The PCLinuxOS magazine
Volume 92 September, 2014

International Day Of Peace

Table Of Contents

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
Patrick Horneker
Meemaw
Gary L. Ratliff, Sr.
Daniel Meiß-Wilhelm
AndrzejL
YouCanToo
Mark Szorady
Darrel Johnston
Andrew Huff
Pete Kelly
Antonis Komis
daiashi

Contributors:
daiashi
critter

Welcome From The Chief Editor

Unix was originally created and implemented in 1969 by the Ken Thompson and the late Dennis Ritchie, working for AT&T Bell Labs. They first released Unix in 1970, and it went on to see widespread adoption by academic institutions and businesses. However, even in the 1990s, a commercially available Unix operating system was too expensive for individual users.

MINIX was an early Unix-like operating system intended for academic uses. While the source code for MINIX was free, modification and redistribution was restricted. MINIX featured a microkernel, and had a 16-bit design. However, new processors – namely, the Intel 80386 – were emerging that embraced a 32-bit architecture.

On August 25, 1991, Linus Torvalds send a message on the comp.os.minix Usenet news group announcing his new operating system. Originally named Freax, Torvalds wanted to create a new operating system that leveraged the capabilities of his new computer, which sported an 80386 processor. One of his co-workers at Helsinki University of Technology, Ari Lemmke, renamed the project Linux without consulting Torvalds, which Torvalds had previously considered, but dismissed as being too egotistical. His message read:

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes – it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(. —Linus Torvalds

So was born Linux. While it was built on a computer running MINIX, it used no code from MINIX. Instead of a microkernel, Linux featured a monolithic kernel. Linux leveraged the 32-bit architecture of the 386/486 processors. It was originally built with the gcc compiler (GNU C Compiler), which is still the main choice for compiling Linux programs. Initially released under its own license that restricted commercial activity, Linux was quickly moved to the GNU General Public License. The end result was a fully functional, free operating system.

Over the years, Linux grew and grew – and continues to grow to this very day. Development is ongoing at a feverish rate. Its adoption continues to grow. Linux not only powers the vast majority of servers that operate as the backbone of our beloved Internet, but it also powers everyday items, such as mobile phones, tablet computers, set top DVD players, ebook readers, and is even making inroads into the automotive industry.

Linux is all around us, whether we realize it or not. It has grown from the original 0.01 kernel of 10,239 lines of code, to the 3.16 kernel, which features nearly 17 million lines of code.

Thanks to Linus Torvalds vision, we all benefit today with one of the most – if not the most – secure operating system around. Without Linus Torvalds vision, our options would not be nearly as robust, and we'd be forced into using less capable operating systems.

Happy 23rd Birthday, Linux!

September 21: International Day Of Peace

by Paul Arnote (parnote)

Take a look around at the world we live in today. Anyone who has read recent news headlines, or who has watched the evening news, will be acutely aware of the tumultuous times that we live in. There hardly seems to be a corner of the world that hasn't been shaken by the strife and turmoil that shatters the peace of our modern world.

At a time when we seemingly need it more than ever, on September 21, the world will celebrate International Peace Day. At Noon in every time zone, a moment of silence will be observed, creating a "wave" of awareness for peace across the globe.

Started in 1981 by the United Nations General Assembly, "Peace Day" was formed to give individuals, organizations and nations a way to create practical acts of peace on a shared day. Pathways To Peace, working with the UN, held the first International Peace Day in 1984, in San Francisco. It included a "Minute of Silence Moment of Peace," where cash registers stopped, the televisions went silent, and the global "Peace Wave," which drove home the movement's message of peace to a global audience.

More than a global peace, International Peace Day also focuses on personal peace. It encourages individuals to find peace within themselves, and in their relationships with others.

Pathways To Peace focuses a lot of attention on education. Specifically, they focus on "peace education, which they define as education that specifically promotes respect, empathy, mutual understanding, and conflict management skills. They promote peace education at every grade level, with

the hope of raising an entire generation committed to a culture of peace.

"Let us pledge to teach our children the value of tolerance and mutual respect. Let us invest in the schools and teachers that will build a fair and inclusive world that embraces diversity. Let us fight for peace and defend it with all our might," says UN Secretary-General Ban Ki-moon.

Created by Pathways To Peace just last year (2013), the Peace Education Resource Center was developed in close cooperation with the IDP-NGO Committee at the United Nations, the National Peace Academy USA and Circle of Peace/CCS Montreal. The goal of PERC is to work with partners and NGOs around the world to gather peace education materials and make them available to teachers at every grade level through a single website.

The 7.2 billion residents of this planet must find a way to overcome and resolve their hostilities and differences with one another. We all have nowhere else to go, so we all must learn to share the one home we have. The International Peace Day is a great way to foster peace, not only on a global scale between all nations and all people, but also on an individual level. Working together, and embracing tolerance and understanding, we can find peace in our own lives and relationships. From there, we can only hope that it will spread like a wildfire, until the entire world embraces peace.

We need it now, perhaps more than ever.

PCLinuxOS Recipe Corner

The PCLinuxOS Magazine

Created with Scribus 1.4.4

Campfire Chili In A Dutch Oven

Ingredients:

1 pound piñto beans, soaked in water for at least 4 hours
2 tablespoons vegetable oil
3 pounds pork shoulder cut into 1- to 2-inch chunks.
1 pound raw hot Italian or chorizo sausage, removed from casing
1 large onion, finely chopped
1 jalapeño chili, finely chopped
3 tablespoons chili powder
1 tablespoon ground cumin
2 teaspoons dried oregano
1 cup finely minced cilantro
1 (28-ounce) can crushed tomatoes
Kosher salt and freshly ground black pepper, to taste.
1/2 cup finely sliced scallions

Directions:

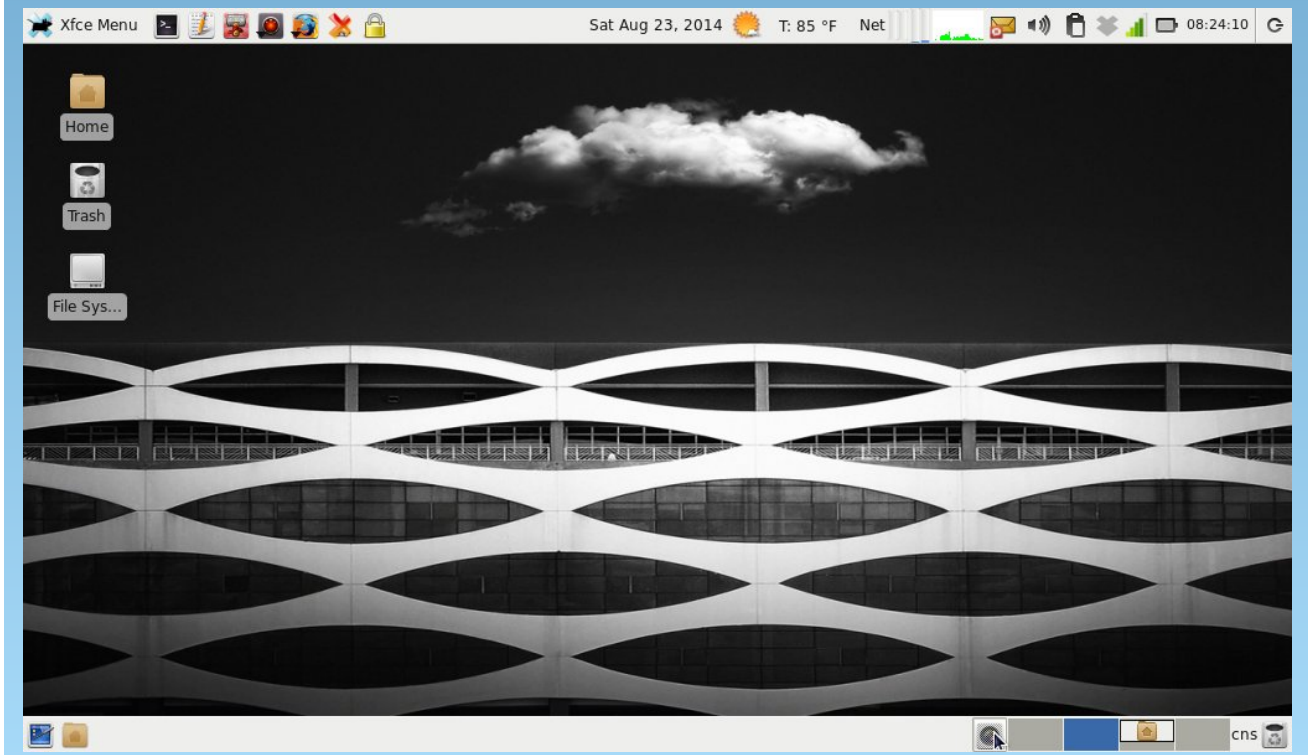
1. Rinse and drain soaked beans. Heat oil in Dutch oven over hot coals until smoking. Add half of pork and cook until well-browned on all sides, about 10 minutes. Transfer to a large bowl and repeat with remaining half of pork. Transfer second batch to bowl with the first batch.

2. Add sausage to pot and cook, breaking it up with a wooden spoon until no longer raw. Return meat to pot with sausage and add onion, jalapeño, chili powder, cumin, oregano, and half of cilantro. Cook, stirring constantly, until aromatic and onions have begun to soften, about 4 minutes.

3. Add soaked beans, tomatoes, enough water to cover meat and beans by 2 inches, and a large pinch of salt (it should still taste under-seasoned, as it will reduce). Place lid on Dutch oven and cover with hot coals. Allow to heat for 10 minutes, then peek and check temperature. Liquid should be mildly bubbling.

4. Allow to cook until beans are soft and creamy, and meat is completely tender, 3 to 6 hours, depending on how hot you cook it (for best results, cook over very low heat for a long period of time). Check on pot as it cooks every hour or so, topping up with water as necessary. After the chili is done, season to taste with salt and pepper and stir in the remaining cilantro and scallions. Serve immediately.

Screenshot Showcase



Posted by parnote, on August 23, 2014, running Xfce.

GIMP Tutorial: Frozen Text

by Meemaw

This tutorial was sent to me a year or so ago, and I thought it was fun. We have done text in GIMP before but this one is a different effect and uses a combination of some skills you already have.

Opening GIMP, create a new 1600 x 1200 canvas with a black background. Click on the text tool, change the foreground color to white, change the text size to 400 px, and type your text. I wrote PCLinuxOS the first time on a bigger canvas, as you see above, but I wrote Linux this time. Choose the Move tool, and move your text to the center of the canvas.

Now, right click the text layer, and choose **Alpha to Selection** to select the text only. Choose **Select > Feather**. In the window that appears, change the number to 10% of your font size, in this case, 40 points.

What we are trying to do is make the text look like ice with the inside clear and the outside frosty. It will be rounded text rather than having sharp corners (center, top).

"Turn off" text layer by clicking on the little eye to the left of it in the Layers window. Create a new transparent layer on top. Using the bucket fill tool, check **Fill Whole Selection** in the tool settings and fill the text with white.

Now use the **Select by Color** tool. Click in the background. Notice that the selection lines are farther out from the letters now.

The background is selected now, so go to **Select > Invert** to select the text. Using the bucket fill again, fill this text with white.

Duplicate this layer, then turn off the original. Select the text layer, and choose **Alpha to Selection**. Then go to **Select > Feather** again. If the feather box is still set to 40 px, click OK.

Select the top layer (mine says "Layer copy") and press Delete. Go to **Select > None** to deselect everything.

If you haven't saved your file, you should do it now.

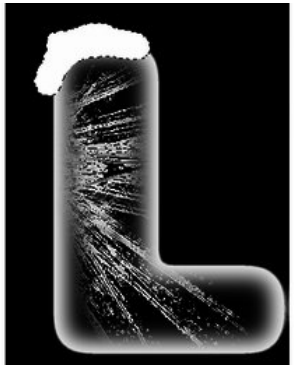
We have the text now, so let's put some ice crystals into it. Create a new transparent layer, check that your foreground color is still white and then select **Filters > Render > Nature > Flame**. In the window

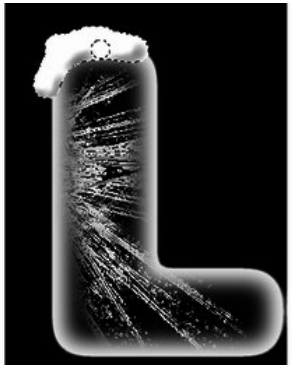
that opens, slide the brightness clear to the right. Click the Edit button at the top, then choose **Bent** from the dropdown in the edit window. Click on the randomize button several times to find a pattern you like. then click OK. (This filter takes a few seconds.) Now click on **Filters > Edge-Detect > Sobel**, then click OK. You'll notice your nice white pattern has changed to black so go to **Colors > Invert**.

We don't want all of that but we are going to clone part of it to make our text look frozen. Find a spot you want to use, then click on the clone tool. Make it the same size as your text with a soft brush. **<CTRL> + click** on the pattern that you want to clone. Turn off that layer and choose the Layer copy. Using smooth strokes, paint your pattern into your letters. It seems to look best if you start in the middle of the letter and stroke up and down without making many short strokes. Make sure you have saved your work (center, top).

Let's add some snow! Add a new, transparent layer on the top. Using the **Lasso** tool, draw an irregular shape on the top of a letter, then bucket fill the shape with white (center, left side).

While your shape is selected, change the foreground color to black, and change to the **Airbrush** tool. Set the opacity of the tool to around 50%, and use a little bigger brush, even bigger than the one you see in the photo below. Put some shading at the bottom of your snow shape (above, right side).

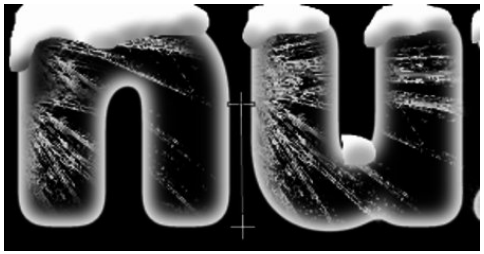




Repeat your snow shapes on each letter. We are to this point now (right, top). Make sure to save your work.

While it looks cool this way, it needs a different background. Click on the black background, most likely the bottom layer. In the toolbox, choose the

Gradient tool. In the tool settings, make sure the **FG to BG** gradient is set, and that your background is set again to white. Change the foreground to a nice blue. The color I used was 485bbc, but you should use whatever looks good to you. (As you saw, the one at the beginning of the article used a darker blue.) Click and drag your mouse straight down from the approximate center of the letter to the bottom.



This illustrates the path of the gradient tool. When you get finished, the text looks like it is sitting on a surface of ice or snow. I added a border to clarify the edges of the drawing, but that's up to you.

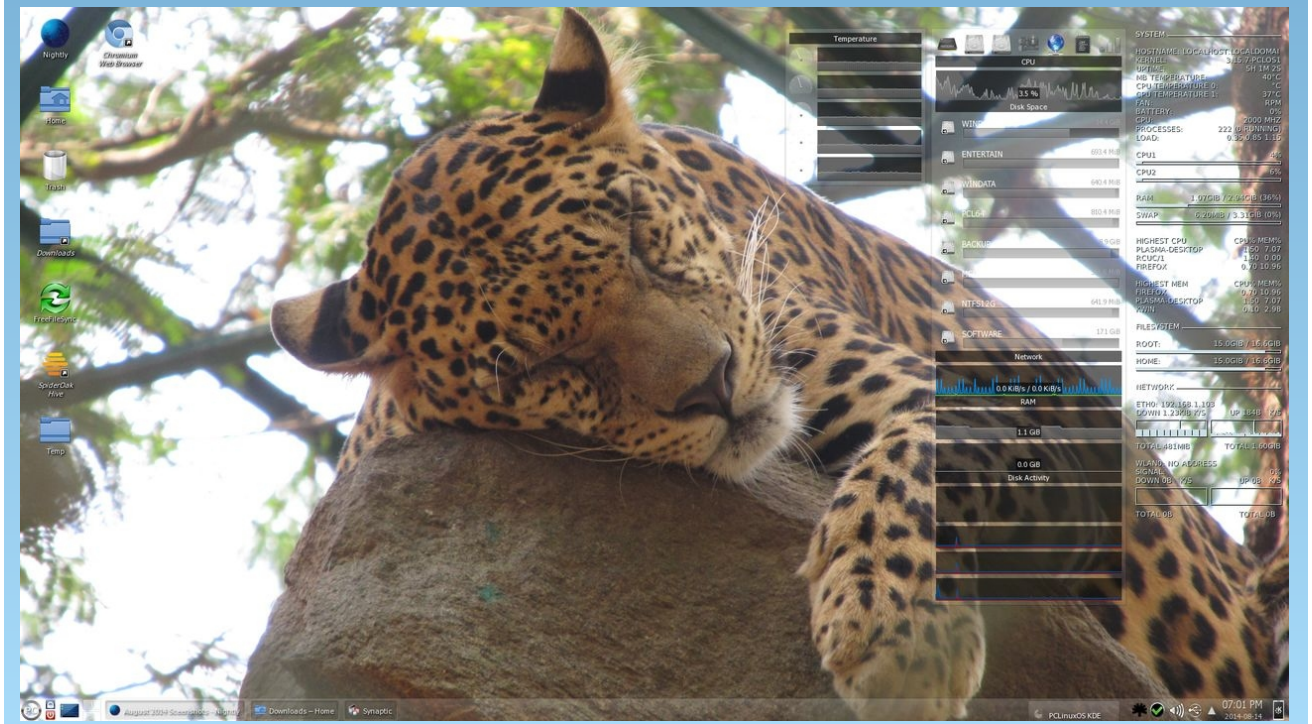
We need snowflakes as well. Add a new layer at the top filled with black. Go to **Filters > Noise > HSV Noise**. In the window that appears, slide the Value slider clear to the right, and click OK. Now, choose **Filters > Light and Shadow > Sparkle**. In that window, change the **Spike Points** slider to 10. Then, in your Layers window, change the **Mode** from **Normal** to **Screen**. Wow!

