case, execution jumps to the code *after* the **end** statement that ends the given **case/when/else** statement.

Where does a Ruby program begin?

As Ruby does not explicitly have a **main()** function like in C and C++, execution of a Ruby program **normally** begins at the beginning of the source file at the first executable statement **not contained in a block**. (This is why I defined what a block is before writing this section.)

Ruby, however, does not always start program execution this way. You can specify where in the source code file Ruby is to begin execution and where the program ends by using the **BEGIN** and **END** statements.

How is **BEGIN** different from **begin**? You type **BEGIN**, in *all capital letters*. Everything in Ruby is **case sensitive**, (not unlike C, C++ and Java), hence **BEGIN** is not the same as **begin**.

Language Note: Pascal, unfortunately, does not make this distinction as the language was developed before case sensitivity was even a consideration.

Hence, we could write our example program as:

```
#!/usr/bin/ruby
```

```
BEGIN
print "Please type in a number "
name = gets
number = name.to_f
validnumber = if (number != 0.0) || (name == "0.0") || (name == "0")
puts "Sorry, what you typed in is not a valid number" unless
validnumber
puts "You have typed the number #{number}" if validnumber
END
```

...and still get the same result. **Most of the time, we will not need to do this when writing simple Ruby code.**

Error Handling with Rescue/Retry

Sometimes, when executing a program and you enter a value that generates a runtime error, it is not appropriate for Ruby to halt execution of the program simply because an invalid value was entered.

To handle this potential situation, we define a block of code as follows:

```
rescue
```

<do error handling code here>

```
retry
end
```

Anything placed here is executed **only if a runtime error** occurs during execution of a program, such as an attempt to divide by zero, or asking for the square root of negative one (in the latter case, this would not generate a runtime error **if we define a module to handle imaginary numbers** as the square root of **-1** is **i**. But that is a topic for another article.)

Otherwise, execution of the code jumps over this block of code.

The Ruby Programming Language: Writing A Ruby Program, Part 2

If this code **does get executed**, control of the program returns to the point where the runtime error occurred.

If you are familiar with some versions of BASIC, this is similar to the **ON ERROR GOTO** statement with the **RESUME** statement at the end of the error handler.

Methods ARE Functions in Ruby

What is the difference between a **method** and a **function**? In Ruby, there is no difference. The difference here depends on whether you are used to object oriented programming, or structured programming.

What I refer to as a **function** in Ruby is actually a **method**. This is because I was educated in the **structured way of programming** (before there was such a thing as object oriented programming), *where there is only one point of entry in a program, procedure or function, and one exit point for the same program, procedure or function.*

While it is possible to write Ruby code in a structured manner, it is not as efficient or productive as writing code in Ruby code in an object oriented manner, as the language was designed as an object oriented language.

As I learn Ruby, I am rediscovering programming techniques that were used for writing applications for vintage computers from the late 1970s through the early 1990s, where memory usage and disk space were at a premium. In addition, I am also learning about what makes object oriented programming languages (such as Ruby) a popular choice for software and internet developers.

Ruby **requires that functions be defined before they are used**. This makes sense as it is true of just about every other programming language available today.

But that was not always the case.

Before high level languages (i.e. languages such as FORTRAN, Pascal, C, C++, PL/I (which is making a comeback) and BASIC came into existence, programs were developed in assembly language, which meant learning about individual microprocessors and their instruction sets.

Once a program started at a specific address in memory, execution was linear until an instruction to jump was executed, with the location being the next two to four bytes in memory.

If functions were to be defined before they were used, you would have to **know**

the exact location in memory where the program begins, and place a jump statement at the beginning of where the processor expects the program to begin so the processor can start the program at the proper location.

As a result, to increase efficiency of the machine language program, functions and procedures in assembly language were typically placed after the main program.

In Ruby, methods (or functions) are coded as follows:

```
def <function name>
  <place code here>
end
```

That's it. Typically, code is indented (two or three spaces for readability) to show that this code is a part of a defined function, and not the actual Ruby program.

The **return** keyword is used when a method is to return a value to the calling program or method. *Methods always return some kind of value. If the **return** keyword is not used in a defined method, the method will return **nil**, meaning no value is returned, and its boolean value is considered to be neither true nor false for most boolean functions and operators.*

But wait. There's more!

Ruby provides two more keywords typically used in loops, namely **redo** and **next**.

