

The PCLinuxOS magazine

Volume 103

August, 2015



Table Of Contents

- 3 Welcome From The Chief Editor
- 4 Turn A Sequence of Images Into A Movie
- 6 Screenshot Showcase
- 8 Inkscape Tutorial: Creating A Glass Of Juice
- 11 PCLinuxOS Recipe Corner
- 12 Screenshot Showcase
- 13 "Do No Evil": Has Google Lost Its Way?
- 16 ms_meme's Nook: PCLOS Nights
- 17 Screenshot Showcase
- 18 Make Your Own Streaming Internet Radio Program
- 29 Tip Top Tips: Brother Printer Driver Installation
- 31 Screenshot Showcase
- 32 Game Zone: DungeonRift
- 34 PCLinuxOS Family Member Spotlight: BobK54
- 36 Screenshot Showcase
- 37 Playing Angry Birds on PCLinuxOS
- 41 Inkscape Tutorial: Abstract Wallpaper 2
- 43 Screenshot Showcase
- 44 Setting Up Your Own Chat Room in
PCLOS-Talk Using Pidgin
- 46 PCLinuxOS Puzzled Partitions
- 49 More Screenshot Showcase

The PCLinuxOS magazine

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

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

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

This release was made possible by the following volunteers:

Chief Editor: Paul Arnote (parnote)

Assistant Editor: Meemaw

Artwork: Sprogy, Timeth, ms_meme, Meemaw

Magazine Layout: Paul Arnote, Meemaw, ms_meme

HTML Layout: YouCanToo

Staff:

ms_meme

Meemaw

Gary L. Ratliff, Sr.

Daniel Meiß-Wilhelm

daiashi

YouCanToo

Pete Kelly

Antonis Komis

Smileeb

Contributors:

Khadis

Agent Smith

The PCLinuxOS Magazine is released under the Creative Commons Attribution-NonCommercial-Share-Alike 3.0 Unported license. Some rights are reserved.
Copyright © 2015.



Welcome From The Chief Editor



Well ... I must first start off by offering up my apologies. If you've been following the Kodi article series, there isn't a Kodi article this month. Between work, being Mr. Mom on the days I'm not working at the hospital, and getting sick twice this past month, I simply didn't have sufficient time to do the Kodi article, as I had planned.

The first time I got sick, it acted more like a summer case of the flu, with fever, chills and body aches. Then, as I was getting over that, I got struck with a mid-summer cold. Thankfully, I already had one article for the August issue completed, and another that was already in the formative stages.

I plan to resume the Kodi article series in the September, 2015 issue. I thought it was better to skip a month, rather than unnecessarily delay the release of the magazine, or publish a half-hearted, half-baked, hastily created Kodi article.

My son, Ryan, got the "bug" rolling through our house. He came down with it first, but you could hardly tell, except for the persistent runny nose he had. Then, my wife got it, and really suffered. She ended up missing a couple of days of work with it. Just when I thought I had dodged the bullet, I got the first go around. After medicating myself into oblivion (and starting to feel better), I picked up a secondary "bug," which manifested itself as a mid-summer cold. YUCK! Back to the medications for a second time.

Working in the hospital, as my wife and I do, we're no stranger to proper handwashing and the other ways to limit the spread of pathogens. But after so many times of wiping Ryan's runny nose, the "chain" got broken somewhere along the line. So our efforts of keeping the "bug" contained – wiping everything down frequently and well, proper handwashing, proper disposal of contaminated tissues, etc. – failed.

In reality, I suspect that this is probably just a prelude to when he goes to school, where he's surrounded by other kids who are sent to school ill by their parents, just so the parents don't have to miss work. For this reason especially, elementary schools are germ factories. The kids, bless them, don't know enough about proper hygiene to prevent the spread of illnesses and diseases. So, the "bug" has a fertile breeding ground. When I used to work as a newspaper photographer, I hated going to cover stories at elementary schools. Without fail, I'd always end up sick, picking up some bug from someone's kid who was sent to school sick, all because the parents didn't want to miss work to stay home with their sick child.

But, if the price to pay for being a parent is picking up all these little "bugs" he brings home, I'll gladly "pay" it. August is the month for my son's birthday (August 6), as well as my birthday (August 25). There's not a day that goes by that I'm not grateful for Ryan, given how we went through seven years of infertility before he finally came into our lives. Every day is amazing, watching him learn new things, and watching him evolve into such a cool, unique little person. Happy second birthday, son!