Save your work and export your creation.

The PCLinuxOS Magazine

Created with Scribus

Screenshot Showcase



Posted by slingshot, on August 14, 2014, running KDE.

Delete-Me-Not Files – Be Gone!

by Paul Arnote (parnote)

I'm sure this has happened to nearly everyone, at one time or another. You find a file stored on your drive(s) that resists all efforts to be deleted. Nothing you do will send this no-longer-wanted file into the digital afterlife. There is little else that is as annoying to computer users than an unwanted file that refuses to die.

Linux is awesome for its ability to read files on all sorts of file systems. EXT4, EXT3, NTFS, BTFS, FAT, FAT32, EXFAT, HPFS, HFS, HFS+, etc. – none are really much of a challenge for Linux. However, that comes at a price, sometimes. All of those different file systems have different rules about how to construct a proper filename, and what characters can and cannot be used in file names. This can cause some huge problems for users who trade files with those who use different file systems … or even for users who make a typo or other mistake when entering a filename in order to save a file.

Illegal Filename Characters

Of course, what's considered an illegal character depends on what file system you are using.

Unsurprisingly, Windows file systems appear to be the most restrictive. Here is a list of illegal characters in Windows file names (in bold): **/ ? < > \ : * | "**. Also illegal is any character you can type with the Ctrl key. Under the FAT file system, the ^ symbol is illegal. FAT file system file names can be up to 255 characters long, while NTFS file system names can be up to 256 characters long. Under FAT and NTFS, the Windows full path may not be more than 260 characters. While the NTFS file system allows a full path of 32,767 characters, with each component of the path limited to 255 characters, many Windows applications will choke on files using such lengthy naming conventions (such as the Windows file manager, Windows Explorer). You also cannot place a space or period at the end of the file names under Windows.

As if all of that isn't enough, Windows also reserves the following file names: **com1, com2, com3, com4, com5, com6, com7, com8, com9, lpt1, lpt2, lpt3, lpt4, lpt5, lpt6, lpt7, lpt8, lpt9, clock$, con, nul,** and **prn**.

At the other extreme, there is only one illegal character for the Mac OS9 and OSX: the colon (:). Under HFS on OS9, filenames may be a maximum of 31 characters. Under HFS+ on OSX, filenames may be a maximum of 255 characters.

Nearly as simple and forgiving are *nix type of file systems. The only illegal characters are /, also known as the forward slash, and a null character. Filenames may be a maximum of 255 characters in length.

Go ahead and give it a try. You can name a file some*file.txt, some?file.txt, some\file.txt, and some:file.txt, but ***not*** some/file.txt. I wouldn't recommend doing this routinely, especially if you trade/send files to users of other file systems – like Windows and Mac users. But, it does illustrate the forgiveness of the *nix type of file systems.

The best defense …

The best way to avoid problems is to adhere to a few simple "rules" when naming your files. First of all, avoid using spaces in filenames and directory names. Sure, the *nix type of file systems can handle them, but you are setting yourself up for plenty of headaches if you are ever forced to use the command line to perform file maintenance duties. You also don't always know which Linux GUI apps are actually acting as a "front" for work that's actually occurring on the command line, behind the scenes. Some GUI applications will handle troublesome files just fine. Others, not so fine. To keep things simple, we'll deal with deleting these stubborn files from the command line.

For example, naming a file "My Yellowstone National Park Vacation 001.jpg" isn't really any more readable than "MyYellowstoneNationalParkVacation001.jpg". Your mind's eye will typically insert the spaces for you. If you find that too hard to read, try using an underscore (_) or dash (-) in place of spaces in your filenames, like this: My-Yellowstone-National-Park-Vacation_001.jpg.

Second of all, adhere to the "rules" of the file system that is most prevalently used. Like it or not, that would be the file system with the ***most*** restrictions, or Windows file system restrictions. By sticking with the rules that apply to Windows file systems, you'll avoid any problems when you trade/swap files with Windows users. Otherwise, you may have to rename the files you share with Windows users before they are able to access them.

Third, avoid using a dash (-) or space as the ***first*** character of your filenames. The reason for avoiding these two characters as the first character will become clearer when we talk about how to get rid of those stubborn files that resist being deleted.

Deleting those files that refuse to be deleted

Like it or not, the best way to eliminate files that otherwise refuse to be deleted is to use the command line. So, before we get started, let's create a couple of files in our ~/Documents folder (expands to home/username/Documents) that will defy "normal" attempts to delete. First, enter **`cd Documents`** at a command line prompt in a terminal session. Next, enter the commands exactly as shown in the command line screenshot below.

```
[parnote-toshiba@localhost Documents]$ touch ./-thatfile.txt
[parnote-toshiba@localhost Documents]$ touch " thisfile.txt"
```

This will create two files. One is named -thatfile.txt, and the other is name " thisfile.txt." Notice that the second file starts with a space, while the first file starts with a dash. Both are problematic.

About the rm command: under PCLinuxOS, the rm command is preconfigured to be interactive (rm -i). This means that, by default, you will be asked if you are sure that you want to delete the file(s) listed. The exception to this is a) if you've defined your own custom set of aliases by creating your own .alias file in your home directory, and haven't copied over the default system-wide aliases to your local .aliases file, or b) you've changed the default system-wide aliases (located at etc/profile.d/60alias.sh).

To illustrate, let's attempt to delete -thatfile.txt. Enter **`rm -thatfile.txt`** on the command line, and press the Enter key.

```
[parnote-toshiba@localhost Documents]$ rm -thatfile.txt
rm: invalid option -- 't'
Try 'rm ./-thatfile.txt' to remove the file '-thatfile.txt'.
Try 'rm --help' for more information.
```

Notice that the rm command returns with an error. The dash that starts the filename makes the rm command think that a command line option is being given – and in this case, an invalid command line option.

There are three ways to work around this problem. I've listed them below.

```
rm -- -thatfile.txt
rm ./-thatfile.txt
rm "-thatfile.txt"
```

The first option uses a double dash immediately following the rm command. This tells the command that what follows isn't an option. The second option uses a relative path statement (./), which tells rm to remove the file -thatfile.txt, which exists in the current directory. The third option encloses the filename within double quotes. All three methods will successfully delete the file.

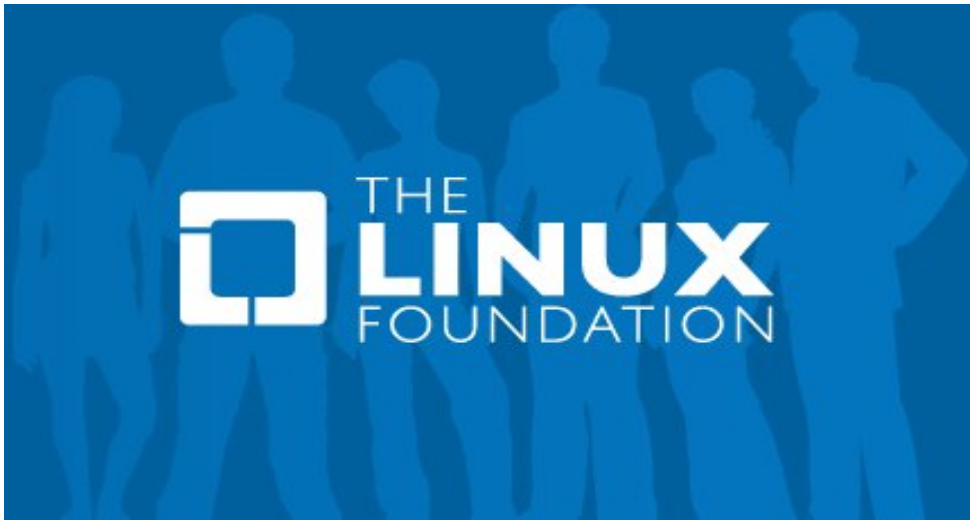
Now, let's try to delete the second file. Enter **`rm thisfile.txt`** on the command line in a terminal session.

```
[parnote-toshiba@localhost Documents]$ rm thisfile.txt
rm: cannot remove 'thisfile.txt': No such file or directory
```

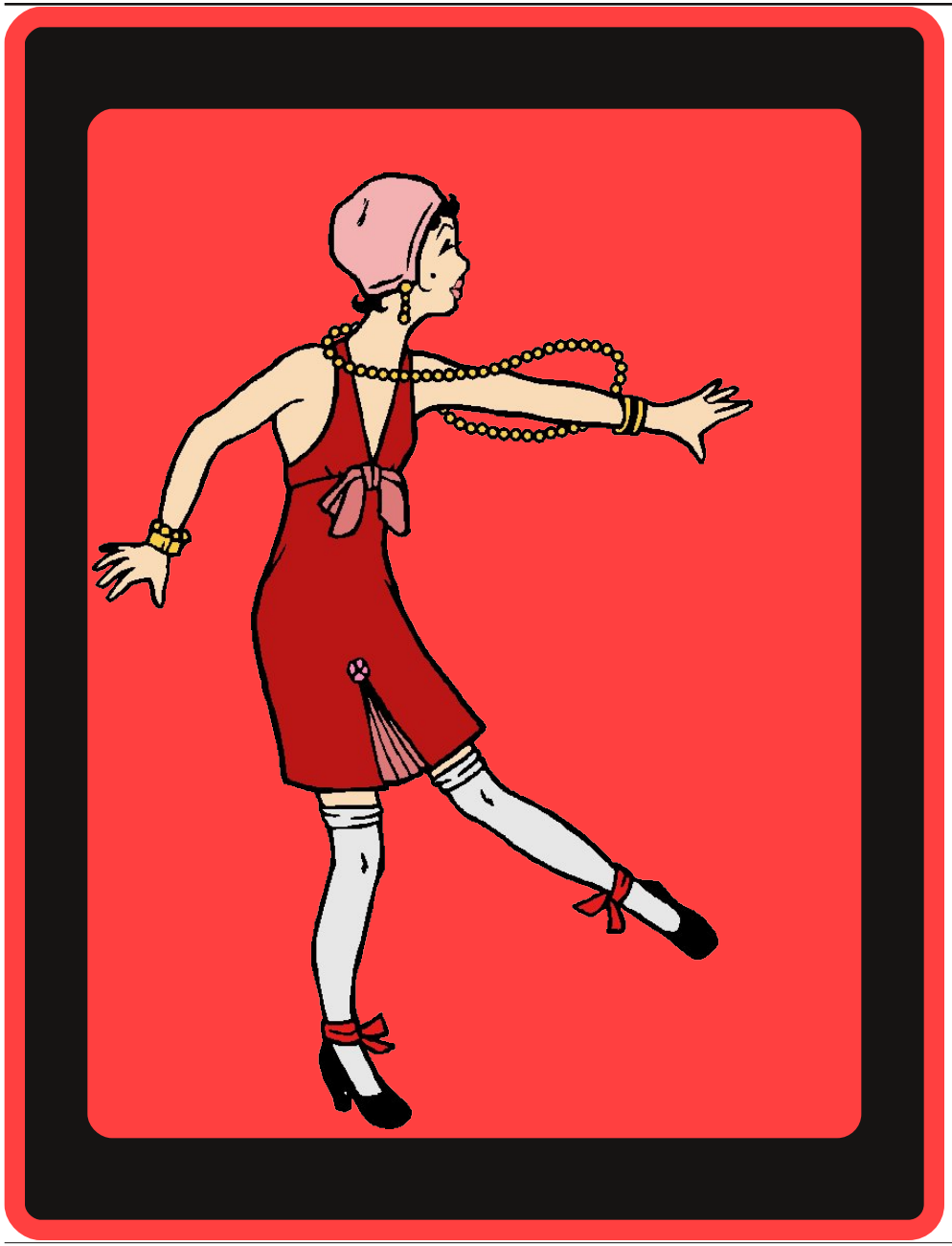
Notice how the rm command cannot find the file. Quite simply, the file named thisfile.txt does not exist. Instead, you can delete this file by using **`rm " thisfile.txt"`**. This encloses the filename in double quotes, with the space at the beginning. You can also use the backslash (\) character to escape the next character (a space), like this: **`rm \ thisfile.txt`**.

Summary

Your best bet to avoid problematic files is two-fold. First eliminate spaces in your filenames. Second, honor the restrictions imposed by Windows with regard to illegal filename characters. By doing both, you'll go a long way towards avoiding any issues with files that refuse to be deleted.



ms_meme's Nook: PCLOS Grows And Grows



In olden days when a virus hit cha
You'd cry a bit and let it git cha
Now heaven knows use Linux no woes

Windows and Apple used to be the rage
But they can't compete in the Linux Age
At them thumb your nose those so and sos

The world is mad today lots of fad today
So rad today but I'm glad today
'Cause everyone knows
PCLOS is on its toes

If you're lookin' for an OS that
Won't give you any stress
Use what I chose
PCLOS grows and grows

MP3 OGG

International Community PCLinuxOS Sites

PCLinuxOS.nl
Netherlands

PCLinuxOS
Daykonun mekanı
Turkey

PC Linux OS
Denmark

Czechoslovakia
PCLinuxOS
síla jednoduchosti

PCLinuxOS.it
Italy

PCLinuxOS Polskie Centrum
Poland

PCLinuxOS FINLAND

PCLinuxOS
Brazil

Please Donate To PCLinuxOS.

LINUX FORUM

Screenshot Showcase

Posted by Nymira, on August 3, 2014, running KDE.

Forum Down? What Will You Do?!

by Paul Arnote (parnote)

On August 25, 2014, the PCLinuxOS forum went down for seven or eight hours. That's right. On the 23rd birthday of Linux, the PCLinuxOS forum went AWOL. While the reported cause was a power outage, early speculation included everything, including a DDoS attack.

Oh my! What do I do with my time?

Fortunately, I've come up a dozen ideas on things you can do while waiting for a forum outage to be rectified. Unfortunately, most of them aren't as fun or fulfilling as visiting the PCLinuxOS forum, but they will help fill the void left by forum withdrawl.

1. Reacquaint yourself with your spouse or significant other.

2. Talk to or visit with your kids.

3. Continue your PCLinuxOS family fraternization with your fellow PCLinuxOS users on the PCLinuxOS IRC channel. This channel is purely for chit-chat. Direct all of your support type questions and discussions to the PCLinuxOS-Support IRC channel. Feel free to drop by the PCLinuxOS Magazine IRC channel, too, while you're on IRC. We'll talk about anything and everything. There's not too much that's considered "off limits." If you hit those limits, we'll give you a kind warning before giving you the boot.

4. Start scratching things off of your "honey do" list. I don't know about you, but my list has become a book. Of course, you'll stay out of the proverbial doghouse if you actually ***do*** the things on your list – not just scratch them off.

5. Find – and read – a good book. You do remember what a book is, don't you?

6. Teach yourself something new. Learn how to use commands on the command line. Learn how to write bash scripts. Learn some new GIMP or Inkscape skills. Follow one of Meemaw's fine graphics tutorials.

7. Do something kind and unexpected. Do the dishes when it's not your turn to do them. Clean the bathroom. Run the vacuum cleaner. Make dinner (but not right after cleaning the bathroom). If you're not the culinary type, take a friend or your significant other out for dinner.

8. Organize your graphics files, or your music files. If you're like me, they do get jumbled … despite the best of intentions to keep them organized.

9. Catch up on a television series that you've always meant to watch, but could never find the time to watch.

10. Search out and watch videos of interest to you on YouTube. With an average of 100 hours of videos being uploaded to YouTube every minute, of every day, you'll be hard pressed to NOT find something to pique your interest on there.

11. Go to The PCLinuxOS Magazine website, and download all of the PDF versions of the magazine. We have every issue ever produced available as a FREE download. So, make sure your collection is complete.

12. Monitor Texstar's Twitter feed to get the latest info on when the forum will be back up.

Undeniably, spending time in the PCLinuxOS forum has become a significant part of many PCLinuxOS users' day. BUT … there are other things to do. I'm as guilty as anyone else. At the time I wrote this article, I have spent over 112 days in the PCLinuxOS forum during my tenure as a PCLinuxOS user.

Whatever it is you decide to fill that void of time with, just be sure to come back to the forum when it comes back online.

See y'all in the forum!

PCLinuxOS Family Member Spotlight: jimwilk

by jimwilk, as told to Smileeb

How old are you?
74 years old.

Married, single or what?
Married to my life's companion for 48 years, and we are still together.

Children, grandchildren?
We have two children (now both in their 40's) who are each married and have provided us with four lovely grandchildren. Their ages range from 12 to 9. We refer to them as the "Awesome Foursome."

Retired or working and for how long and at what?
Although I studied geology at Victoria University in Wellington, New Zealand, I had a fair bit of mathematics, physics and chemistry in my degree. Thus, in 1966, I switched from working as a hydrologist to teaching high school mathematics in various locations around New Zealand. I retired at the end of 2004, but continued as a relief teacher (supply teacher) until the end of 2011.

What is the area you live in like? Weather, Quietness, Scenery?
My wife and I live in the city of Palmerston North, which is situated about two hours by road north of the country's capital city Wellington, in the southern North Island. The area is on the banks of the Manawatu River, which flows between two mountain ranges through the Manawatu Gorge. The climate is temperate, seldom reaching temperatures above 30 C or below 7 C. The Manawatu area is quite windy, with fairly frequent northwest winds of up to 90 km/hour.