The keyword **redo** tells Ruby to restart the current loop block using the same value in the iterator variable.

The keyword **next** tells Ruby to skip the code between the statement containing **next** and the end of the loop, then increment the iterator variable by one.

These are those things that are not really needed most of the time, but are available in case we come to a situation where such things are required.

I wanted to get to the string manipulation in Ruby, but in my research, I found that there is more than enough material (including many Ruby methods) to warrant an article **dedicated to this topic**.

As Ruby is an object oriented language, I would need to get to Classes (where functions and procedures are called **methods**) first.

What I just demonstrated is the use of Ruby as a language for structured programming.

Now we can guess what I am going to have to cover next.

The phrase "But wait. There's more!" became one of my favorite expressions after watching Jaw Tooth's railroad videos on YouTube. In most of his videos there is always additional material added to the video. This material is introduced with Jaw Tooth saying "But wait! There's More!" followed by the additional material.

**PCLOS-Talk**
Instant Messaging Server
Sign up TODAY! <http://pclostalk.pclosusers.com>



*Looking for an old article?
Can't find what you want? Try the*

**PCLinuxOS Magazine's
searchable index!**

The **PCLinuxOS** magazine

**PATREON**

**DONATE
TODAY**

*Help PCLinuxOS
Thrive & Survive*

Screenshot Showcase



Posted by mutse, on April 3, 2019, running Mate.

FREE Linux Help Books

by Paul Arnote (parnote)

Anyone who has spent any amount of time using Linux already knows that it can sometimes be a daunting task to find help on Linux topics. None of us were born knowing how to run Linux, nor any other operating system. Fortunately, RTFM (read the freaking manual) isn't used in the PCLinuxOS forum. Unfortunately, RTFM is SOP (standard operating procedure) in the forums of other Linux distros. In the PCLinuxOS forum, members are eager to actually help with the situation or problem at hand. We were all once beginners to Linux, after all.

Most everyone can find help for the commercially available operating systems, thanks largely to their level of market penetration. Walk into any bookstore, and the number of "help" books for the commercially available operating systems dwarfs the number available for Linux or Unix. Under Linux, it can be more challenging to find the necessary information. Just as Linux is seen as being "home grown," the documentation can be even more "home grown." Someone, somewhere, has probably taken the time to write up some helpful "how to" covering just about everything you might want to know. But the real challenge is finding it. Many of these helpful texts exist on obscure, remote servers, and often on web pages that are little more than personal web pages.

Ask 10 (or 100) different Linux users what their favorite "help" resources are, and you'll likely get 10 (or 100) strikingly different answers. Of course, each user will have distinct and unique needs, depending not only on how they use their computer, but also what those needs are.

What actually got me thinking about Linux help books was a recent [list](#) of over 200 **free** Linux help

books that appeared on a Ubuntu focused site. As I went through the list, I found some books to be less than helpful, others too "technical," quite a few broken links to guides that have faded off into the great digital ethernet sunset, quite a bit of content that hadn't been updated in many years, and yet others that are very helpful. There are even duplicates of book titles on the list. While it's admirable to try to make as complete of a list as possible, the sheer volume of the number of titles on that list can be overwhelming, making it even more difficult for users to find the help they are looking for. Plus, even with over 200 books listed, I found that two of my favorites were omitted from the list.



So, let's take a look at 10 (or so) free Linux books that **I** think are actually helpful to the largest percentage of Linux users, old and new. Some appear on the aforementioned list. But others –

including my two favorites – are in addition to those that appear on the list of 200+ free Linux books. Of course, feel free to browse the list at the previous website.

Advanced Bash-Scripting Guide

by Mendel Cooper

This is one of the two books that I was surprised to see omitted from the list. This book is also one of my go-to reference manuals. This book was instrumental in me learning bash scripting. It remains a frequently visited reference as I continue to learn more and more about bash scripting. The most current version (Revision 10) was released March 10, 2014. It is available both as a [PDF](#) version (916 pages), and an online [HTML](#) version.

Linux Guide

Wikibooks

If you're looking for a decent beginner's guide, you could do a lot worse than Linux Guide. Formerly called "Linux For Newbies," Linux Guide helps a new user understand what Linux is, how to install Linux, how Linux works, some basic Linux commands, explores most of the more common desktops and window managers, installing software, how to manage Linux, set up networking, and then walks the user through performing some common tasks (like using Audacity to make a MP3, or burning a CD/DVD).