Until next month, I bid you peace, happiness, serenity, prosperity ... and good health!

Turn A Sequence Of Images Into A Movie

by Paul Arnote (parnote)

While I don't advertise it much, I actually love to hunt. I love to hunt for deer, squirrels, rabbits, quail, pheasant ... pretty much any and all wildlife that is legal to hunt in my neck of the woods. Until this year, I've always hunted with a rifle or a shotgun. But this year, I've purchased a compound bow. It'll give me a chance to finally learn how to bow hunt, as well as expanding my deer hunting "season" considerably.

I'm also not a "trophy hunter." I'm more about hunting to help provide meat for my family, so we eat the meat from my successful hunts. The meat is leaner and more healthy than anything you can buy in the local supermarket, as nature meant it to be, and free from artificial growth hormones and excessive antibiotic use. While it's nice to bag "trophy" game, in the end it's all about harvesting the meat.

See? Get me on this topic, and I'll ramble on forever.

My best friend and I set up a feeder to draw the deer onto his land at the Lake of the Ozarks, and get them into the habit of coming to the property long before deer season begins. We were curious about what wildlife was showing up at the feeder, since the feeder was being literally drained every couple of weeks of 50 lbs. (22.7 Kg) of cracked corn. Plus, deer are very plentiful in this area of Missouri. So, I purchased a [game camera](#) from one of my favorite online merchants, [Sportsman's Guide](#).

This game camera (some people call them trail cameras) takes 5.0 MP color images by day, and 1.3 MP infrared images by night. The night images are aided by the use of a black infrared flash. It's motion activated, so any motion in front of the camera



causes it to take a photograph. It stores its images on a SD memory card. We had put a 4GB card in, thinking it would be sufficient to store a couple of weeks of images.

Boy, were we ever wrong! When my friend went to swap out the SD card after a couple of weeks, he discovered that the game camera was "turned off." He thought that the batteries must have been exhausted. Doubtful, he put in the new, blank SD card and noticed that the camera turned on again when he pressed the power button. Upon closer examination, the game camera turned off because the SD memory card was full!

In three days, we captured over 3,500 images of wildlife coming to the feeder. Below are a couple of images from the first SD card.



A fawn makes its way across the camera's field of view (lower left) during this daytime color image.



Late at night, an infrared image shows a deer running off a raccoon from the feeder.

Turn A Sequence Of Images Into A Movie

We captured images of not only deer coming in to feed, but also raccoons, squirrels and birds taking advantage of the feeder to gain access to an easy meal. We even captured the images of a few bats that flew across the camera's field of view.

Now, 3500+ images are a lot to look through. Even after eliminating the images with squirrels, raccoons, bats and other critters, I was left with almost 1200 images of deer coming in to feed. That is still a large number of images to view.

So, it got me to thinking ... what if I could turn all of these images into a movie? That would be easier to view the images, en masse, than clicking through each one of them, one at a time, in GPicView (my favorite picture viewer on Xfce).

Are there other ways to view a slideshow of images? Yes, there are. In fact, GPicView has the built in ability to do it. But what's the fun in that? What would/could I stand to learn by merely changing the amount of time to display each image in the slideshow controls?

Fortunately, there are multiple ways to create a movie from a sequence of images. I first tried **Avidemux**, which can import sequentially numbered image files. Unfortunately, Avidemux didn't work so well, for me. Your mileage may vary. To start off, Avidemux didn't like the JPG format that the game camera used to save the files. So, I had to convert the images all to PNG files, which did work better. But then, after importing 400 or so images (I did it by dragging 99 images at a time from my file manager, then dropping them on the Avidemux window), Avidemux would crash. Poof! It just went away, disappearing from my screen. I tried this four times, with the same results each time. So the Avidemux route was a bust.