Are you handy with your hands and have any hobbies?
I enjoy spending time in my workshop, keeping a reasonable vegetable garden going, and also working as a volunteer driver on the Palmerston North Esplanade miniature railway.

What is your education level?
BSc with Honours in Geology

Do you like to travel, go camping?
We don't now travel as much as we used to. Our grandchildren are all close by, and we don't need to drive for more than an hour to go visit them. We have never been interested in camping.

What caused you to try Linux and join this forum?
After I retired from teaching, I realised that I was no longer entitled to retain the school supplied copy of Windows XP Pro. So I decided to learn to use Linux as a retirement project. After using Mandrake and then Xandros, I was introduced to PCLinuxOS by a fellow Kiwi. I started with version 0.93 and, apart from short term dabbling with Mepis and Klikit, have been with PCLinuxOS ever since.

PCLinuxOS Family Member Spotlight is an exclusive, monthly column by smileeb, featuring PCLinuxOS forum members. 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 smileeb in the PCLinuxOS forum expressing your interest.

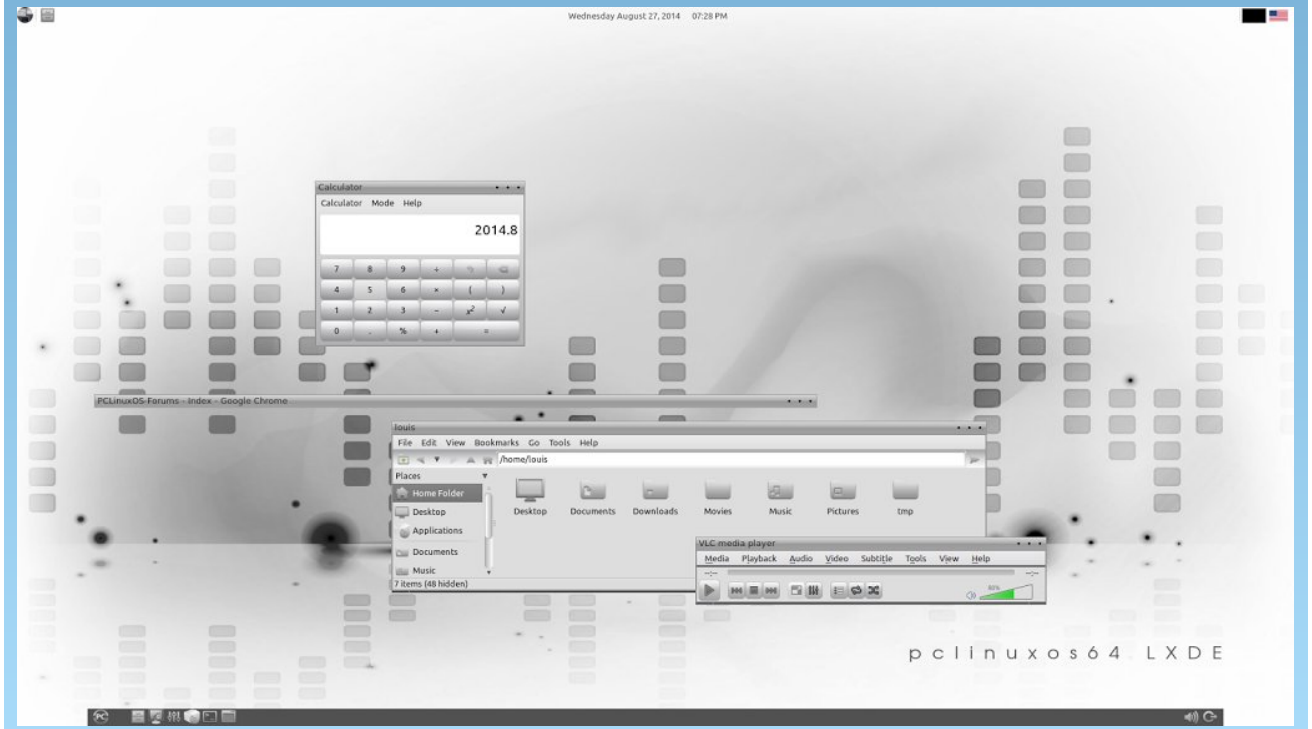
Want To Help?

Would you like to help with the PCLinuxOS Magazine? Opportunities abound. So get involved!

You can write articles, help edit articles, serve as a "technical advisor" to insure articles are correct, create artwork, or help with the magazine's layout.

Join us on our **Google Group mailing list.**

Screenshot Showcase



Posted by LKJ, on August 27, 2014, running LXDE.

Boot Up With Grub2

by Peter Kelly (critter)

Introduction

When you turn on your computer to start up an operating system, the process is handled by a program known as a boot loader. The PCLinuxOS distribution of Linux uses a boot loader known as Grub (Grand Unified Bootloader) from the GNU Project. This has worked just fine for many years and has proved itself to be both a reliable and effective method of starting up the system.

The default version of this program used by PCLinuxOS is version 0.97, which is no longer actively maintained and has had to have several patches applied to enable it to keep up with recent developments in hardware, software and firmware. To overcome these obstacles and any future restrictions, a new system has been written from the ground up which is quite a lot different in use from the original. To distinguish between the two versions, it is common to refer to the original as **grub-legacy** and the new version as **grub2**.

Many distributions have now switched over to Grub2 and, despite the initial uptake being rather slow and cautious, it now dominates the Linux distribution scene. Although grub-legacy still performs its function for PCLinuxOS, its days are numbered, at least as a preferred application.

With grub2 reaching full version 2.00 status (some distributions adopted it as early as V.1.97 I think, which is considered by many to denote a sort of V.2 'beta' release), Texstar (PCLinuxOS' lead developer) has introduced it to the repositories as an option. The inclusion of Grub2 in the repository occurs perhaps with just a little reticence, judging by a comment he made on twitter. Meanwhile, the plans are to keep grub-legacy as the distribution default.

I liked grub-legacy and understood its operation quite well, which enabled me to always get to a bootable system no matter how much I abused my system with partition changes and installations of other 'distros'. There are many PCLinuxOS users who feel strongly about the transition and at first glance, Grub2 does seem to be alien and unfriendly. I believe that moving to Grub2 is inevitable, and therefore, I decided that rather than take a Luddite approach, I would meet the beast head on. The outcome? I like it, I really do. It is not unfriendly, although it is capable of some rather advanced tricks if you want to use them. It is very, very customisable – much more so than its predecessor. If you like eye candy, you'll love Grub2.

The recommended method of booting PCLinuxOS is still grub-legacy, so if you decide to try Grub2, then you may get little support from the forum if something goes wrong. You should not, therefore, try this on a production machine. I used a virtual-box installation for my trials, but have now switched over to Grub2 on all of my machines, after a full system backup.

Installation

Installation is, like anything released by Texstar to the repositories, simple and functional. You should also install the grub2-theme-pclinuxos package to get a flavour of the theming capabilities in Grub2. There are two versions of Grub2 in the repositories, grub2 and grub2-efi. As I don't have any systems that need efi support, I can't comment on the second one, but I would expect that if it made it to the repository, then it will work just fine.

Grub2 should preferably be installed to the Master Boot Record (MBR) of the boot drive. Although it is possible to install it to a partition, this is discouraged. I mean here the core boot image, not the grub configuration files, which will normally be stored in the /boot directory. With the packages installed from the PCLinuxOS repositories, the MBR will still contain grub-legacy code and this must be over-written with grub2 code by using, as root, the command

```
grub2-install /dev/sda
```

Replace /dev/sda with the destination of the drive that you use to boot from. This will hopefully terminate with the message

```
Installation finished. No error reported.
```

Follow this with the command

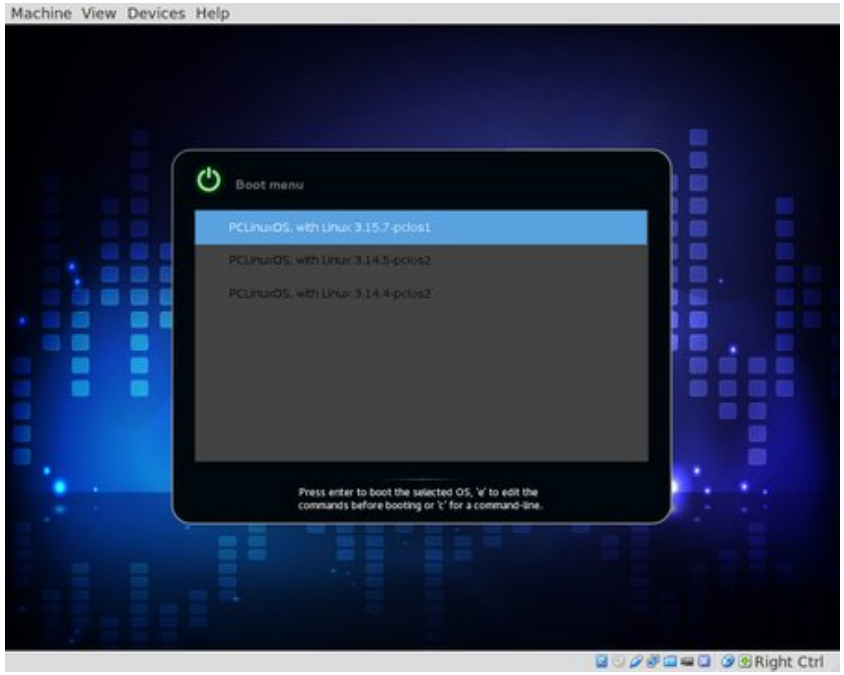
```
update-grub2
```

Which will generate output similar to this:

```
Generating grub.cfg...
```

```
Found theme: /boot/grub2/themes/pclinuxos/theme.txt
Found linux image: /boot/vmlinuz-3.15.7-pclos1
Found initrd image: /boot/initrd-3.15.7-pclos1.img
Found linux image: /boot/vmlinuz-3.14.5-pclos2
Found initrd image: /boot/initrd-3.14.5-pclos2.img
Found linux image: /boot/vmlinuz-3.14.4-pclos2
Found initrd image: /boot/initrd-3.14.4-pclos2.img
done
```

The actual output will depend upon your installation, but here grub has found the theme files, three kernels, and their respective initrd.img files, so things are looking good. A reboot shows this.



Here we can see the nice theme that Tex has provided, and we can select to boot from any of the three kernels displayed.

Now, that wasn't too unfriendly, was it?

The graphic appearance of a screen is a rather subjective thing, and the one above may not be to your taste. Fortunately with Grub2, this is easy to change. I will show you several ways to accomplish this using a simple text editor and your favourite graphics utility.You don't even need any artistic ability, just common sense.

To get started I'll adjust a few colors. I find the lower two options difficult to read with my poor old eyes, so I am going to apply a few tweaks. The file that I am going to edit is

```
/boot/grub2/themes/pclinuxos/theme.txt
```

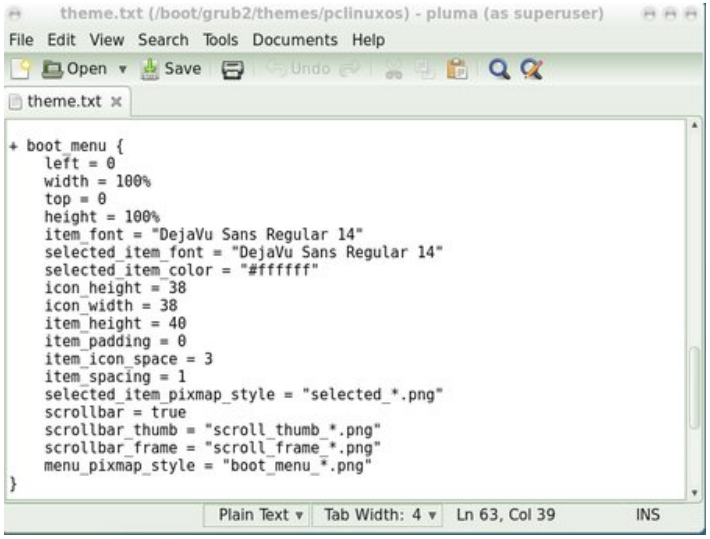
This file is owned by root, and as an ordinary user I am not allowed to edit it. I must do this as root. Be sure to first make a backup of the file. This is only the theme, nothing that will hurt the system, and in the worst case, you could simply re-install the theme package from the PCLinuxOS repository.

> I tend to use the command line a lot but if you are more comfortable using graphical applications, then most of the editing that follows can be done from your file manager by right clicking the file, then choosing the option 'open as administrator.' If you are unsure about doing this, then you should seek assistance as we are soon going to be working with some system files that control how, and more importantly if, your system boots up.
>
> Even if you are confident editing system files, practicing in a VirtualBox environment is a good idea when you first start out.

As root:

```
cd /boot/grub2/themes/pclinuxos
cp theme.txt theme.txt.bak
```

This puts you in the theme directory where the only files are theme files, not system critical files (pclinuxos is the name of this theme), and then makes a backup copy of the Grub configuration file. Next, I need to open the file in a text editor. Use whatever you are comfortable with: vim, nano or a graphical one such as Kate. As I am using the mate DE, I shall use the Pluma text editor.



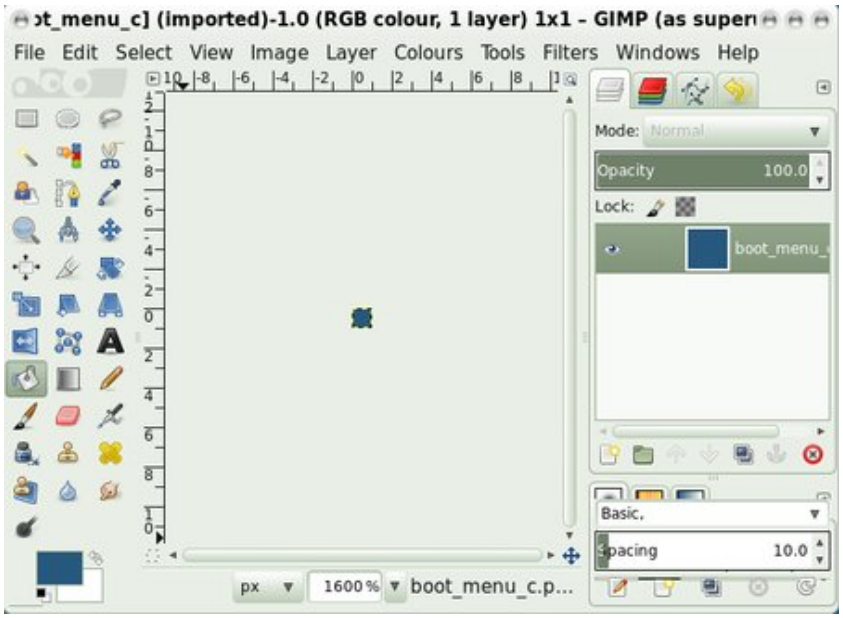
Scroll down to the section named boot_menu, and change the font name

in selected_item_font from DejaVu Sans **Regular** 14 to DejaVu Sans **Bold** 14. Next, change selected_item_color from #ffffff to #00008a, and then add a new line item_color #ffffff.

Save the modified file and close the editor.

To change the background color of the boot menu, we need to use a graphics editor to modify the file boot_menu_c.png. This file is used to fill the working area of the boot menu. I use GIMP for this, but again, I would make a backup of the original file.

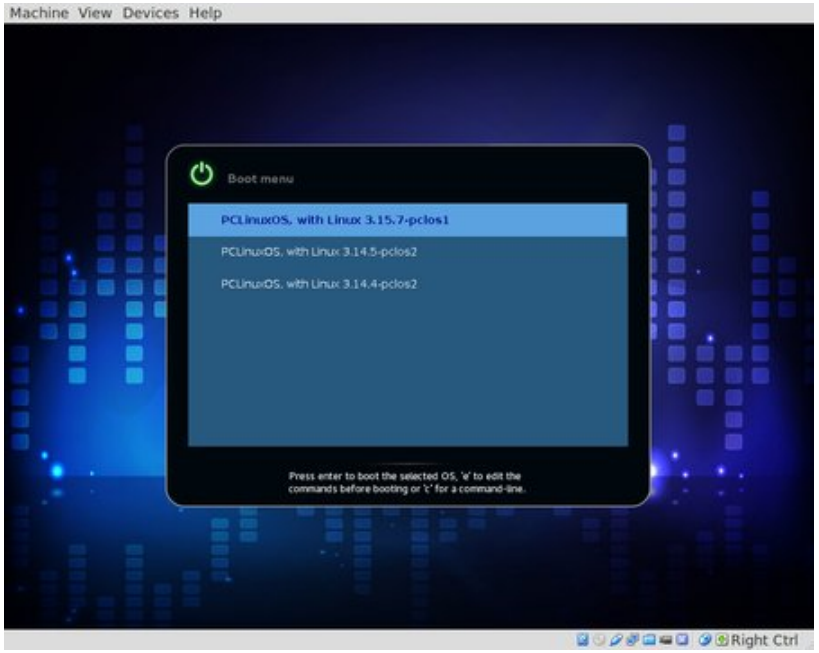
```
cp boot_menu_c.png boot_menu_c.png.bak
gimp boot_menu_c.png
```



The image is only one pixel square, so you will have to zoom in to see it. That's the 5 key in Gimp. Using the Bucket Fill tool, change the color of the image. Overwrite the original .png file (that's why I made the copy). Reboot to see the effect (right, top).