As with anything "wiki" based, it is continually evolving. It also means that you can contribute to the project, supplying information that is currently lacking. To be able to edit, you will need to create an account and be logged in. Unfortunately (or

fortunately for one of our beloved PCLinuxOS community members), there is currently no entry in the wiki for PCLinuxOS (hint, hint). There is also no entry for the Openbox, Mate or Trinity desktops or window managers.

You can view the [HTML](#) version of the Linux Guide here. Once there, you can create a PDF ebook of Linux Guide. At the time of the writing of this article, the PDF convertor was down and unavailable.

GNU/Linux Desktop Survival Guide

Graham Williams

Despite emphasizing Ubuntu and Debian versions of Linux, there's something for every Linux user in this book. Started in 1995, it was last updated in April, 2019. To say that this book was comprehensive would be an understatement. Every Linux user, from a noob to a seasoned Linux user, will find helpful information inside its covers. There is a [HTML](#) version available. There is a [PDF](#) version of the book available for a \$40 (U.S.) "donation."

GNU Awk

Arnold D. Robbins

Anyone who has messed around, even briefly, with bash programming has seen the Awk command crop up periodically. I have to be honest ... I didn't know how extensive the use of Awk could be. Awk can be viewed as a whole programming language! Well, this 566 page book ([PDF](#)) details all of the ins and outs of Awk. Prepare yourself for a deep, deep trip into the use Awk.

A Byte of Vim

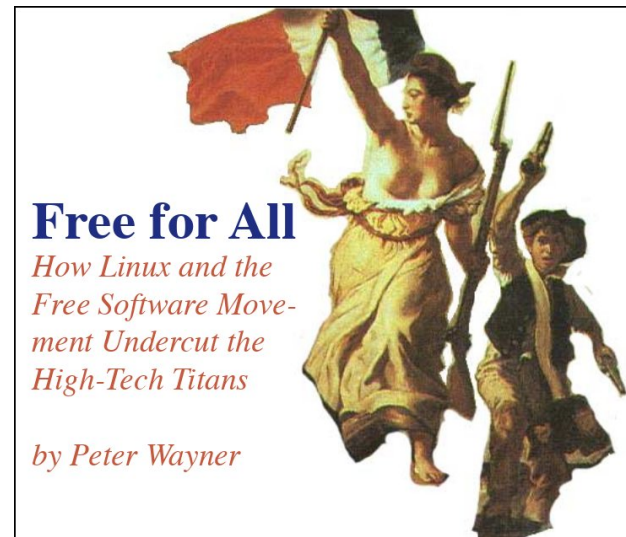
Swaroop

There are text editors. And then there is Vim. You either love it, learn to love it, or just hate it. But, if

you spend any amount of time at all using Linux, you will probably want to learn to use Vim.

It is a very powerful text editor, and it is unlike any other text editor you've ever used. Vim exists as a command line tool, and as a GUI tool. In either case, the commands are the same. Vim uses `:` to preface commands. For example, to quit a Vim session, you type `:q`.

There are few text editors (maybe Emacs) that come to mind that require an extensive manual or how-to. To learn to use Vim effectively, you will want to read this 89 page book, presented as a free [PDF](#).



Free for All How Linux and the Free Software Movement Undercut the High-Tech Titans

Peter Wayner

Well, this one isn't so much of a Linux "help" book, but any Linux user should find it an interesting read. It is the classic story of the Davids of the world going up against the Goliaths of the world. Of course, the role of the Davids is played by the authors of free, open source software. The role of the Goliaths is

played by the multinational technology behemoths, such as Microsoft. It is a well written account of the battle(s), written from the perspective of the free and open source software community. You can download your free [PDF](#) copy here (351 pages). Originally released in 2000, this 2002 "modded" version is as good of a read for FOSS advocates today as it was when it was written nearly two decades ago.

The Ultimate Linux Newbie Guide

Alistair J. Ross

The Ultimate Linux Newbie Guide is a continually updated [website](#), presented in "chapters" as a book is. If you're just getting started with Linux, this "book" may help take away some of the anxiety of getting started using Linux. It starts with explaining what Linux is, and walks the user through installing Linux and using it every day. Even though it is quite Ubuntu-centric, users of non-Ubuntu versions of Linux still stand to gain considerable knowledge from its pages.