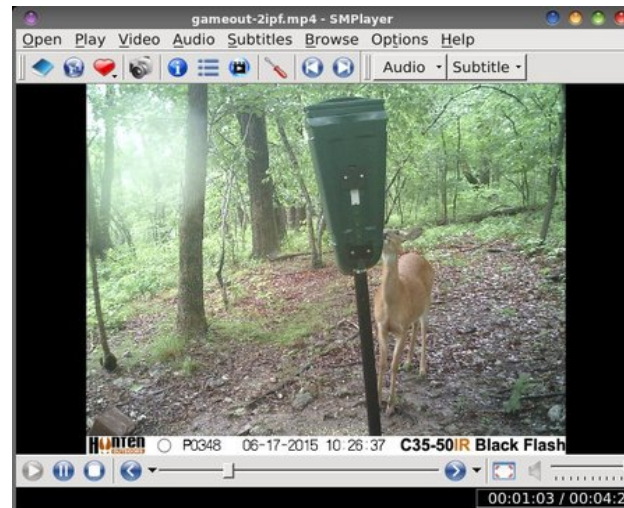
Still seeking a way to do this with a GUI program, I then tried **Winff** from the repos. While it "appeared" to go through the motions of creating the video, no video was created in the end. But it sure put on a

good "show" in the process. So, Winff ended up being a bust, as well.

The next two methods involve using the command line. The first method uses **ffmpeg**. To get started, open up a terminal session, and **cd** to the directory containing your image sequences. Next, enter the command, as below (all on one line):

```
ffmpeg -f image2 -r 1/1 -pattern_type glob -i './*.JPG' -c:v libx264 -pix_fmt yuv420p gameout-1fps.mp4
```

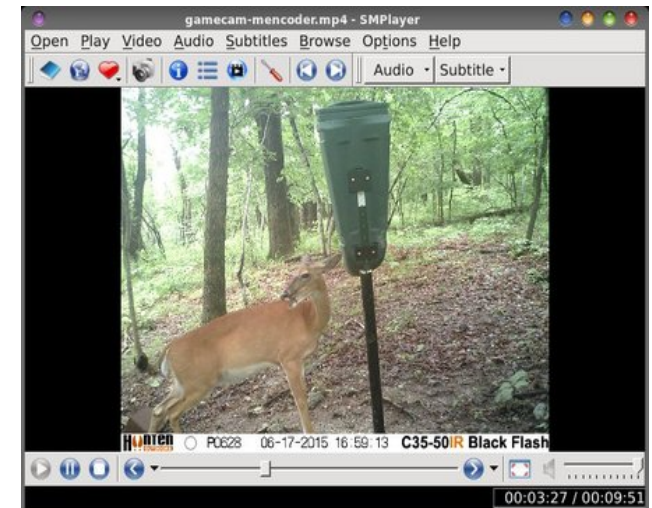
The **-f image2** part of the command is optional. It tells ffmpeg to force the input format to image files. The **-r 1/1** part of the command tells ffmpeg to display one picture every one second. If you replace this with **2/1**, it will display two images per second. Similarly, if you specify **1/2**, ffmpeg will display each image for two seconds. The **-pattern_type glob -i './*.JPG'** part of the command tells ffmpeg to find each image with the JPG file extension in the current directory. The **-c:v libx264** part of the command tells ffmpeg to use the libx264 video codec to create the video. The **-pix_fmt yuv420p** part of the command tells ffmpeg how to reproduce the pixels of the images in the resultant movie. Finally, the **gameout-1fps.mp4** part of the command specifies the name of the output file.



The second method uses the **mencoder** command line utility. To use it, open a terminal session and **cd** to the directory that contains all of your sequential images. Next, enter the command as follows:

```
mencoder mf://*.JPG -mf fps=2:type=jpg -ovc x264 -x264encopts bitrate=1200:threads=2 -o outputfile.mp4
```

The **mf://*.JPG** part of the command tells mencoder to use all of the JPG files in the current directory to make the movie. The **-mf fps=2:type=jpg** sets the frames per second to two. Since each image counts as a frame, setting the fps to two means that two images will be displayed every second, each one for half of a second. If you replace this with **-mf fps=1:type=jpg**, each picture will be displayed for one second. Similarly, if you replace this with **-mf fps=3:type=jpg**, each picture will be displayed for one third of a second. The **-ovc x264** part of the command tells mencoder to create the output movie using the x264 video codec. The **-x264encopts bitrate=1200:threads=2** part of the command tells mencoder to encode the video at 1200 bits per second, and to use two threads to complete the task. Finally, the **-o outputfile.mp4** part of the command tells mencoder the name of the output file to which to write the resulting movie file.