That is how easy it is to change some of the elements of a theme: however, if you want to do more, then read on.

You do not need to use a theme to decorate the Grub2 menu screen, although there are many themes available for download that you may use or modify as



shown. Grub2 allows you to apply background images and cosmetic changes at a very basic level. Themes are much more powerful, though.

To display a background image in Grub2 without using a theme is quite simple. There are a couple of ways to accomplish it, but the preferred way is simple enough, so I will stick to that. Edit the following file (as root)

```
/etc/default/grub
```

Locate or add a line that sets a grub parameter to the full pathname of your file like this:

```
GRUB_BACKGROUND=/boot/grub2/my_background.png
```

Files need to be .png, .jpg or .tga format. If the /etc/default/grub file contains a line pointing to a theme file, then comment it out like this:

```
# GRUB_THEME=/boot/grub2/themes/pclinuxos/theme.txt
```

Save the file, and then run the following command as root

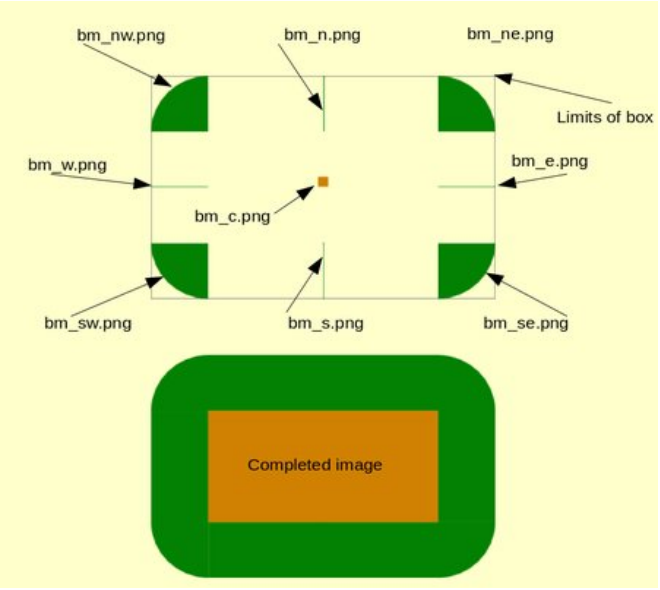
```
update-grub2
```

This command must be executed **every** time you make some changes so that the configuration files can be re-generated.

Other features, such as fonts, can be changed in a similar manner by editing the `/etc/default/grub` file. That's what it is for.

If you prefer to go along the theme route, then the documentation in the manual is quite good. However, I find it a lot easier to take an existing theme and modify it, retaining any references to the previous developer(s) of the original.

A theme comprises two parts, the text or script that describes the positioning and properties of the graphical elements and the graphics themselves. Most of the usual graphical elements are supported including text labels, images and progress bars, but Grub2 makes use of so-called 'styled boxes' to display some graphic components. A styled box consists of 9 divisions or slices, a central slice and 8 peripheral slices sited at, and named after, the compass points: N, S,W,E,NW, NE, SW, and SE. Styled boxes are used to display the boot menu, the horizontal progress indicator, the slider and the slider thumb, although not all slices need to be used to display an element. The four corner slices are not scaled, but the cardinal slices, (N, S, E, and W) which will normally require only a dimension of one pixel in the scaling direction, will be scaled or stretched to fill the gap between the corners. The central slice is a fill slice that will occupy all of the space inside the other slices, and is typically one pixel square. The slider thumb is an example of an exception to this general rule comprising of only the north, south and central slices, with the latter being the full width of the other slices. See the diagram below of a styled box for a boot menu named 'bm.'

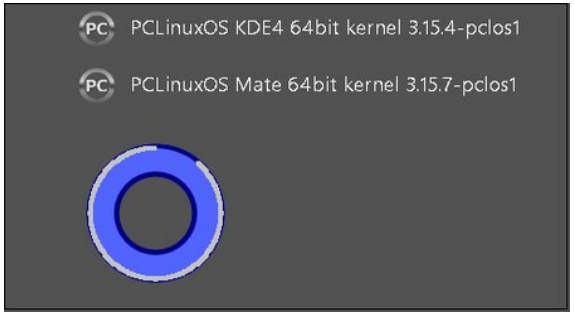


If you would prefer to display a circular progress indicator, then you will need to supply two graphics, a center graphic and a tick graphic. The center image will be displayed centrally within the bounds of its left, top, width and height properties but will be not be scaled to fit that space. Therefore, the values should be selected to wholly contain the center graphic. The tick graphic will be used to update the progress along the circumference of the indicator, and this will be neither scaled or rotated, and so should be drawn so that the lack of rotation is not detrimental. The circular progress element has the following properties that should be set in its definition:

```
left
top
width
height
id = "__timeout__"  This parameter ties the indicator to
                              the GRUB_TIMEOUT value.
center_bitmap
tick_bitmap
num_ticks
start_angle
ticks_disappear
```

The last three need a little explanation. num_ticks is the total number of ticks that will be drawn to make a full circle, and start_angle is the angle to start drawing, measured from the positive x-axis *clockwise*. Seems simple, eh! Hold on there, not so fast. The angle is measured in *parrots*. At least that's what they are called on one website I visited, and while the definition given there seems accurate, this is a unit I have never before heard of. True or false, I like the idea, so I shall call them parrots. A parrot is apparently 1/256 of a full circle, so that to start at top-dead-center (12 o'clock), you would need to supply a value of -64. Knowing this, life becomes simple again.

Normally, the ticks are progressively drawn to complete the circle. Setting ticks_disappear to true starts with a full set of ticks, and it slowly wipes them out until none remain.



```
+ circular_progress {
      id = "__timeout__"
      left = 112
      top = 466
      width = 120
      height = 120
      num_ticks = 360
      start_angle = -64
      ticks_disappear = false
      center_bitmap = "cp_center.png"
      tick_bitmap = "cp_tick.png"
}
```

I have found Inkscape to be extremely useful for creating theme graphics, as you have complete control over pixel size and position. The four corners of a styled box can be created by using a combination of rotation and reflection, and the side slices made by selecting a single row of pixels from each of the corner slices.

The naming of the individual slices is important, and they are called from the script with a single line, such as:

```
menu_pixmap_style="bm_*.png"
```

Items to appear on the boot screen are defined by type, properties and content. For example, an image to be displayed might be defined as:

```
+ image {
      top = 100
      left = 100
      width = 350
      height = 80
      file = "pclos_logo.png"
}
```

These definitions are collected together in horizontal, vertical or rectangular 'canvas' containers in the script. This controls the layout of the boot screen. In the PCLinuxOS theme the rectangular container 'canvas' is used to hold all of the boot menu items which are positioned relative to the bounds of the container. Every item defined between the first brace of the canvas definition and the closing brace of that definition will be contained within the boundaries of the canvas. Items outside of that definition may appear elsewhere on the screen.

To create my new theme, I made a copy of the PCLinuxOS theme, gave it the catchy title of pclinuxos2, and then changed the entry in */etc/default/grub* to point to that theme.

I prefer a lighter, uncomplicated theme that allows me to concentrate on the business at hand, which is simply to choose an operating system to boot into. Geometry, particularly circles, can become distorted by different screen aspect ratios, so I looked for either an abstract design or something fairly linear. I finally settled on a beach scene. After an hour or so in GIMP, I had mangled the original graphics to fit my design (which I previously drafted in Inkscape to get the positional data).

In order to get the menu items to display an associated icon, one line in each stanza must be edited to contain a statement such as

```
--class os-name
```

where os-name is the name of the operating system. This should also be the first --class statement on the line. There must also exist in /boot/grub2/themes/theme-name/icons an icon with the corresponding 'os-name.png' type name, such as pclinuxos.png or, if you really must, windows.png.

That just left me the theme.txt script to re-arrange. The result is shown below.

What! No MS Windows?

Want to add some graphics to it? No problem, simply add a definition or two at the end of the theme.txt script, or in the appropriate place. The images should be in the theme directory, or the full path given.

```
+ image {top = 100%-125
      left = 25
      width = 100
      height = 100
      file="tux-surf.png"}
+ image {top = 100%-130
      left = 125
      width = 84
      height = 100
      file="tuxess-lua.png"}
```

Don't forget to execute

```
update-grub2
```

Then reboot.

This was my re-worked theme.txt script. Note that the image bug.png has been replaced by a block of boot menu background color to repaint the missing progress bar after a visit to the command line. The rest you should be able to figure out for yourself.

```
# Theme for GRUB2
# Copyright © 2012 Ulrich Hansen
# Modified for PCLinuxOS magazine august 2014
# License: GNU GPL v2 or (at your option) any later
version.

#general settings
message-font: "DejaVu Sans Bold 14"
title-text: ""
message-color: "#ffffff"
message-bg-color: "#000"
desktop-image: "background.png"
desktop-color: "#495585 "
terminal-box: "terminal_box_*.png"
terminal-font: "DejaVu Sans Mono Regular 14"

+ progress_bar {
  id = "__timeout__"
  top = 80%-66
  left = 50%-175
  height = 30
  width = 350
  bar_style = "progress_bar_*.png"
  highlight_style = "progress_highlight_*.png"
}
# start of the canvas container
# this controls the positions of title.png, two instances
of bar.png
# and the boot menu
+ canvas {
      left = 20%
      width = 60%
      top = 20%
      height = 60%
+ image {
      top = 18
      left = 18
      width = 140
      height = 40
      file="title.png"
}
+ image {
      top = 70
      left = 50%-120
      width = 240
      height = 2
```

```
      file="bar.png"
}
+ image {
      top = 100%-86
      left = 50%-120
      width = 240
      height = 2
      file="bar.png"
}
+ boot_menu {
      left = 0
      width = 100%
      top = 0
      height = 100%
      item_font = "DejaVu Sans Regular 14"
      selected_item_font = "DejaVu Sans Bold 14"
      selected_item_color = "#ffffff"
      item_color = "#e09f5a"
      icon_height = 38
      icon_width = 38
      item_height = 40
      item_padding = 60
      item_icon_space = 30
      item_spacing = 1
      scrollbar = true
      scrollbar_thumb = "scroll_thumb_*.png"
      scrollbar_frame = "scroll_frame_*.png"
      menu_pixmap_style = "bm_*.png"
}
} # end of canvas container

# Operating instructions
+ image {
      top = 100%-66
      left = 50%-175
      width = 350
      height = 30
      file="help.png"
}

# This is added as workaround for a Grub2 bug,
# that leaves the space of the progress bar unrendered
# if we return from the Grub2 terminal.
#

+ image {
      top = 100%-66
      left = 50%-175
      width = 350
      height = 30
      file="bug.png"
}
+ image {
      top = 100%-125
      left = 25
      width = 100
      height = 100
      file="tux-surf.png"
}
+ image {
      top = 100%-130
      left = 125
      width = 84
      height = 100
      file="tuxess-lua.png"
}
```

Operation

This text describes grub version 2.00 under PCLinuxOS as displayed by the command

grub2-install -v

There may be small differences in other versions or other distributions implementations.

Note: from the manual we are advised - "*Currently autogenerating config files for multi-boot environments depends on os-prober and has several shortcomings. Fixing it is scheduled for the next release.*"

This may obsolete some of the following methods, but the principles of booting a system remain the same. Keeping a backup copy your grub.cfg file, and using the information that follows, should enable you to recover from most situations.

If ever we have a problem with booting, we need to know the basics of how Grub2 operates and is configured, as this differs greatly from the old grub-legacy

where you, or the system administrator, wrote the commands and boot stanzas in a text file saved as

```
/boot/grub/menu.lst.
```

The configuration files are now found in three places:

• The file */etc/default/grub* is used to set certain grub2 parameters which control the default operation of Grub2. This also the place to change your theme if you have more than one.

• The directory */boot/grub2* holds files and directories for use by Grub2 during the boot process. The file grub.cfg should not be edited by users although it is safe to do so. This file is automatically generated by the update-grub2 command, which runs a series of executable scripts stored in the directory /etc/grub.d. Therefore, any changes that you make by editing this file will be overwritten the next time the system is updated by the update-grub2 command. The file is quite complex

and difficult to follow, even for experienced script hackers, so is best left alone. Changes should instead be made by editing the scripts in /etc/grub.d. This is not as difficult as it sounds, and there are a few cheats that you can use to make life easier.

• The scripts used to generate the */boot/grub2/grub.cfg* configuration file are stored in the directory */etc/grub.d*. Each of the executable scripts are run in turn (sorted in shell expansion order so 20_linux_xen is executed before 20_ppc_terminfo), and their output is put into the file */boot/grub.cfg*. At the time of writing, this directory contains the following files:

```
00_header
10_linux
20_linux_xen
20_ppc_terminfo
30_os-prober
40_custom
41_custom
90_persistent
93_memtest
README
```

When you are used to the */etc/boot/grub/menu.lst* configuration file of grub-legacy and you first see a grub.cfg for Grub2, you will be appalled. It is horrible. Only masochists would want to write this stuff. Well, good for them. Happily, we don't have to since the update-grub2 utility will write them for us. We can make use of some of the stuff in here though, even if we don't understand how it works.

The /etc/default/grub file as installed by PCLinuxOS looks like this.

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash=silent"
GRUB_DISABLE_OS_PROBER=false
GRUB_DISABLE_RECOVERY=true
GRUB_TERMINAL_OUTPUT=gfxterm
GRUB_DISABLE_SUBMENU=y
GRUB_DISTRIBUTOR=PCLinuxOS
GRUB_GFXMODE=1024x768
GRUB_GFXPAYLOAD_LINUX=keep
GRUB_TIMEOUT=10
GRUB_THEME=/boot/grub2/themes/pclinuxos2/theme.txt
```

So, let's look at a line by line explanation:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash=silent"
```

This is a list of arguments to be appended to the Linux kernel line. In grub-legacy, we used to write these on the kernel line of each stanza. But, in Grub2 this will be done for us. Just add any kernel parameters that you need to this one place.

```
GRUB_DISABLE_OS_PROBER=false
```

If this is set to false, as it is here, then when the update-grub2 command is issued, Grub2 will use the os-prober utility to look for other operating systems, and then add them to grub.cfg to be displayed in the menu. More correctly, the script /etc/grub.d/30_os-prober will be executed (if it has execute permissions set).

```
GRUB_DISABLE_RECOVERY=true
```

This doesn't really mean what it seems to say. If set to true, then the 'safe-mode' entries, the ones that drop you to a single user prompt, will not be generated.

```
GRUB_TERMINAL_OUTPUT=gfxterm
```

This sets the terminal output device, which in this case, is a graphical terminal to enable our nice themes and backgrounds to be displayed. Not many will want/need to change this.

```
GRUB_DISABLE_SUBMENU=y
```

If this is set to 'n' then the script that generates the entries for the current OS will create an entry for the current kernel and a sub-menu entitled 'Advanced options for…' where … is the name of the above OS. Selecting this entry displays a new

menu containing entries for the other kernels or kernel options, such as text mode booting. Set to 'y' each entry gets its own line in the main menu. This can be used to keep things tidy when you have a lot of options.

```
GRUB_DISTRIBUTOR=PCLinuxOS
```

The value here is added to the first line of the stanzas for the current OS in the format --class PCLinuxOS This provides additional information for menu generation, allowing the use of OS specific icons alongside the menu items.

```
GRUB_GFXMODE=1024x768
```

The value here must be one supported by your graphics card using Vesa Bios Extensions (VBE). The value may contain an optional color depth e.g. 1024x768x32, which may be ignored if unavailable. I have found the above to be an acceptable compromise for all of my displays. This is, after all, just a boot menu. To discover which modes are available, from a grub command prompt issue the command vbeinfo. If there are so many that they scroll off the screen before you have a chance to read them, then the command set pager=1 will show them in a more controlled manner, like using less under bash.

```
GRUB_GFXPAYLOAD_LINUX=keep
```

Setting this option to 'keep' retains the value set in the option above for the next part of the boot process. An alternative is 'text' for a text only boot, or you can set an alternative resolution. The keep option is a good choice here.

```
GRUB_TIMEOUT=10
```

The delay in seconds before the default entry is booted, the countdown is cancelled by pressing any key. Set this to '0' to boot immediately or '-1' to wait for a manual selection.

```
GRUB_THEME=/boot/grub2/themes/pclinuxos2/theme.txt
```

This one I have already covered and must be the full path to the theme script.

The scripts in /etc/grub.d may, with root permissions, may be edited to customise the grub.cfg file generated by the update-grub2 utility (which is simply a link to /usr/bin/update-grub, so don't worry if you forget to add the final '2').

Before starting to edit these scripts, it is necessary to understand a little about the new format configuration file grub.cfg that is generated by these scripts.