Bash Guide for Beginners

Machtelt Garrels

Currently version 1.11, this 173 page ebook ([PDF](#)) was last updated in 2008. But face it, not much has changed with how to script for bash since 2008, either. This excellent guide will help the beginning bash programmer to find their way through the maze of learning how to write bash scripts. This book is about as complete a guide as commercially available programming books. Finishing this book, and with a reasonable amount of practice, anyone should be able to become a skilled bash programmer.



Like Us On Facebook!
The PCLinuxOS Magazine
PCLinuxOS Fan Club



Running Linux

Matthias Kalle Dalheimer, Terry Dawson, Lar Kaufman, Matt Welsh
Copyright © 2003 O'Reilly & Associates, Inc.

This is the 4th edition of this helpful book, available as a free 614 page [PDF](#). The current version of this book is the 5th edition, published in 2009, which you can purchase from Amazon and other book retailers. The authors have attempted to take a refreshing, distro-neutral approach to teaching the reader how to run and use Linux. While there are some aspects of the older edition that are (obviously) outdated, much of the information helps form a solid Linux foundation for Linux users.

DESTINATION LINUX
LINUX IS OUR PASSION

Does your computer run slow?

Are you tired of all the "Blue Screens of Death" computer crashes?



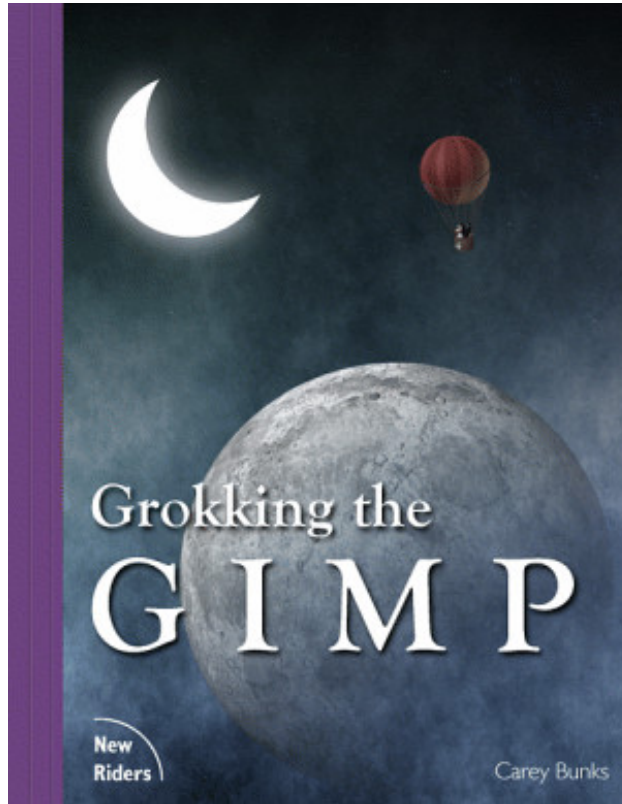
Are viruses, adware, malware & spyware slowing you down?

Get your PC back to good health TODAY!

Get



Download your copy today! FREE!



Grokking the GIMP

Carey Bunks

If you're looking for the definitive "how to" book on the GIMP, look no more. Even though this was written in 2000 for older versions of the GIMP, the teachings in this book are just as applicable today as they were when it originally was released. This book is also the second title that I was surprised to not find on the 200+ title book list. If it's detailed, structured and orderly information on how to best use the GIMP is what you're looking for, then you can't do any better than this book. Even after nearly two decades, this book withstands the tests of time. You can view the book online as an [HTML](#) file. Alternatively, you can buy a printed, hard copy version of the book on [Amazon](#) ... for \$42 (U.S.).

Summary

There you have it. Ten Linux books that I think every Linux user should have on hand, or at least have access to. It makes sense to have reference material on hand should you ever need it, even if you don't find yourself using it all that often. Chances are, too, that you may find yourself using the reference materials more than you think.

Through it all, don't forget some other useful resources from The PCLinuxOS Magazine. Just about anything from the magazine's Special Edition [page](#) should be considered for your Linux library, especially the Command Line Interface Special Edition.