If you want to see the actual “movie” I made, you can view it [here](#), on YouTube. Subsequently, the second SD card filled up with 4,920 images (more 1.3 MP nighttime images than 5.0 MP daytime images, hence the greater number of images stored on the card), over the course of about five or six days. I sent my friend off with a 16GB card, which should take between 10 and 14 days to fill up (judging by the rate at which the 4GB SD cards were filled up). The first 4GB SD card filled up in about three days. So, the 16GB SD card should do the job.

Summary

Are there more command line parameters you can add to either of the commands? You betcha! It all depends on how complex you want to make your movie. You can add audio to your movie by including audio settings for either of the commands.

While I used a hunter’s game/trail camera to produce my image sequence, you could easily do this with any group of images you want to turn into a movie. Have a bunch of vacation photos you’d like to share with friends and family? Have a bunch of personal photos you’d like to turn into a movie? This method can be used to turn **any** group of images into a personalized movie. With the addition of a soundtrack (the addition of sound adds very little to the overall file size), you can easily make viewing your photos a very unique and memorable experience.

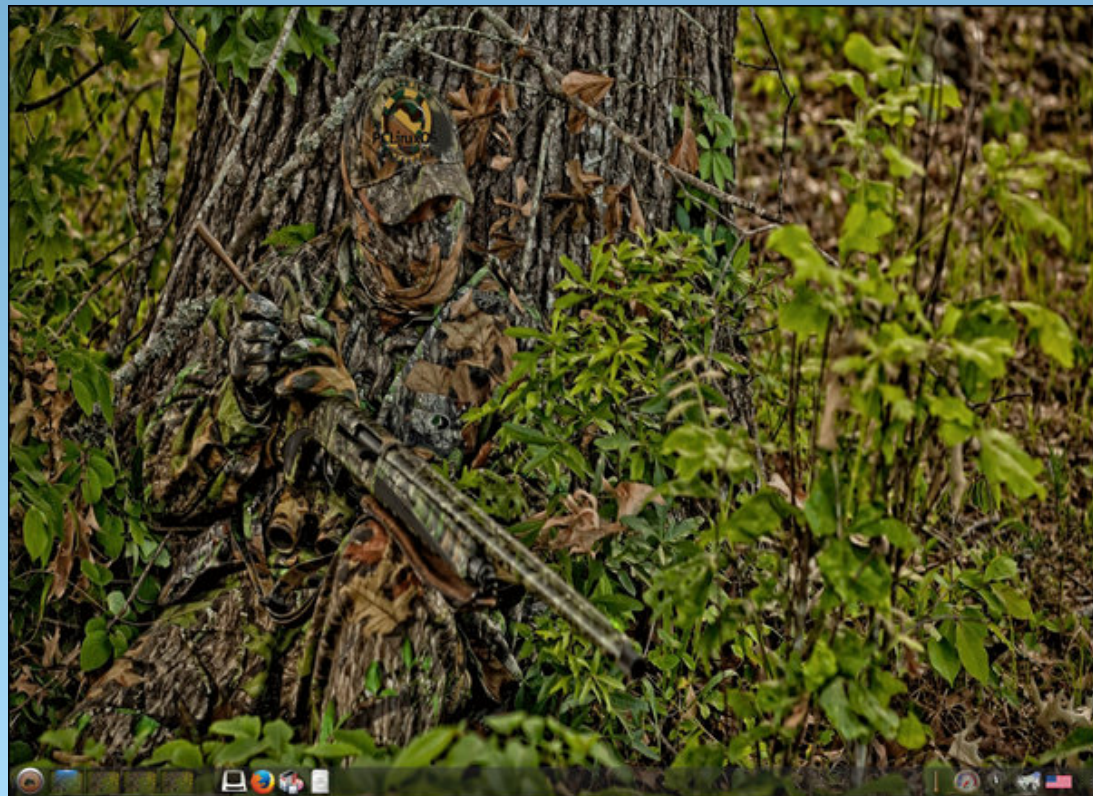
Since ffmpeg will basically import the images at basically full size (resizing them so that they are a multiple of 16), you can add a `-s 800x640` parameter right after the `-r 1/1` parameter to resize the images – and the resulting movie – to 800 x 640 pixels. You can also add the `-aspect 16:9` (or aspect 1.7777), `-aspect 4:3` (or aspect 1.3333) command in a similar location within the command line structure to control the display aspect ratio of the movie you create. Use caution, because you could cause your resulting movie to be distorted. You can also control the video

bitrate in the ffmpeg command by inserting the `-b:1200` parameter in the similar position within the command line. The `-b:1200` parameter will restrict the video output to 1200 bits per second.