Ignore the Martian nonsense at the beginning and scroll down to the line

```
### BEGIN /etc/grub.d/10_linux ###
```

This is the start of the part generated by the script mentioned, and contains the boot stanzas for the kernels available in the current operating system. This is the first menu entry on my system.

```
menuentry 'PCLinuxOS, with Linux 3.15.9-pclos1' --class
pclinuxos --class gnu-linux --class gnu --class os
$menuentry_id_option

'gnulinux-3.15.9-pclos1-advanced-41dedc65-55b5-4b38-8610-
b03fbb79fc9c' {

      load_video
      set gfxpayload=keep

      insmod gzio
      insmod part_msdos
      insmod ext2
      set root='hd0,msdos10'
      if [ x$feature_platform_search_hint = xy ]; then
        search --no-floppy --fs-uuid --set=root --hint-
bios=hd0,msdos10 --hint-efi=hd0,msdos10 --hint-
baremetal=ahci0,msdos10  41dedc65-55b5-4b38-8610-
b03fbb79fc9c
      else
        search --no-floppy --fs-uuid --set=root 41dedc65-
55b5-4b38-8610-b03fbb79fc9c
      fi
      linux /boot/vmlinuz-3.15.9-pclos1 root=UUID=41dedc65-
55b5-4b38-8610-b03fbb79fc9c ro  quiet splash=silent
      initrd      /boot/initrd-3.15.9-pclos1.img
}
```

The first line contains the title as it will appear in the menu, and this can be whatever you want. Next, on the same line, are a series of --class statements. If a theme has been specified, then these are read sequentially and grub tries to find an icon of the same name in the themes icon directory. If successful, the following --class statements are skipped and the icon is used in the menu display otherwise no icon is shown. We can ignore for now the rest of this line.

Next are a series of statements to tell grub which modules it needs to load to recognize the operating system. These replace the old grub-legacy stage1_5 files and, as far as I can tell, some of the stage 2 code.

After that, we tell grub where the root device is. Note: drives are counted from 0 and partitions from 1. This is a deviation from grub-legacy. Also, the partition table type is often required. Usually, this will be an mdsos or gpt type partition. Here, it is msdos.

Skip down now to the linux and initrd lines. These are really all that grub needs, and are very similar to those used in grub-legacys' menu.lst stanzas. The rest of the stuff is there to make things run smoothly if your system has special requirements.

Scroll down now to the line

```
### BEGIN /etc/grub.d/30_os-prober ###
```

This section contains the stanzas for the other operating systems that the os-prober utility has found. The format here is the same for other linux systems but may vary for other systems. Windows, for example, requires its own boot loader, which must be chainloaded from grub. Below is a stanza that does just that.

```
menuentry 'Windows XP Professional x64 Edition (on
/dev/sdb1)' --class windows --class os $menuentry_id_option
'osprober-chain-FCACFB37ACFAEB52' {
      insmod part_msdos
      insmod ntfs
      set root='hd1,msdos1'
      if [ x$feature_platform_search_hint = xy ]; then
        search --no-floppy --fs-uuid --set=root --hint-
bios=hd1,msdos1 --hint-efi=hd1,msdos1 --hint-
baremetal=ahci1,msdos1  FCACFB37ACFAEB52
      else
        search --no-floppy --fs-uuid --set=root
FCACFB37ACFAEB52
      fi
      drivemap -s (hd0) ${root}
      chainloader +1
}
```

Fortunately, these are automatically generated for us.

```
### BEGIN /etc/grub.d/40_custom ###
```

If you haven't already modified your system, then this section will be empty. This is where the stanzas that we require to be in the menu will appear.

```
### BEGIN /etc/grub.d/93_memtest ###
```

You may have an entry here if the memtest utility has been included in your system. If you do not want it to appear in your menu, remove the execute permissions from the named script, and the next time that you execute update-grub2, it will not be included.

```
chmod -x /etc/grub.d/93_memtest
```

The other scripts you can probably ignore.

Grub2, left to its own devices, will do a good job of setting up the available operating systems on your machine, but the resulting menu can soon become untidy and difficult to navigate. This next part is for those who want to take control of the menu and are prepared to do a little bit of administrative maintenance to keep things under their control.

Tidying up the menu is a relatively simple matter.

1. Run update-grub2 with its default options to populate /boot/grub2/grub.cfg with stanzas for all of your installed operating systems that grub and os-prober can find. Then copy those stanzas for the operating systems which you wish to appear in the menu to the end of `/etc/grub.d/40_custom`.

2. Edit the titles of the copied stanzas to your liking, and if you want to use icons, make sure that the first --class statement has the name of the icon to use from your theme/icons directory.

3. Edit /etc/default/grub and change `GRUB_DISABLE_OS_PROBE`R from false to true.

4. Remove the execute permissions from /etc/grub.d/10_linux with the command `chmod -x /etc/grub.d/10_linux`

5. Run update_grub2 to regenerate the `/boot/grub2/grub.cfg fil`e.

This has the disadvantage that the menu will not be updated when a new kernel is installed. In this case, reverse steps 3 & 4 above, run update-grub2 to generate the new stanza, then copy it to the appropriate place in `/etc/grub.d/40_custom`. The stanza for the previous kernel can be removed once the new kernel has been confirmed to operate correctly. Repeat steps 2 through 5.

System Recovery

Since switching to Grub2, I have had no issues with booting. In fact, I have actually had to work hard introducing errors to test out the procedures outlined below. These methods worked for me, but they are not the only way out. Device and partition ordering are one of the biggest problems in booting. Removing a drive or deleting a partition may cause the system to re-number those remaining. Grub is not notified of this and may on next boot fail to find required components. If you multiboot and often add or remove partitions, then you will benefit from reading the grub manual and from an understanding of the drivemap and grub2-mkrescue commands.

Grub2 is very stable and reliable but, stuff happens...

Unfortunately, there are times when things don't go according to plan and the system, for some reason, will not boot. If the menu displays at all, then grub has

successfully loaded, and grub is itself a mini operating system with a powerful command shell. If you have an idea of why the boot failed, perhaps the disk was not recognized, then you can try editing the menu entry by pressing 'e'. From here, the menu entry is displayed in a terminal editor, and limited command completion is supported by pressing the Tab key. You can try changing things here. The changes will not be written to disk, but may enable you to find a bootable configuration.

If your attempts to rectify the menu entry fails, then you can jump to a command line by pressing 'c' from the menu screen. Grub2 has a very powerful command interpreter, with excellent command completion features, which enable you to examine the available disks and their files.

You can even test background images for suitability with the background_Image command. For example:

```
grub> background_image
/boot/grub2/themes/pclinuxos2/my_background.png
```

You can use command completion to good effect here to enable you to locate suitable images, which must be on a mounted partition. Usually, only the boot partition is available. Typing the background_image command with no file name will clear the image.

To boot a Linux system, grub really only needs to know where the kernel and initrd are. The command for the kernel is 'linux'. The following is a transcript of a successful boot using only the command line. The text in blue is the only typing that I had to do, everything else was completed by grub.

```
grub> linux (                                          tab
Possible devices are:

 hd0 hd1
grub> linux (hd0                                       tab
possible partitions are:

Device hd0: No known file system detected - Sector size
512B - Total size
16777216KiB
               Partition hd0,msdos1: Filesystem type ext* -
Label 'mate' … Last
modification time 2014-08-17 10:48:08 Sunday, UUID
4cfb85de-9093-46c9-96fe-c2301ef9e49e - Partition start at
31.5KiB - Total size
8739328.5KiB
               Partition hd0,msdos5: No known file system
detected - Partition start at 8739391.5KiB - Total size
971901KiB
               Partition hd0,msdos6: Filesystem type ext* -
Label 'mate_home' - Last modification time 2014-08-17
10:47:01 Sunday, UUID
9a254f68-c23e-4590-acca-0b91bb89d6cb - Partition start at
9711324KiB - Total
size 7060536KiB

grub> linux (hd0,msdos1                                tab
grub> linux (hd0,msdos1)                               tab
grub> linux (hd0,msdos1)/                              tab
grub> linux (hd0,msdos1)/                              tab

Possible files are:

lost+found/ home/ dev/ etc/ mnt/ tmp/ var/ root/ proc/ sys/
boot/
Module.symvers  bin/ include/ lib/ lib64/ modules.order
null opt/ sbin/ usr/
run/ media/ initrd/
grub> linux (hd0,msdos1)/boot/                         tab

Possible files are:

config System.map System.map-3.14.4-pclos2 System.map-
3.14.5-pclos2
boot.backup.sda config-3.14.4-pclos2 config-3.14.5-pclos2
gfxmenu grub/
```

```
initrd-3.14.4-pclos2.img initrd-3.14.5-pclos2.img
initrd.img kernel.h
kernel.h-3.14.4-pclos2 kernel.h-3.14.5-pclos2 vmlinuz
vmlinuz-3.14.4-pclos2
vmlinuz-3.14.5-pclos2 initrd-3.15.7-pclos1.img System.map-
3.15.7-pclos1
config-3.15.7-pclos1 vmlinuz-3.15.7-pclos1 grub2/ kernel.h-
3.15.7-pclos1
grub> linux (hd0,msdos1)/boot/vm                tab
grub> linux (hd0,msdos1)/boot/vmlinuz           enter
grub> initrd (hd0,msdos1)/boot/in               tab

Possible files are:

initrd-3.14.4-pclos2.img initrd-3.14.5-pclos2.img
initrd.img
initrd-3.15.7-pclos1.img
grub> initrd (hd0,msdos1)/boot/initrd.img       enter
grub> boot                                      enter
```

As you can see, grub's command completion gives me all the information needed to locate the files. Of the three partitions displayed, one has no recognizable file system, as this is an extended partition which serves only as a container for logical partitions. Of the other two partitions the labels helps me to choose. I like labels. Once the kernel and initrd image have been located, the single boot command boots the system, and from there I can repair the damaged grub. Booting the system in this manner will give you a very basic, text only (no pretty plymouth theme display) start up, but it should boot.

It is worth bearing in mind that in order for grub to access the files that it needs, they must be in a readable location. Grub cannot access encrypted partitions, or some rarer or newer file systems for which support is not included. Logical volume arrays and raid set-ups may also pose problems. If this is a concern for you, then consider using a separate boot partition. The instructions for setting this up are clearly outlined in the manual, so I will not repeat them here.

There may come a time when grub doesn't get as far as displaying the boot menu, and drops you to a prompt. The reason you get here is because grub has failed to find the grub directory, the grub configuration file, or the modules need to complete the loading of grub, and so we have to help grub to find them.

Apart from the kernel and initrd discussed above, grub needs to know two more things: The location of the root device and the location of the grub2 folder, which it stores in the variables 'root' and 'prefix' respectively.

If this is a normal grub> prompt then you will have a few commands available to help you recover. To see the commands, first type

```
set pager=1
```

to prevent them scrolling off the screen then type

```
help.
```

For help on a particular command type

```
help command_name.
```

From the prompt type

```
ls
```

This will list all of the available devices and partitions available

```
(hd0) (hd0,msdos6) (hd0,msdos5) (hd0,msdos1) (hd1)
(hd1,msdos1)
```

Now you have to do a little detective work to find the files. type

```
set
```

to list the variables and their values. Look for root and prefix. We need to know just how much grub already knows. The values in root and prefix, if set, will tell us where grub has been looking for its files. Using the values found, use the ls command to verify that they are actually there. If not, then we need to locate them, and re-direct grub to the correct location by setting the variables to the actual location. Another possibility is that the configuration file has been corrupted, moved or renamed. Corruption is the more serious problem, and a reboot from a bootable cd or pen drive is the easiest solution. From the booted system, the config file can be replaced by mounting the partition and copying a corrected file over.

If the file has simply been moved or renamed, then issue the configfile command with the correct path and location of the actual file. If the file had perhaps lost the final 'g' from the file name, i.e. grub.cf, then the command

```
configfile (hd0,msdos1)/boot/grub2/grub.cf
```

will bring up the menu. This file should then be correctly renamed once the system is up and running.

If the variables root and prefix are not, or are incorrectly set to point to the actual device or path, then they need to be set with the set command. This should get you to a working boot menu.

Rescue Mode

A worse case is when you get to a rescue prompt. This is a very limited shell with few useful commands, but recovery is still possible. A rescue prompt is displayed when grub fails to find any of the files that it needs in order to continue in a 'normal' manner. That is, it fails to insert and execute the 'normal' module. We can recover even from this, despite having very few usable commands at our disposal.

The Grub2 manual should take a leaf from Douglas Adams book and, like the

'Hitchhikers Guide To The Galaxy,' have printed in large friendly letters on the front cover, the words 'Don't Panic!' Almost all disasters can be turned into triumphs with a little determination and an understanding of the Grub2 system.

In rescue mode, the prompt looks like this

```
grub rescue>
```

This leaves you in no doubt as to how deeply in trouble you are. The first thing to try here is to type

```
normal
```

to try to get grub to continue in its normal mode of operation This will probably not work, but is worth trying anyway, as grub may have just had a 'senior moment' during the boot process.

The commands available here include:

```
boot cat load insmod linux ls normal search & set
```

and not much else.

The procedure to recover is similar to those previously outlined. Use the ls command to list the available devices and partitions. Use the ls command and a partition as argument to locate a /boot directory. Use the ls command to search the /boot directory for a kernel and initrd. Use the set command to put the correct values into the root and prefix variables.

```
set prefix=(hd0,msdos1)/boot/grub2
set root=hd0,msdos1
```

Use the insmod command to load the normal module. Issue the command normal to resume grubs normal mode of operation, with the variables correctly set.

Point grub at the kernel and the initrd as in the previous example. When all is correctly set, issue the boot command.

With the system recovered, it is advisable to re-install grub with the

```
/usr/sbin/grub2-install device_name
```

command to ensure that grub will be able to find its files after a reboot.

That, and what has previously been set out, should enable you to recover from most emergencies. However, this cannot possibly cover all situations and reading the manual, which is not particularly difficult to follow, is recommended.

The grub manual is available here. An excellent introduction to developing grub2 themes is available here. Explanations of Grub2 command line use is available here.

Grub2 is a powerful and comprehensive system. It comes with many useful utilities not discussed in this article, but waiting to be discovered by more curious and adventurous users. An understanding of the basics outlined here can save much distress when things go wrong, and will allow you to customise completely the look of your system on startup with only a minimum of effort.

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, rot 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.

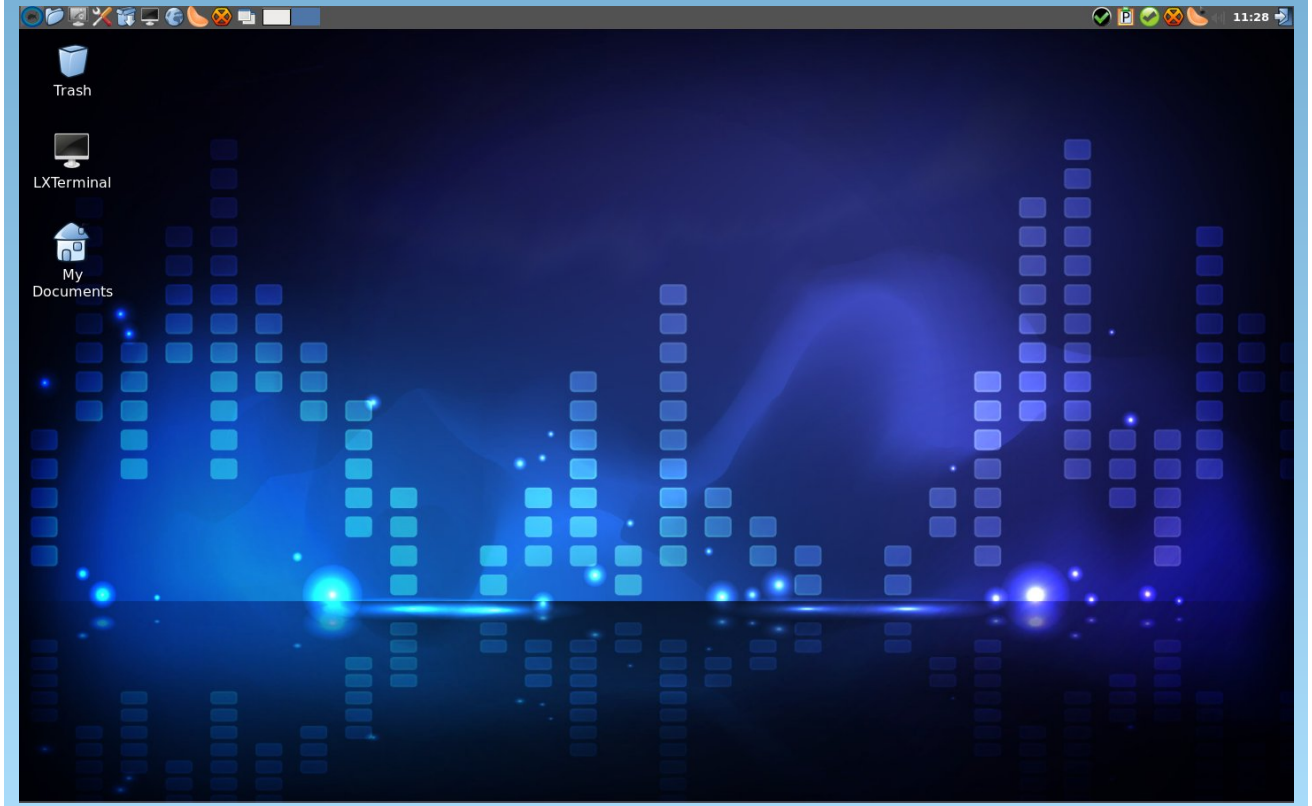
Entire Agreement
These terms constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes and replaces all prior or contemporaneous understandings or agreements, written or oral, regarding such subject matter.

The PCLinuxOS Magazine

Created with Scribus 1.4.4

Linux Docs
Linux Man Pages

Screenshot Showcase



Posted by Hertz, on August 21, 2014, running LXDE.