There is another list of books you might want to take a look at, too. Over at [Linuxtopia](#), there is a long list of available books that you might want to consider adding to your catalog of available resources. From what I can tell, all books listed are in HTML format. Be careful, though. Its outward appearance looks like a page that is sporadically updated, and in fact, may have been abandoned. Most of the stuff there is quite old (Android 6? CentOS 6?), so some of the information may have radically changed in newer versions of the software covered. The copyright statement at the bottom of the page says 2005-2010, but Android 6 came out quite in October 2015, some time after the 2010 date in the copyright statement.

Just remember that knowledge is power. So, what are you waiting for? Let's power up!



PCLinuxOS Bonus Recipe Corner



Cheesy Ground Beef Manicotti

Ingredients

14 uncooked manicotti shells
1 lb lean (at least 80%) ground beef
1 large onion, chopped (1 cup)
2 cloves garlic, finely chopped
1 jar (26 to 30 oz) tomato pasta sauce (any variety)
2 boxes (9 oz each) frozen chopped spinach, thawed
2 cups small curd cottage cheese
1 can (8 oz) mushroom pieces and stems, drained
1/3 cup grated Parmesan cheese
1/4 teaspoon ground nutmeg
1/4 teaspoon pepper
2 cups shredded mozzarella cheese (8 oz)
2 tablespoons grated Parmesan cheese

Directions

1. Cook and drain manicotti as directed on package using minimum cooking time (cooking for the minimum time helps prevent the shells from tearing while filling).

2. Meanwhile, in 10-inch skillet, cook beef, onion and garlic over medium heat 8 to 10 minutes, stirring occasionally, until beef is brown; drain. Stir in pasta sauce.

3. Heat oven to 350F. Spray 13x9-inch glass baking dish with cooking spray.

4. Squeeze thawed spinach to drain; spread on paper towels and pat dry. In medium bowl, mix spinach, cottage cheese, mushrooms, 1/3 cup Parmesan cheese, the nutmeg and pepper.

5. In baking dish, spread 1 cup of the beef mixture. Fill manicotti shells with spinach mixture. Place shells on beef mixture in dish. Pour remaining beef mixture evenly over shells, covering shells completely. Sprinkle with mozzarella cheese and 2 tablespoons Parmesan cheese.

6. Cover and bake 30 minutes. Uncover and bake 20 to 25 minutes longer or until hot and bubbly.



A magazine just isn't a magazine without articles to fill the pages.

If you have article ideas, or if you would like to contribute articles to the PCLinuxOS Magazine, send an email to:

pclinuxos.mag@gmail.com

We are interested in general articles about Linux, and (of course), articles specific to PCLinuxOS.



ms_meme's Nook: If I Had PCLinuxOS

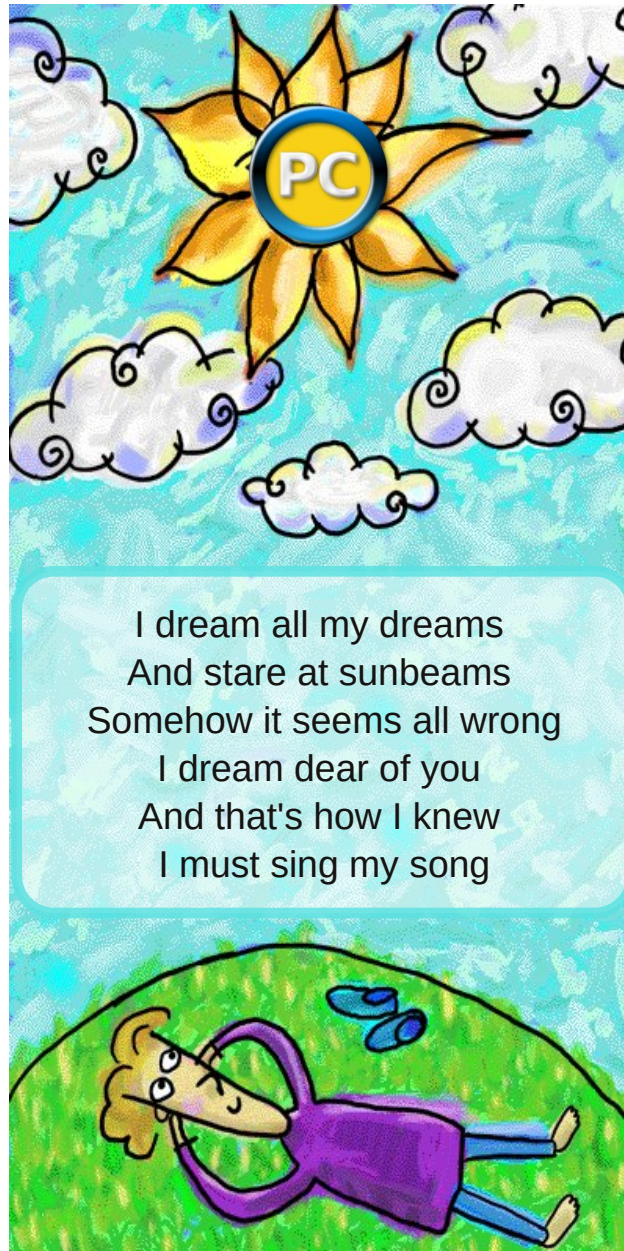
I could surf the web and smile
All of my time would be worthwhile
I could visit every venue
If I had you