With the mencoder command, you can also change the bitrate, which will alter the finished size of your

movie file. Increase the bitrate, and the size of your movie file will increase. Decrease the bitrate, and the size of your movie file will decrease. Be careful to not decrease the bitrate by too much, though. Decreasing the bitrate will have an adverse affect on the quality of your image. Personally, I would consider a bitrate of 1200 the minimum level that

Screenshot Showcase



Posted by OnlyHuman, on July 6, 2015, running e19

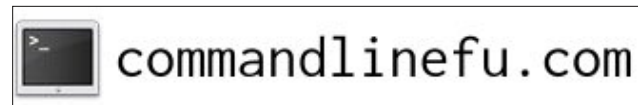
preserves adequate quality. You can also add the `-aspect 16:9` (or `-aspect 2.35:1`) command just before the `-ovc` command to stretch your movie out to 16:9 display aspect ratio. Again, use caution, because you could cause your resulting movie to be distorted.

Additionally, you can set the command(s) up in a bash file, if you like. You can make it as fancy as you want, or as simple as you want. Fancy would be adding either Zenity or YAD dialog boxes to prompt for input and to display output as the movie is processed. Using critter's (Pete Kelly's) excellent articles as a guide, I'll leave it as an exercise for you to pursue, should you choose (and I'd love to see what you come up with, if you choose to pursue this route). Simple would be just placing each command (set up in whatever manner works best for your needs) into its own separate bash file, then calling that bash file from whatever directory contains your image sequences, via a terminal session.

One final thing ... executing these commands are CPU and memory intensive. They really give the CPU and your memory a workout. So, if you have the option, perform the conversion on a multicore processor with ample amounts of memory. If you do, you'll save yourself a considerable amount of time. Will it work on a single core processor with as little as 2GB of RAM? Yep, it sure will. In fact, the movies I created here were on a laptop with a single core

Celeron processor and only 2GB of RAM. It took a bit longer, but it still got the job done ... so don't shy away from this, even if all that you have is a single core processor with modest amounts of RAM.

However you decide to do it, making a movie file of your favorite image sequences is relatively easy to do, and a lot of fun. You have the added benefit of being able to watch your images the same way, every time, without having to click through each individual image.