The New Cookie Monster: Privacy Badger

by Paul Arnote (parnote)

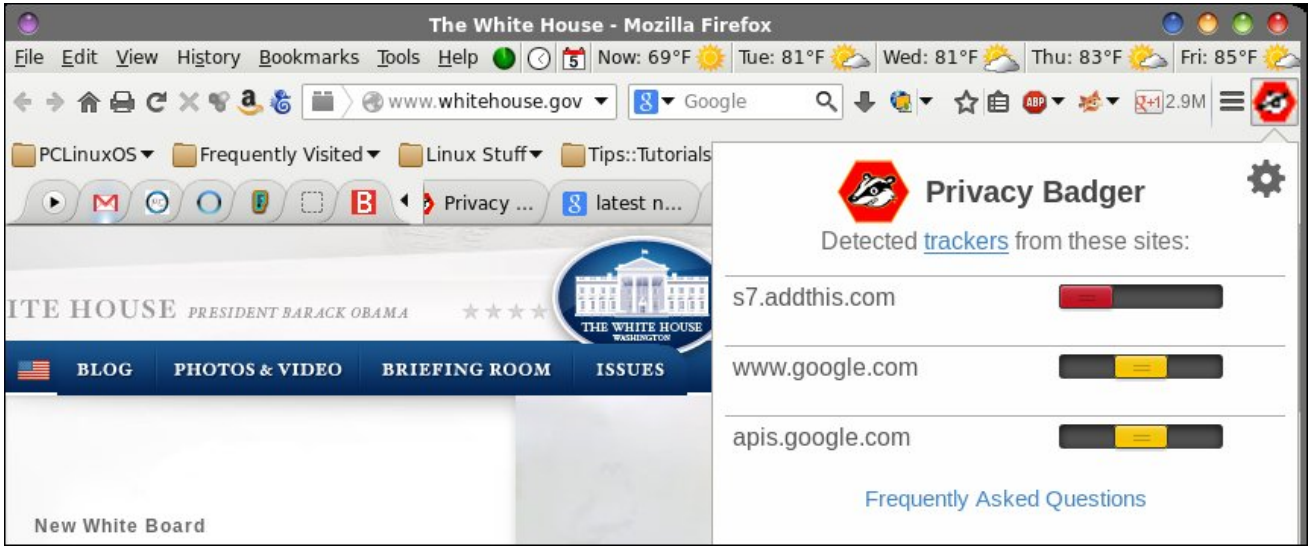
As if obtrusive online ads and illegal internet spying weren't enough, we now have a new form of tracking to worry about. Called "canvas fingerprinting," it's a sneaky trick that some websites use to track their users' internet activity.

Here's how it works. When a browser loads the page code, embedded JavaScript code leverages the canvas API that is now included in most modern browsers. This API accesses the graphics chip on the user's computer. The website asks your browser to then render a hidden image. Since each computer will render the image differently, partially dependent on the hardware in that computer, that returned image is assigned a unique number, which can then be used to track that particular user as they travel around the internet.

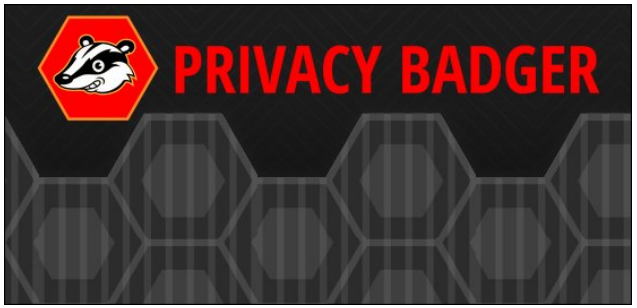
Advertisers and others who want to track users across the internet are wanting to get away from using cookies. Cookies can be blocked by the end user, and cookies can also be physically deleted by the end user. Such activities render cookies useless for tracking a user, especially when attempting to send targeted advertising to that particular user.

Canvas fingerprinting is much more stealthy. NOTHING is stored on the user's computer, so it's a lot harder for the average user to detect. In fact, most users probably aren't even aware that they are being tracked, much less that canvas fingerprinting even exists.

Unfortunately, canvas fingerprinting has infiltrated sites that you would never suspect, such as the White House website. One tracking "widget" that especially notorious for implementing canvas



fingerprinting is called AddThis. The makers of AddThis insist that the canvas fingerprinting code was removed from their widget in early July, but admitted to experimenting with it during a five month test run. Of course, none of this was divulged to the average internet user, and all tracking activity occurred without the knowledge of the users it was tracking. At one point, over 5,000 high profile sites were using the AddThis tracking widget. As the image above shows, the White House website is still using it.



So, what can a user do to protect his/her privacy? Well, you can thank the Electronic Frontier Foundation. They have created Privacy Badger. Privacy Badger eats cookies for breakfast, lunch, dinner and late night snacks. Blocking canvas fingerprinting is a feature that they are looking to add in future versions. Currently, the notorious AddThis widget is blocked, because of their failure to honor users' "Do Not Track" requests.

Here is the description for Privacy Badger, from its download page:

Privacy Badger blocks spying ads and invisible trackers. It's there to ensure that companies can't track your browsing without your consent.

This extension is designed to automatically protect your privacy from third party trackers that load invisibly when you browse the web. We send the Do Not Track header with each request, and our extension evaluates the likelihood that you are still being tracked. If the algorithm

deems the likelihood is too high, we automatically block your request from being sent to the domain. Please understand that Privacy Badger is in beta, and the algorithm's determination is not conclusive that the domain is tracking you.

Our extension has three states. ***Red*** *means Privacy Badger believes this domain is a tracker, and has blocked it.* ***Yellow*** *means the domain is believed to be both a tracker and necessary for the functioning of the page, so Privacy Badger is allowing it but blocking its cookies.* ***Green*** *means that Privacy Badger believes this is not tracker. You can click on the Privacy Badger icon in your browser's toolbar if you wish to override the automatic blocking settings. Or, you can browse in peace as Privacy Badger starts finding and eating up web trackers one by one.*

Nothing can stop the Privacy Badger from eating cookies when it's hungry!

Privacy Badger is a project of the Electronic Frontier Foundation.

Privacy Badger "report" on The PCLinuxOS Magazine website.

Currently, Privacy Badger is available for the Firefox and Chrome web browsers. Installation is easy, and follows the same procedure you would use to install any browser extension or add-on. You can control the activity of individual trackers by sliding the slider control to allow (green), allow but block cookies (yellow), or block entirely (red) the selected tracker.

Summary

The fervor over illegal government internet spying on common citizens is far from over. I suspect that it hasn't even begun to reach it heights. The last thing the common "netizen" needs is another threat to their privacy. Plus, privacy on the internet and our other modern forms of communication (cell phones, for example) is at the forefront of the public consciousness.

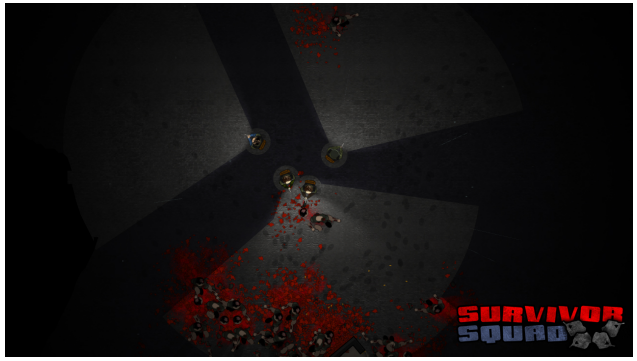
If you run NoScript, then you **should** be relatively safe from the canvas fingerprinting scheme, as NoScript should be able to prevent the JavaScript code from executing and creating that unique fingerprint. For the rest of us, it's nice to see something like Privacy Badger, where trackers are managed automatically and in the background. Given the EFF's passion for all issues related to privacy, I can think of no one I trust more to help manage my privacy and internet "footprint."

Screenshot Showcase

Posted by gseaman, on August 21, 2014, running KDE.

Game Zone: Survivor Squad

by daiashi



About The Game

Survivor Squad is a strategy action game where you control a squad of up to four survivors, and guide them through a highly randomized world looking for supplies to aid you on your journey.

Scavenge for supplies in every corner of every building, craft your gear, pick your skills and loadout while keeping your squad alive by covering every corner and moving as a group.

It is a game that encourages quick thinking and fast paced play since you need to pay attention to all of your squad mates. If you leave someone behind, chances are they will be pounced on and die. Move as a squad, and cover every corner.

The game features:

1. Top down 2D strategy action

2. Use melee weapons to take down enemies silently or go guns blazing

3. Scavenge buildings for materials to craft your gear

4. Equip gadgets to assist you in combat

5. Highly randomized world providing infinite possibilities

6. Capture infected buildings and defend them from the horde

7. Several types of special infected that must be dealt with quickly

8. Rename your survivors to have a more intense personal experience

9. Four game modes: **Campaign, Survival, Death Lab** and **Multiplayer**.

Gameplay:

Control your squad with the mouse, carefully planning where each survivor goes and where to look. The survivors have a limited view range, so it is vital to ensure you are looking where the enemy will most likely come from.

Certain events require you to move around a building while fighting off a horde, and you must keep a close eye on your squad or they will perish one by one.

Various types of the special infected have abilities, such as blinding powder or acid pool, that encourage you to quickly move your survivors, while making sure they are never alone. In campaign mode, you can capture infected buildings that generate resources, but must be defended from horde invasions.

If anyone remembers my review on Door Kickers, this title is similar, except this time you're up against hordes of zombies.

Play through a highly randomized world following a simple storyline with unique events and a conclusion.

Survival: Score based mode, go through as many randomly generated buildings as you can, never looking back.

Death Lab: Equip your survivor squad with a limited budget, and take them through a collapsed Lab. Good micro management of your squad is essential to your survival.

Multiplayer: The new multiplayer mode is a 1:1 scenario (Online/LAN), where one player controls the survivors, and another player controls the infected. The survivors must reach the extraction zone, and the infected must stop them by spawning various types of Infected around them. Will the survivors reach their extraction zone alive and well, or will they be devoured along the way? It's up to you!

System requirements:

Fully updated PCLinuxOS and Steam

Hardware:

Minimum:
* OS: PCLinuxOS
* Processor: 1.7 GHz dual core

* Memory: 512 GB RAM
* Graphics:Radeon 3850 or equivalent
* Hard Drive: 900 MB available space

About The Company

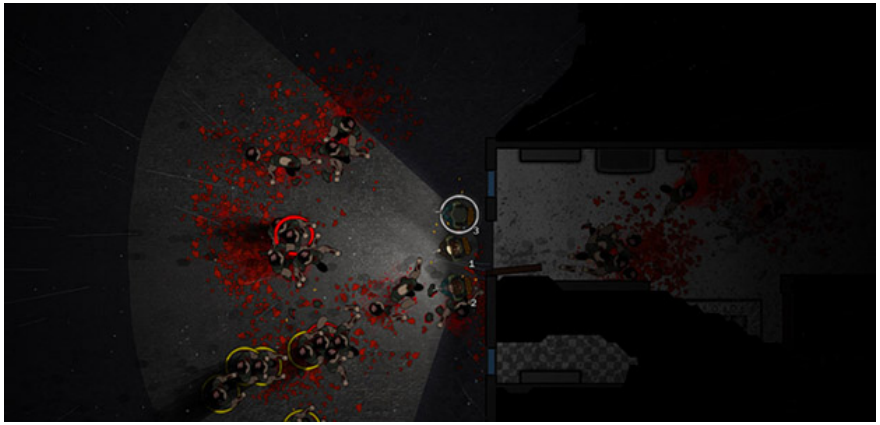
Hugo Cardoso is the owner of Endless Loop Studios and developer behind Survivor Squad. This is his first commercial PC title, after developing several successful Flash Games. I hope you like what you see. If you have any questions post them in the Steam page discussions and Hugo will do his best to answer them.

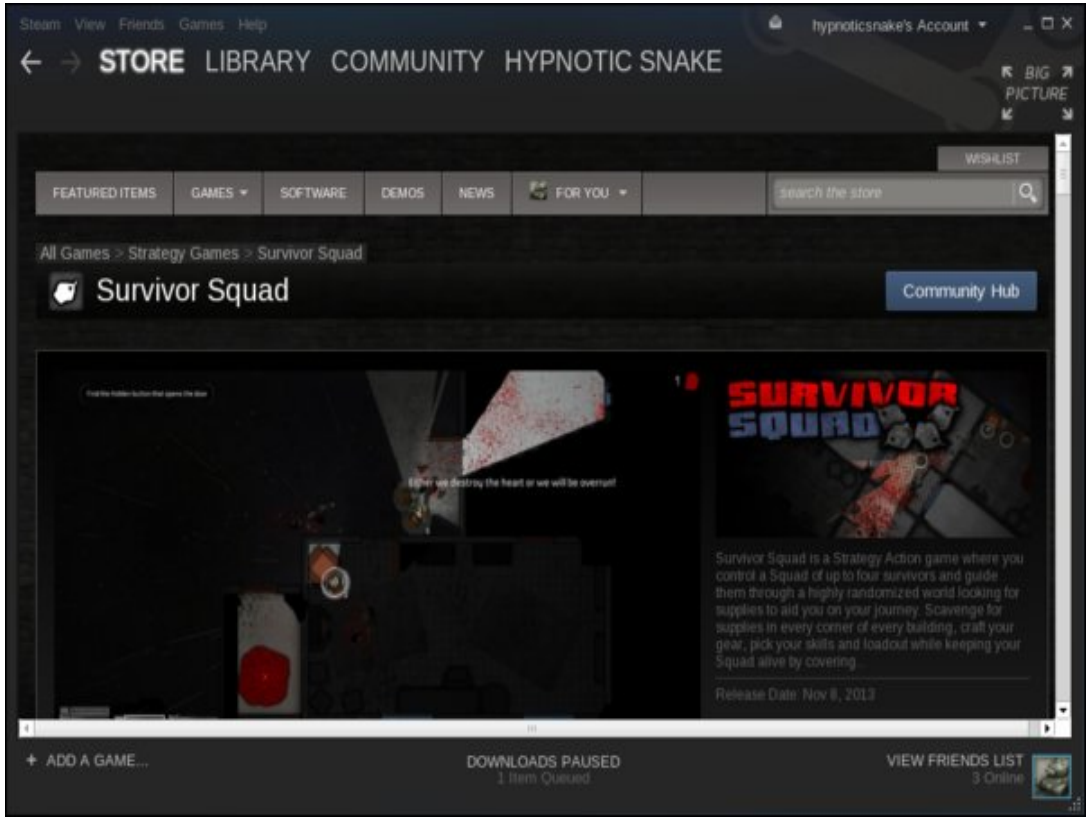
Getting It To Run

Install Steam (if you don't have it installed already), then start it. You will need to create a new account, if you do not already have one. Once you have Steam up and running, go to the store tab. Click on the Linux tab if you wish and search for Survivor Squad. Click on and download the demo. If you have updated your system, including graphics drivers, you should be good to go.

Survivor Squad

Some Gameplay Screenshots





PCLinuxOS Puzzled Partitions

	8		3					5
	4	1					7	8
2		5	8	1				
7	5	6	1					
					4	6	5	1
				9	8	5		3
5	6					2	8	
9					5		6	

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.

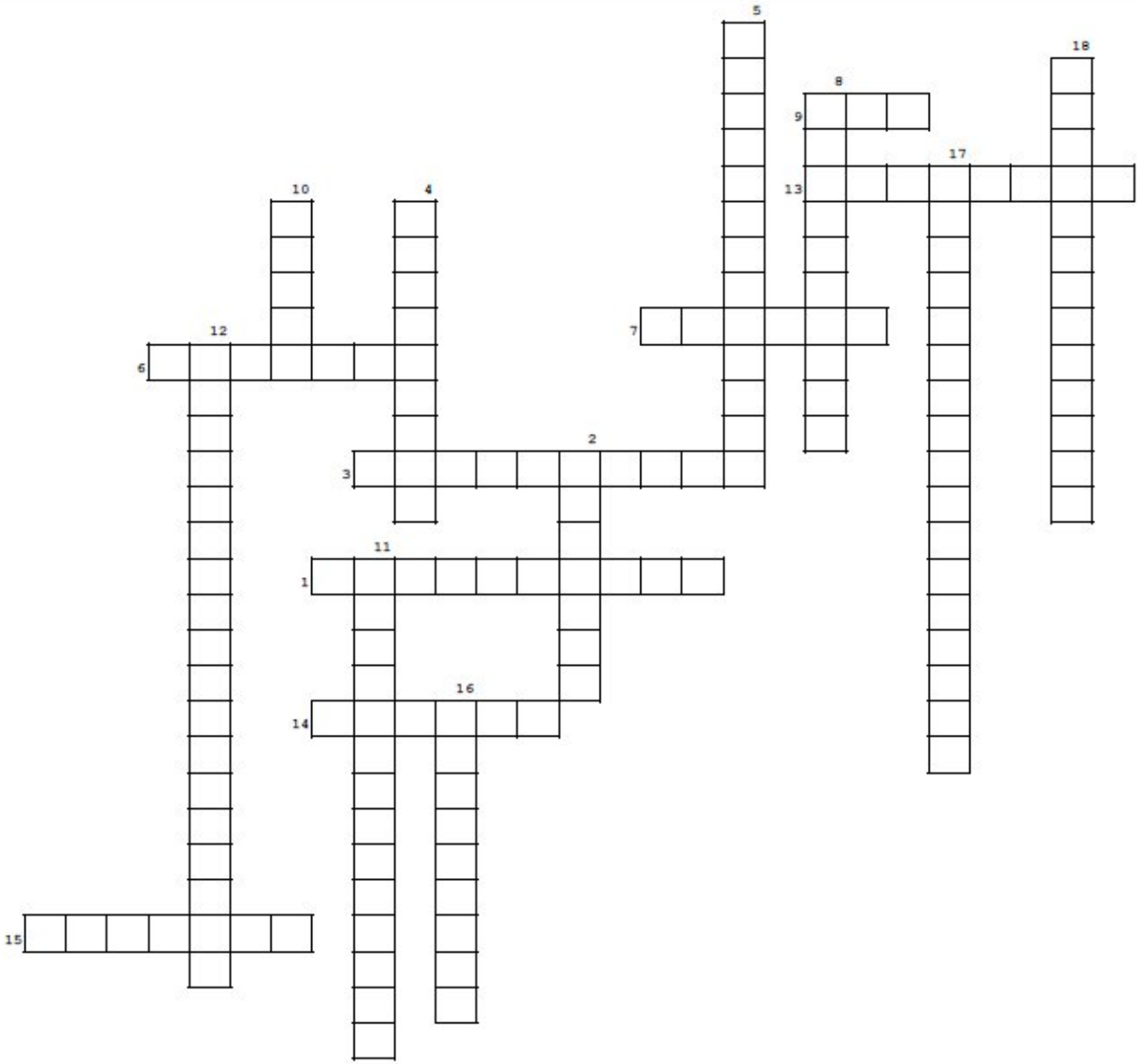
SCRAPPLER RULES:

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!