I could put Windows behind
Get that OS out of my mind
I could start to boot anew
If I had you

I could have free updating
With you for my guide
I could have no more weeping
You would make me satisfied

I could sing a happy sound
Tell everyone what I had found
There is nothing I couldn't do
If I had you

MP3



I dream all my dreams
And stare at sunbeams
Somehow it seems all wrong
I dream dear of you
And that's how I knew
I must sing my song

I now surf the web and smile
All of my time is worthwhile
I now visit every venue
For I have you

I have put Windows behind
Got that OS out of my mind
I start every boot anew
For I have you

I now have free updating
With you for my guide
I now have no more weeping
For you make me satisfied

I now sing a happy sound
Tell everyone what I have found
There is nothing that I can't do
For I have you

OGG

PCLinuxOS Puzzled Partitions

9			2					
2		5				9	1	7
	6							8
				7	1			
				4			8	
	7	4					5	1
8	9			2	6			
	1						2	
						5	7	

SUDOKU RULES: There is only one valid solution to each Sudoku puzzle. The only way the puzzle can be considered solved correctly is when all 81 boxes contain numbers and the other Sudoku rules have been followed.

When you start a game of Sudoku, some blocks will be prefilled for you. You cannot change these numbers in the course of the game.

Each column must contain all of the numbers 1 through 9 and no two numbers in the same column of a Sudoku puzzle can be the same. Each row must contain all of the numbers 1 through 9 and no two numbers in the same row of a Sudoku puzzle can be the same.

Each block must contain all of the numbers 1 through 9 and no two numbers in the same block of a Sudoku puzzle can be the same.



SCRAPPLER RULES:

1. Follow the rules of Scrabble®. You can view them [here](#). You have seven (7) letter tiles with which to make as long of a word as you possibly can. Words are based on the English language. Non-English language words are NOT allowed.

2. Red letters are scored double points. Green letters are scored triple points.

3. Add up the score of all the letters that you used. Unused letters are not scored. For red or green letters, apply the multiplier when tallying up your score. Next, apply any additional scoring multipliers, such as double or triple word score.

4. An additional 50 points is added for using all seven (7) of your tiles in a set to make your word. You will not necessarily be able to use all seven (7) of the letters in your set to form a "legal" word.

5. In case you are having difficulty seeing the point value on the letter tiles, here is a list of how they are scored:

0 points: 2 blank tiles

1 point: E, A, I, O, N, R, T, L, S, U

2 points: D, G

3 points: B, C, M, P

4 points: F, H, V, W, Y

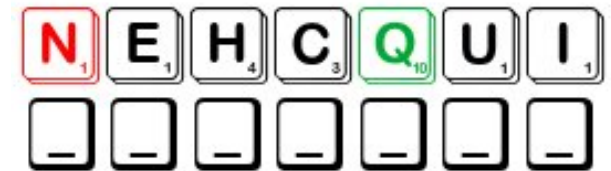
5 points: K

8 points: J, X

10 points: Q, Z

6. Optionally, a time limit of 60 minutes should apply to the game, averaging to 12 minutes per letter tile set.

7. Have fun! It's only a game!



Double Word



Triple Word



Possible score 277, average score 194.