Download Puzzle Solutions Here

Possible score 248, average score 174.

PCLinuxOS Crossword Puzzle: September 2014 School Days



1. Study of behavior
2. One of the science classes about living things
3. Teachers use this to help explain the lesson.
4. Land characteristics all over the world.
5. Displays student works or important lessons.
6. The person who does the educating.
7. Learning how to make a logical argument.
8. Keeping financial records.
9. Painting, crafts, etc.
10. Singing and playing instruments.
11. Head of the school district.
12. Term papers, spelling, grammar & such.
13. Hopefully this class includes Linux!
14. To clear the chalkboard or correct your mistakes.
15. Past happenings
16. She keeps things running in each building.
17. Playing sports and staying in shape.
18. Learning to cook and sew, among other things.

Download Puzzle Solutions Here

School Days Word Find

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

ACCOUNTING
ART
BIOLOGY
BULLETIN BOARD
CHALKBOARD
CHEMISTRY
CLASSES
COMPUTER
COUNSELOR
DEBATE
DRAMA
ENGLISH COMPOSITION
ERASER
FOREIGN LANGUAGES
GEOGRAPHY
HISTORY
HOME ECONOMICS
MATHEMATICS
MUSIC
PHYSICS
PRINCIPAL
PSYCHOLOGY
READING
PHYSICAL EDUCATION
RECESS
SECRETARY
SUPERINTENDENT
TEACHER
TEXTBOOKS

Download Puzzle Solutions Here

The Z-Shell

by Peter Kelly (critter)

What is it?

When you enter a command on the Linux command line, you are typing text into a shell, which acts as an interpreter between you and the operating system. This is mostly transparent and users can simply type in their commands or execute their scripts without having to know about such mundane matters.

The shell program that most Linux distributions default to is the Bourne Again SHell, more commonly referred to as **bash**. Unix users might use the more powerful Korn shell, ksh or the c language type shell known as csh or the c-shell. If however, you spend a lot of time on the command line, then you may like to try the even more powerful z-shell, also known as zsh, which is available in the PCLinuxOS repositories.

What can it do?

These are some of the reasons to consider using zsh:

- Command completion that is extended to include both command options and arguments.
- Spelling correction
- Batch renaming
- Theming and enhanced prompting including right align and auto-hide
- Extended globbing
- Multiple redirection without tee
- Loadable modules to extend things such as math routines and network interoperability
- Multi-line command editing
- More complete expansion and substitution
- Menu selection
- Variable editing

There is a lot more, but that should give you an idea of what zsh is capable of.

How does it do this?

The GNU utilities, being text based, need a method of inputting and editing a line of text to be used by the utility. This functionality is provided by the readline library. The bash shell and many of its 'built-in' functions use this library to provide some quite impressive editing features, such as command completion.

The z shell improves on this by implementing its own z-shell line editor, or zle. This editor is mostly backwards compatible with the bash editing features, but is capable of a lot more useful editing time-savers and tricks. Options are available to make zsh more, or less, compatible with bash, and with more restrictive shells such as sh, ash and dash, as well as the POSIX shell (which confusingly insists on also being named sh!).

The z-shell is most like ksh, the korn shell, and least like csh, the c-shell. For users familiar with only the bash shell, the one thing that most will notice is the addition of more features, should you wish to activate them. Everything else should just work out of the box, so you won't have to re-learn your command line syntax. There is one particular exception to this (although I have had no problems), in the handling of multiple-word variable assignments, but even this has an option to allow more bash-like action, should it affect you.

There are many more differences in zsh, especially with respect to scripting and array handling. Array elements are indexed from zero in bash and from one in zsh for example. This behavior may be changed by setting the KSH_ARRAYS option, but all of your existing scripts should be fine, thanks to the #!/bin/bash on the first line, which tells the operating system to use bash (or whatever else you have put there: /bin/sh /usr/bin/perl...) to interpret the commands in the file.

Not all of the abilities I have mentioned above are switched on by default, and a first run would seem to be rather unexciting (unexciting is actually quite a good thing on the command line). You are presented with a first-run setup screen. This allows you to activate some of the features that distinguish zsh from more conventional shells.

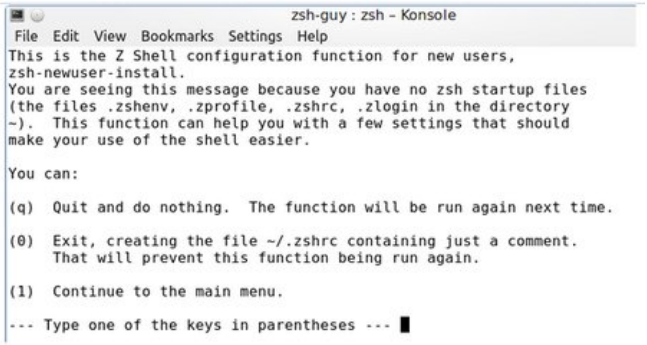
Distributions such as Arch and Gentoo have ample documentation for zsh in their respective wikis, but they refer expressly to zsh as compiled/packaged for that particular distribution. In this introduction I want to show how to get up and running with the PCLinuxOS offering, how to set up and use some of the more useful features, and perhaps to whet your appetite to investigate further.

Installation and initial set up

Open Synaptic, re-load and mark all updates, then mark zsh and zsh-doc for installation or, as root from a command line, since this is what the article is about, do this.

```
# apt-get update
# apt-get dist-upgrade
# apt-get install zsh zsh-doc
```

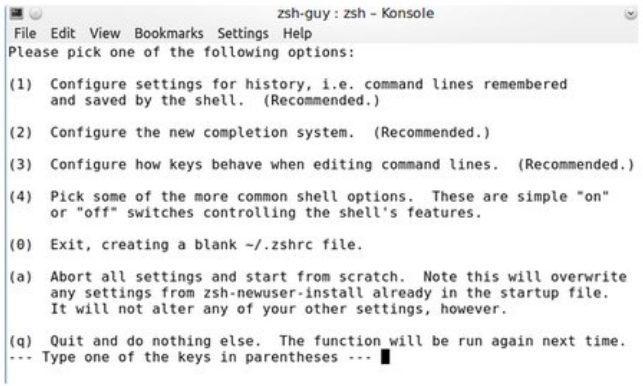
The first two commands are optional but recommended.



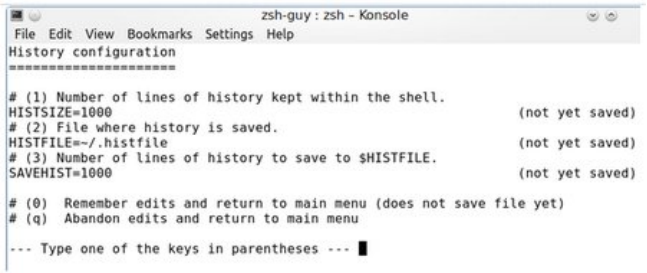
Open a command line (you will still currently be using whatever shell you usually use, probably bash) and type **`$ zsh`**.

This will open an instance of the z-shell, and since you haven't yet got any configuration files set up, you will be presented with the following initialization screen.

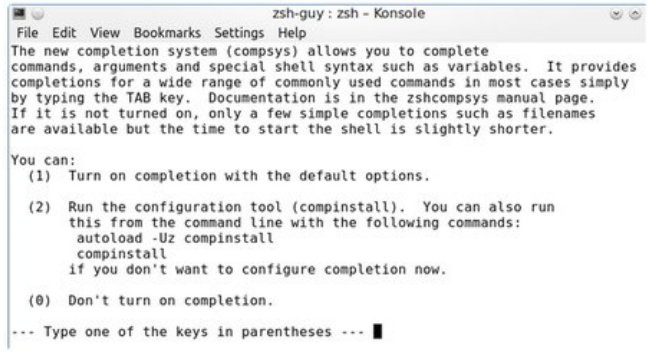
This and the following screens allow you to set up the initial working environment.



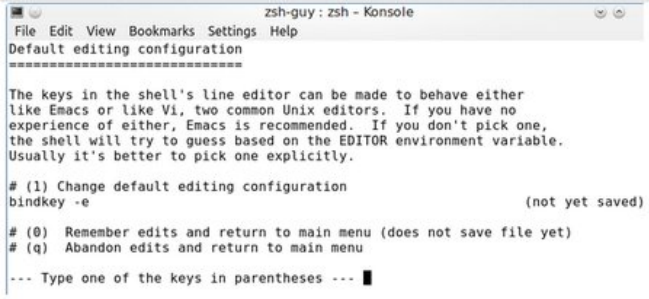
Press 1 and you will get a menu like that shown below. These options are saved to your local zsh configuration files, so they may be later changed with a simple text editor.



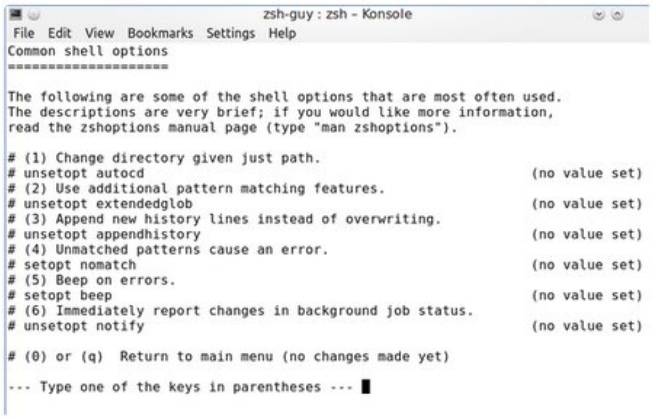
Option 1 brings up the screen below. Unless you have a reason to change any of these settings, just enter 0. Nothing will be saved until you accept the changes when you exit this utility.



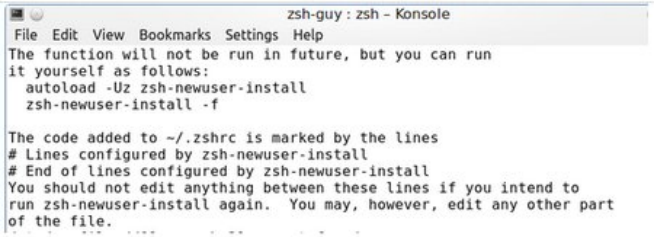
With option 2 you get the screen below. For now, just type 1 for the default options. Completion in zsh is very comprehensive, and the options can at first be somewhat confusing.



Option 3 gets us this screen, below. If you use the vi editor, you may be tempted to change the editing keys to reflect that, but bash uses emacs keys by default (e.g. control-a to go to beginning of the line). So, if you are used to those command line shortcuts, you may be better off sticking with those to avoid confusion.



Option 4 is a little more interesting. I would suggest setting shell options 1 and 2.
Type the option number and choose set.



Finally, return to the main menu and choose the option to save your settings

Now that the configuration files have been written, it is time to make zsh your default shell. This is done by editing the file /etc/passwd as root, or with the following command:

`chsh -s /bin/zsh`

You will be prompted for your password, and then it will be necessary to log out for the change to take effect.

Some terminal emulators, such as kde's konsole and mate-terminal, use profiles which may specify which shell to start on launch, so it may be necessary to edit or to change the profiles to open in zsh.

You may see the warning
/etc/profile.d/01msec.sh:21: = not found.

The script issuing this warning is one of msecs security scripts which checks for the current directory being in the PATH (a known security issue and PCLinuxOS wisely sets this to 'no' by default). The reason for the message is because of the way that zsh handles tests. The code which is being complained about is, as stated in the message, line 21 of /etc/profile.d/01msec.sh:

```
if [ "$ALLOW_CURDIR_IN_PATH" == "yes" ]; then
```

The problem is the handling of '==='. Change the old-style test operators [ ] to the newer, preferred operators [[ ]], zsh will be happy, and the messages will go away. This should be perfectly safe, as the result of the test is the same. I have had no problems anyway. As far as I know, the new style is not defined in the POSIX standard, so be aware of that if compliance is an issue for you.

Note that any aliases and functions defined in your .bashrc or .bash_profile files are not available here, as this is a new environment (although functions can be exported from bash to a sub-shell with export -f).

Command completion

You are now in the z-shell. Although it may not look very different – well maybe the prompt, and we'll change that soon – but try this. You will no doubt be familiar with bash command completion, but now type a command that you regularly use but forget the options of. let's type mount, type a space, then one or two dashes, then press Tab. Now zshells completion prompts us with all of the options for the command (long options only, if you entered two dashes), along with their descriptions. Type the first letter or two of the option, and press Tab again for zshell to complete the typing for us.

Completion is something that zsh does extraordinarily well. If you look at your .zshrc you will find the lines

autoload -Uz compinit
compinit

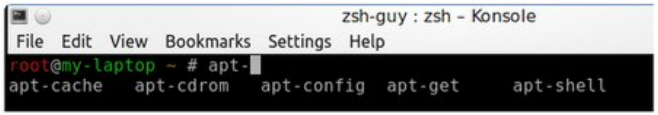
This means that the module *compinit* which sets up completion in zsh has been loaded and executed.

Menu selection

Sometimes you will get a great many completion options, and tabbing through them all would be very tedious. So zsh provides a feature known as menu select. Add this line to the end of your .zshrc file:

```
zstyle ':completion:*' menu select
```

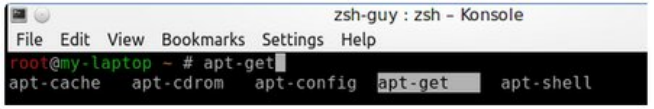
Now suppose, that as root (I'm presuming that root has also migrated to zshell), you want to install the gnumeric spreadsheet application, but are unsure of the correct package name to pass to the package management tool.



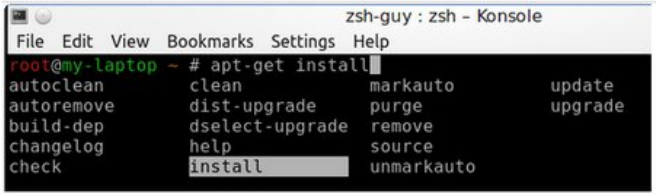
Type

apt- (tab)

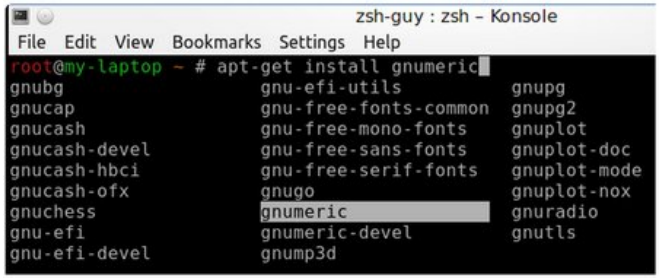
You will be presented with a list of available commands.



Press tab again and the first one will be highlighted. Now use the arrow keys to select apt-get.



Type a space then the tab key twice to get a list of sub-commands to apt-get and then arrow over and down to the install sub-command

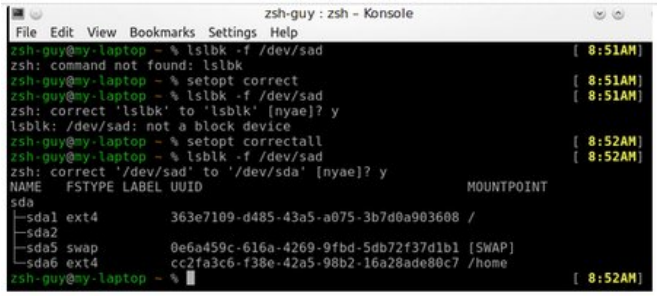


Now type a space, followed by 'gnu', then two tabs, arrow over to gnumeric, and press return. Apt will list any dependencies that also need to be installed/upgraded before asking for confirmation to go ahead and install.

Although zsh knows a lot of completions for a lot of commands, it is inevitable that you will find some command whose options are unknown to zsh. In that case, zsh allows you to program your own completion commands as script functions, but that would perhaps be too much for this introduction.

Spelling correction

There are two options that control spell checking, *correct* and *correctall*. The first provides spell checking for commands, while the second checks arguments. Look at this screen-shot.



The command that I wanted to execute was

```
lsblk -f /dev/sda
```

but both lsblk and the argument */dev/sda* contain typos. Pressing enter, the shell responds that it doesn't recognize *lslbk* as a command and aborts. With the *correct* option set, recalling the command and pressing enter the shell now recognizes the spelling error and prompts to correct it with

zsh: correct 'lslbk' to 'lsblk' [nyae]?.

Typing 'n' means no correction. Perhaps you have an alias called lslbk not known to the spell checker. A 'y' means go ahead and change it, 'a' means abort the command and 'e' displays the line for editing, just in case you don't like the offered correction.

With 'y' entered, the command is evaluated, but now we get a complaint that device */dev/sad* does not exist. Setting correctall and trying again, the shell now offers to correct */dev/sad* to */dev/sda*. Typing 'y' corrects the argument and the command is executed correctly.

Having these two *setoption* statements in your .zshrc makes all of this happen automatically.

Batch renaming

Normally when you want to rename a file, you use the mv command to move it from one name to another. That is the way that it has always been done under Unix/Linux. If you needed to rename a batch of files, perhaps changing a common extension say .zip to .z, then things are not so simple.

You could set up a routine to loop through a set of files, identifying possible candidates, or you could use *find* and *xargs* or *exec*. You could also install one of the utilities that has been written through the years by others facing this same, wearisome task. However, if you are running zsh, you could use *zmv*.

The *zmv* command is not available by default, and must be loaded by the command

```
autoload -U zmv
```

Usually this would be put into your .zshrc file. The basic use of *zmv* is the same as *mv*. That is, **`mv source dest`** becomes **`zmv source dest`**.

But there is one very important difference. If we introduce a wild-card character (also known as a globbing operator) into the source, then we can re-use the expanded glob as a back reference in the destination. Taking our zip example this is how it works.

```
zmv '(*).zip' '$1.z'
```

The quotes are needed to prevent the shell from 'interfering' before the zmv command gets hold of it, and the parentheses are used to create the back reference. We can use more wild cards and create more back references. If we had files like this (unlikely I know, but...)

onetestcase.zip
twotestfile.zip
.....

and we needed to rename them like this

one_results_case.z
two_results_file.z
.....

Then we could simply use this command:

```
zmv '(*)test(*).zip' '$1_results_$2.z'
```

and the renaming would be done.

Personalizing and prompting

Now, about that prompt. Add these lines to .zshrc

```
autoload -U promptinit
promptinit
```

This loads the promptinit module and then executes it. zsh comes with quite a few pre-defined prompts which you can list with

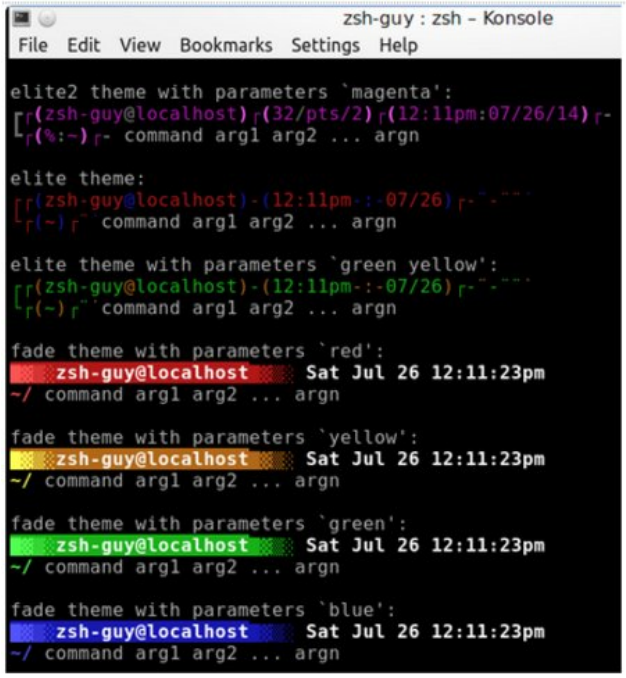
```
prompt -l
```

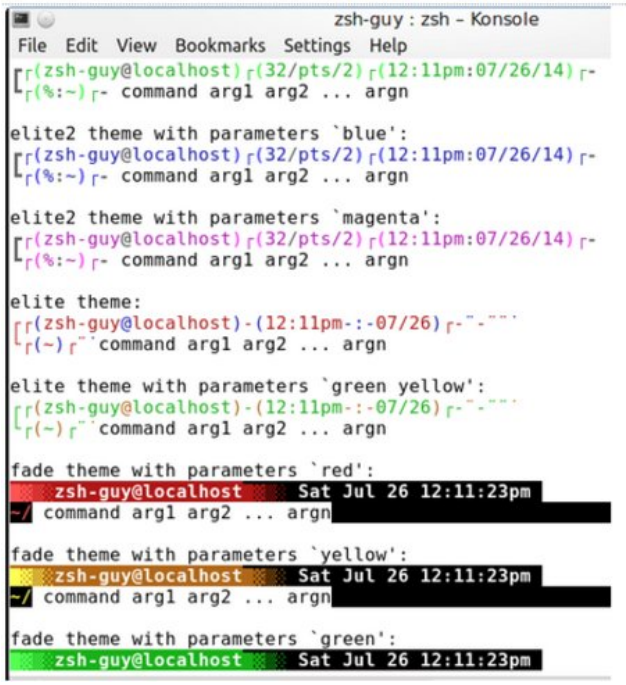
or preview with

```
prompt -p
```

Depending on your terminal background color you will get something like these previews (next page, left).

To set the prompt to one of these, simply add the name and parameters to .zshrc after the initialization of promptinit. For example, to set the prompt to the





fade theme with the parameter 'blue', enter this in the configuration file:

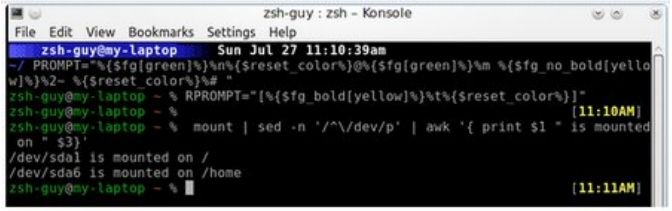
```
prompt fade blue
```

Note: The default prompt for normal users is usually '$', but the default in zsh is '%'. I like this as it reminds me that I am using zsh, not bash. The prompt for elevated privilege users (e'g' root) remains '#'.

If you prefer to construct your own prompt then fine. Zsh even allows a left and a right aligned prompt, with the right hand prompt auto-hiding as your typing approaches it. Colors are handled differently in zsh, so before defining a colorful prompt you should add:

```
autoload -U colors
colors
```

to the resource file.



The prompt is set using PROMPT= and the right prompt using RPROMPT=.

The prompt is assembled from a string of text and variables, which are described admirably on the Arch Linux wiki here.

Notice how the right prompt disappears out of the way as you type longer commands. Once you have it as you like it, save it to .zshrc, after the required autoload functions.

If you want more personalization, then you can install one of the community driven frameworks, such as Robby Russell's oh-my-zsh (you will need to install git and wget from the PCLinuxOS repositories, Oh! And back up your .zshrc before installing). Oh-my-zshell comes with over 120 themes and a heap of plug-ins, modules, aliases and functions that enable you to completely transform your command line experience.

It is generally frowned upon to install anything from outside of the repositories, but I believe that it is safe to make an exception for this package. However, you will have to decide for yourself. If you want to try it, then enter this at the terminal.

```
wget --no-check-certificate
https://github.com/robbyrussell/oh-my-
zsh/raw/master/tools/install.sh -O - |
sh
```

That will set up everything on your machine. Restart zsh, or logout/in, and then start reading the oh-my-zsh docs.

Extended globbing

File-name globbing is the strange term used when we substitute wildcard characters, such as '*' and '?' for characters we don't know. The command

```
ls -ld D*
```

will produce a long listing of all directories that begin with an uppercase 'D', followed by zero or more characters. We can also specify ranges, classes and groups of characters or integers. While all of this is very useful, zsh takes it a stage further.

A very useful feature is recursive globbing, which enables you to find any file matching a pattern in the current and all lower level directories.

```
ls **/.*rc
```

This will find all hidden configuration files in the current directory tree. The double asterisk is used to

trigger this. If you want to include symbolic links, then use three asterisks ***.
To find all files except those matching the pattern, precede the pattern with a '^'.

```
ls -d ^D*
```

This finds all directories that do not start with uppercase 'D'.

Another useful feature introduced into the zsh version of globbing is qualifiers. This is an expression, in parentheses, used at the very end of the search pattern, which restricts the output to only qualifying files.

```
ls -l /etc/*(#q@)
```

This example will find all symbolic links in the /etc directory. Use '.' For ordinary files, '/' for directories, '@' for links, '=' for sockets, 'p' for pipes. There are many, many more listed in the documentation. You may also find useful: * for executable files, and r, w, x for owner read, write and execute enabled files.

We even have a form of 'fuzzy' matching in zsh which will try to match files even if they are misspelled. This is useful when you are not entirely sure of the target name.

```
ls -d (#a1)Ducuments
```

will correctly identify the directory Documents. The number 1 in the parentheses is the number of mistakes to forgive but if you are too forgiving then you will get too many matches, change the 1 to 10 to see what I mean.

Multiple redirection

This one is easy and so useful. In bash, to copy the output from a command to a file and also display the output on a terminal, you would have use the tee command but you are limited to two data streams.

```
cat test_result | tee copy1
```

In zsh, you can do this:

```
<test_result >copy1 >copy2 >copy3
```

and get as many copies as you like (I don't know what the limit is). That's just one input and three outputs. If you want it to also appear on screen, then pipe the output to a final cat command like this:

```
<test_result >copy1 >copy2 >copy3 |
cat
```

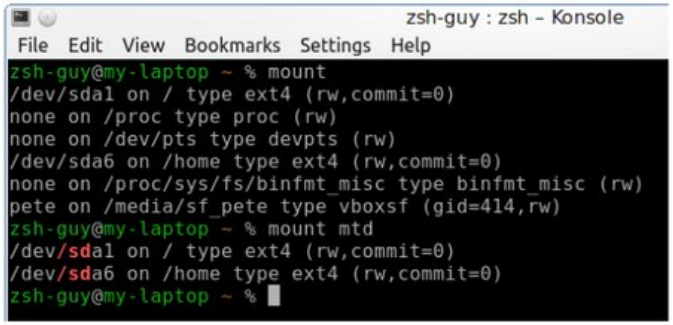
I think that's a lot more logical.

Global aliases

This is an extension to the already useful alias provided by bash-like shells. Almost anything can go into a global alias, and it can be used almost anywhere. When I type the command 'mount' on the command line, I get more than I usually need. The output is littered with virtual and non-hardware related mount points when I usually am interested in only the drives the system has marked as mounted.

A global alias takes the option -g, and I have one defined like this

```
alias -g mtd='| grep /sd'
```



Notice that I can include the vertical bar pipe symbol within the alias definition.

Alias suffixes

Most modern desktop environments allow you to associate certain file types with a particular application. With z-shell's alias suffix feature, this is now possible on the command line. Use alias with the -s option, the file extension to associate as the name of the alias, and then add the application to use. Here are a couple I like from the z-shell wiki.

```
alias -s txt='less -rE'
```

This one allows you to type the name of a file with a .txt extension on the command line and then press enter to examine the file using less. You are returned to the command line when you quit.

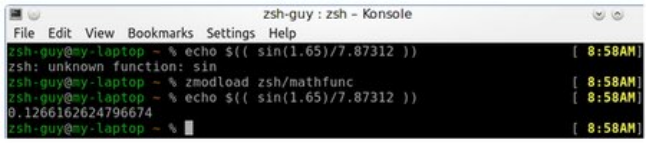
This next one I really like.

```
alias -s
html="/usr/share/vim/vim62/macros/less
.sh"
```

Press enter on any .html file, and it is opened in less, but with vim's syntax highlighting.

Math

Floating point arithmetic is enabled by default in zsh. Integers behave as in bash.
Echo $((7/3)) produces the result 2, while echo $((7.0/3.0)), which in bash produces an error, zsh determines to be 2.3333333333333335 – close enough for most purposes.



For more serious number crunching, the bc and dc utilities used to be called upon, but zsh has a very good math module. Load it with the command

```
zmodload zsh/mathfunc
```

You really do need to read the documentation if this is of use to you, as it covers almost all the math that non-Einsteins would require.

Auto CD

One of the things that I recommended activating in the initial set-up was 'change directory given just path.' This adds 'setopt autocd' to your .zshrc. This is very useful, since the cd command is one of the most used commands in a terminal – but now you can forget it, almost!

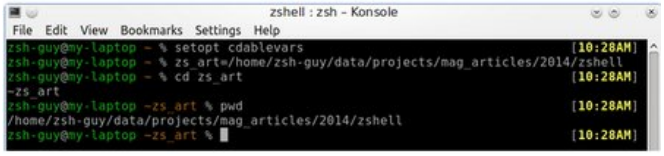
With this option set you simply type the name of the directory to go there.

```
/etc          change to /etc
~             change to home directory
..            go up a directory
/usr/share/icons/locolor
              change to this directory
../hicolor    change to
              /usr/share/icons/hicolor
```

Naturally tab completion works to help typing the directory names.

Another option that may sometimes be useful is named directory paths. This is activated with the command

```
setopt cdablevars
```



If you have some directories that you often use, but they are buried deep in the directory structure, then you can set a variable to point to that directory. Then cd to the *variable* and, yes, you do need to type cd here.

The name of the *variable* goes into the prompt, so make sure that you use meaningful, to you at least, variable names. I think you can see from the time prompts on the right of the screen-shot how much all this completion and automation improves production, and I am not a fast typist.

Using the directory stack

Most, if not all, shells variants have some means of manipulating the directory stack but I have never seen a system as flexible as the one in zsh. Bash uses the commands pushd and popd to good effect, but it is quite easy to get lost when changing rapidly between many directories. The advantage that zsh gives us is to be able to directly manipulate the stack. This is in addition to the traditional pushd and popd commands.

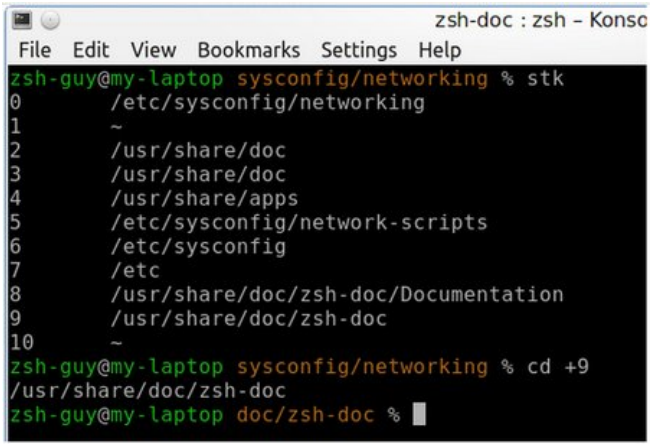
In a long terminal session, I may visit dozens of directories, some of these buried deep in sub-directories and often I need to re-visit some of them from far away in the depths of another directory.

With the option autopushd set, this is so much easier. Each directory that you visit is pushed onto the stack and you can view the current stack with the following command:

```
dirs -v
```

I'm so lazy, I even use an alias for this short command:

```
alias stk='dirs -v'
```



Now you can simply cd to the numbered directory. Use 'cd +n' or 'cd -n' to go to the nth directory from the top or bottom of the stack respectively. Here is a short demonstration of it in action.

And more…

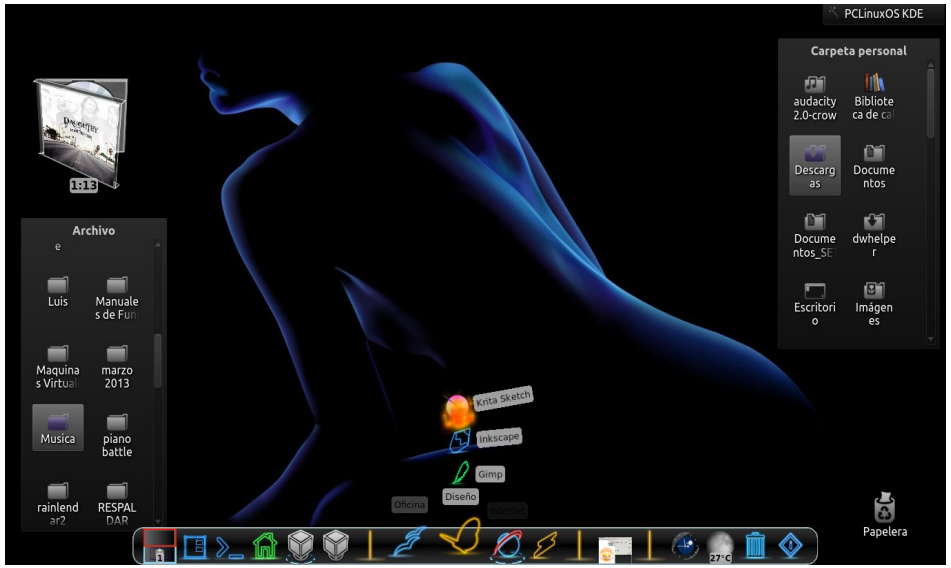
Yes, there is more. This introduction has barely scratched the surface, but I hope that it has shown enough to give you the flavor of zsh. Although a lot is possible with zsh, it is also possible to use it just like bash, or you can add just the bits you like.

The documentation is good, although a little technical. There are a set of man pages, and even functions to help you find things in them, and there is also a ton of info on the internet.

More Screenshot Showcase

Posted by francesco bat, on August 20, 2014, running IceWM.

Posted by ff103, on August 12, 2014, running KDE.



Posted by Crow, on August 23, 2014, running KDE.

Posted by Only Human, on August 8, 2014, running e17.