Download Puzzle Solutions Here

PCLinuxOS Word Find: May 2019

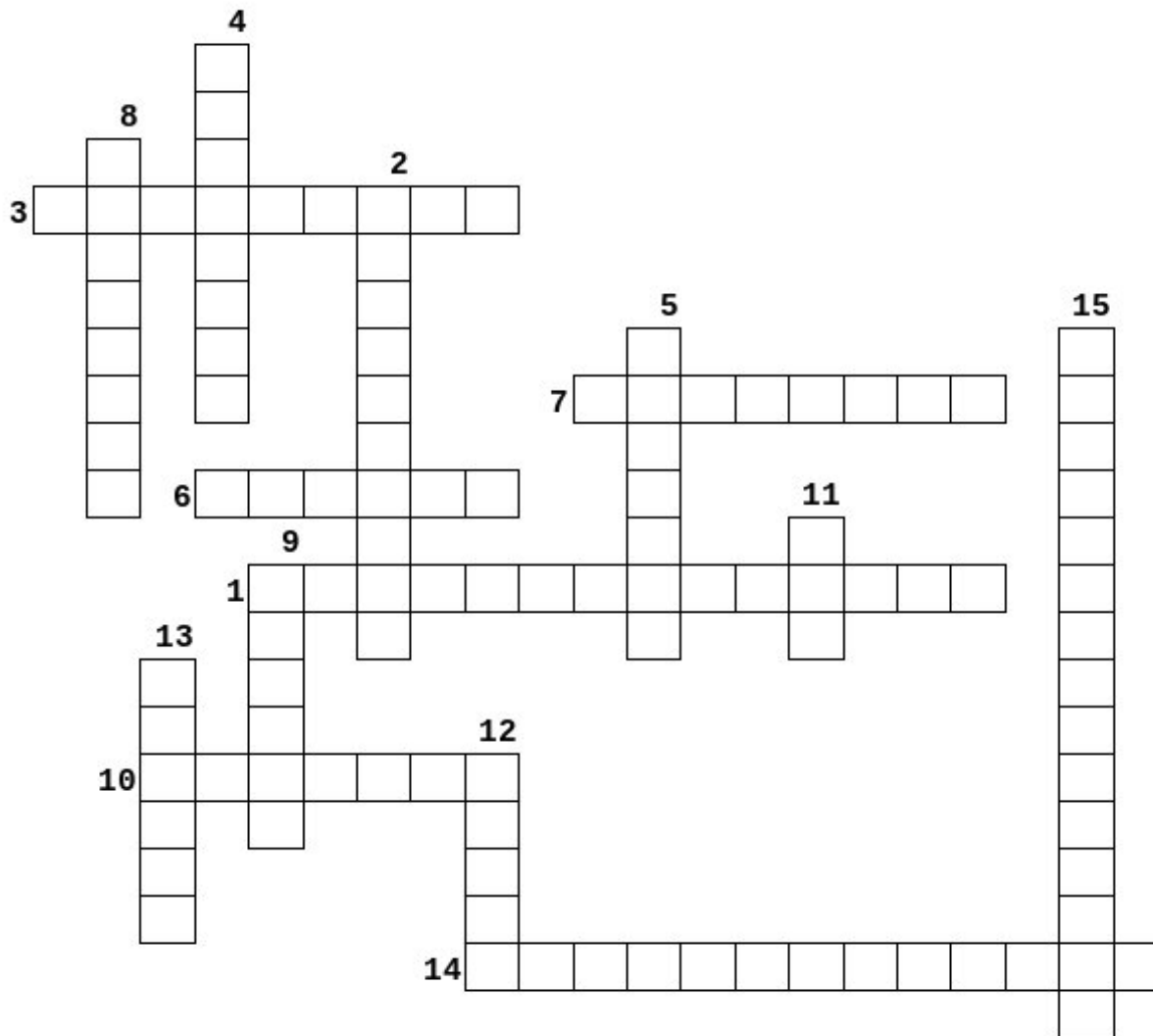
Cooking Tools

M L V O F R E K A M T R U G O Y L M W I N R H A N D V X M Z
 E T Y D E F E D D R B A L U T A P S N I K E M A R W R Q J D
 I L P V L N O W M Z R E N N I P S D A L A S I R Q U J U X V
 O R D H S T L G I K C R A C L E T T E G R I L L J S Q B U T
 N E R D R N J Z X R E T T A L P J R J B D S X S E S C T Q O
 Z D O E I M C R E F B W W N S O L Z Q G L O J T I E L E D X
 N N B X D R G A R X S X I P L O D W B S O F E E T R I E Z L
 B A S T I N G B R U S H E N O C E Y F F M V V O W P K H U X
 O L S K H J I J E H Q G A L W S E X U I I E H R E R K S H W
 J O E M M W Z R E I G P X M C M W L N R A I S S S E Y G C C
 Z C R Z J X O J G C G M C F O A W E T E P S S H P G F N X E
 A I P P O D W A M N C O J G O E W N W P R E O B R R O I R U
 E O C Y S H Q I I R R W L H K R N E W A A K U S E U N K E Q
 C U I T G C N Y Z T G B R Z E C H V R P F B F Y S B D A C I
 L R L R N W R O S H N E N Z R E L O W T Z V F J S M U B I P
 M X R I O F F E Z X S T W A K C C N N N Y R L Q O A E X R C
 O C A L T Y J L P U P P R R S I A O G E T I E M M H P E K L
 R Y G R W M I A F E V Y O Q H Y Z I U M Y K D H A W O W O I
 T T O Q A B U N D T P A N Q S N O T K H T G I M C I T R W L
 A A N B U W I W W R W A E H R E M C Z C G O S Z H A X G R L
 R R A P X A L K M E B A N F W P C E N R G X H R I V O G B Z
 A T S O E B E A L N O G O E X U S V A A N B O G N G N P T T
 N P I T U F N E F E B J T U S T V N P P Q Q N J E Z V F R J
 D A L K A D L S W P C A S T E D S O E Q H I V S P S Z E U O
 P N U R O E L X B O R W A R B I G C H Y E U C Y L C T Y L Y
 E M A L R K N K R N D R Z F I P D Y C G Y R A S K A R C C F
 S C I E T W J E T A D X Z D U Q Z T I F G R G R R N M S K B
 T N U B U E R C O C S K I A C C M C U I C I N G G W V L B B
 L L R K M B B P U S D U P P A R I Q Q U D Q K Y N O K X R Z
 E F C V N D F P O U G S X L C A E C O T C Y Q O O G W V Z L

baking sheet	basting brush
Bundt pan	can opener
carafe	colander
convection oven	corer
crepe pan	custard cup
espresso machine	fondue pot
frying pan	garlic press
grater	griddle
grinder	hamburger press
hand	mixer
honey	pot
ice cream scoop	icing
infuser	mandolin
mortar and pestle	mold
parchment paper	pizza stone
platter	poacher
quiche pan	raclette grill
ramekin	ricer
salad spinner	sieve
slow cooker	souffle dish
tart pan	tea infuser
tongs	trivet
wok	yogurt maker
spatula	zester

[Download Puzzle Solutions Here](#)

Cooking Tools Crossword



1. a cooking device that heats food by the circulation of hot air
2. a thin slab of ceramic on which pizza or bread may be baked
3. a pot used to melt cheese or heat oil to cook meat for eating
4. a cake pan with fluted or grooved sides and a central tube
5. a small dish for baking and serving an individual portion of food
6. a kitchen utensil for removing fine shreds of zest from citrus fruit.
7. a kitchen utensil consisting of a flat frame with adjustable blades, for slicing vegetables
8. a perforated bowl used to strain off liquid from food
9. an open-topped glass flask typically used for serving wine or water.
10. a device for holding tea leaves for steeping or brewing
11. a bowl-shaped frying pan used typically in Chinese cooking.
12. a utensil with small holes through which boiled potatoes or other soft food can be pushed to form particles of a similar size to grains of rice.
13. a small metal structure that keeps a pan above a fire or plate which protects a table from a hot dish
14. a Swiss dish consisting of cheese melted over a fire and then scraped onto bread or boiled potatoes
15. a stone bowl and tool used to grind up things, such as spices or herbs

[Download Puzzle Solutions Here](#)

Mixed-Up-Meme Scrambler



Chef YouCanToo prepared a surprise dish from his Linux

_____.

Use the clues to unmix the letters to make a new word. Remix the letters in the red boxes to solve the puzzle.

Fate

MARAK

Voucher

POUCON

Weapon

TROEKC

Fire

HOSOT

Short

FRIEB

[Download Puzzle Solutions Here](#)

More Screenshot Showcase



Posted by tuxlink, on April 5, 2019, running KDE.



Posted by parnote, on April 6, 2019, running Xfce.



Posted by OnlyHuman, on April 30, 2019, running e17.



Posted by mutse, on April 24, 2019, running Mate.