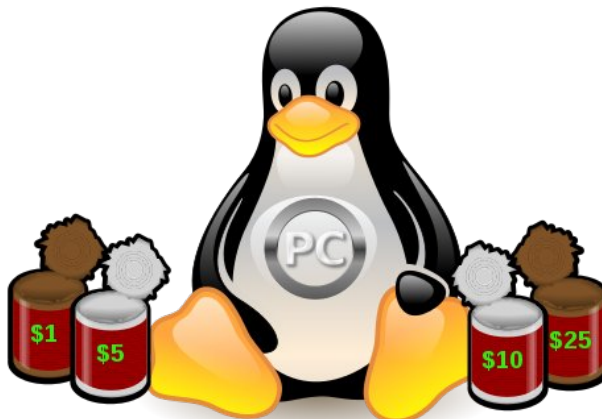


Donate To PCLinuxOS

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

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

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



International Community PCLinuxOS Sites

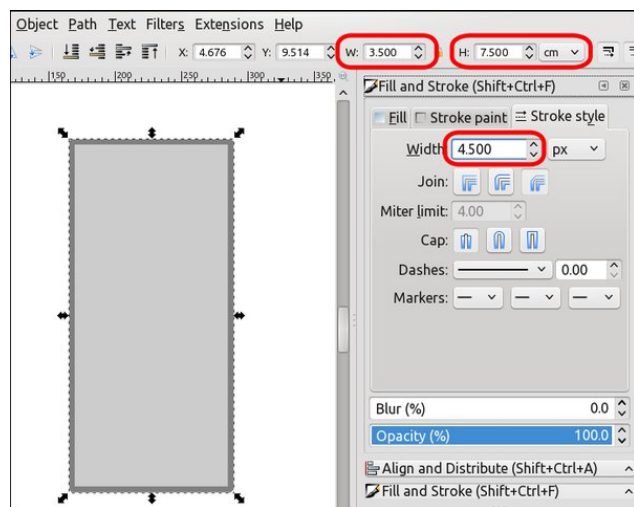


Inkscape Tutorial: Creating A Glass Of Juice

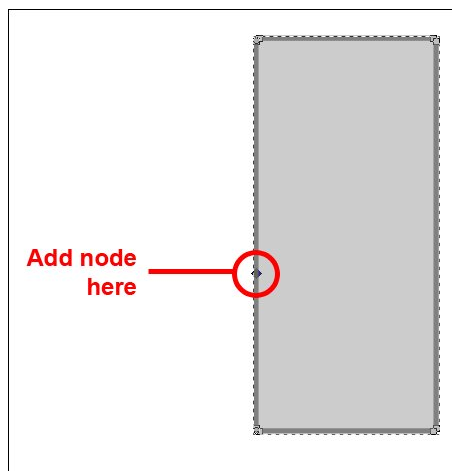
By Khadis

We can find a lot of clip art about foods and beverages out there on the internet. Do you ever imagine creating your very own clip art? If you do, let's create a piece of clip art using Inkscape. The clip art image we are going to create is a glass filled with orange juice.

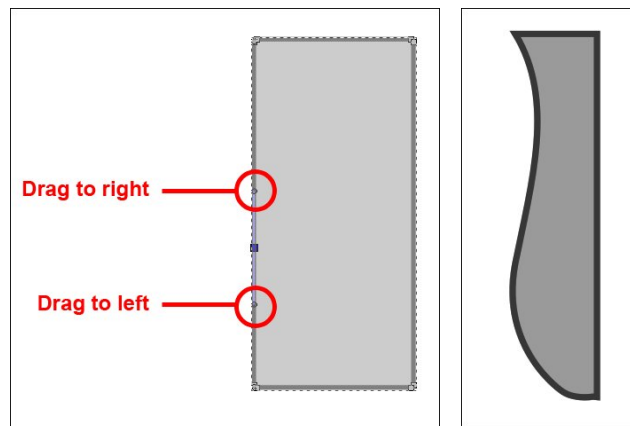
Step One: Open up your Inkscape and create a rectangle with your own preferred size. I used 3.5 cm x 7.5 cm. If necessary, use **Zoom (Z)** tool to get a better view. I used a 20% gray color fill and used 50% gray color for the stroke. I also set the stroke width to 4.5 px.



Step Two: Convert the rectangle into a path using the **Path – Object to Path** menu or by using the **Shift + Ctrl + C** shortcut. Modify the rectangle using the **Edit paths by nodes (F2)** tool. Add a node on the left side of the rectangle. Look at the picture at center top:

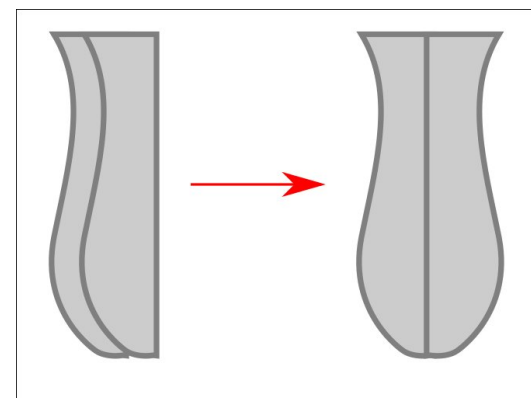


Step Three: Click the new node, then click the upper handle and drag it to the right to bend the left side of your rectangle. You can repeat this step until you get your most perfect shape. You can also move the other handles to make more adjustments.

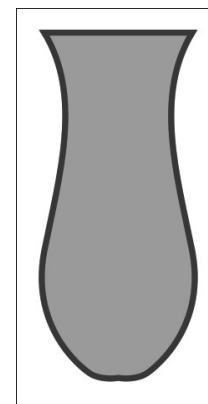


Step Four: After you get the shape you want, you can do the same steps above to be applied to the right side of the (ex-) rectangle. Or, you can

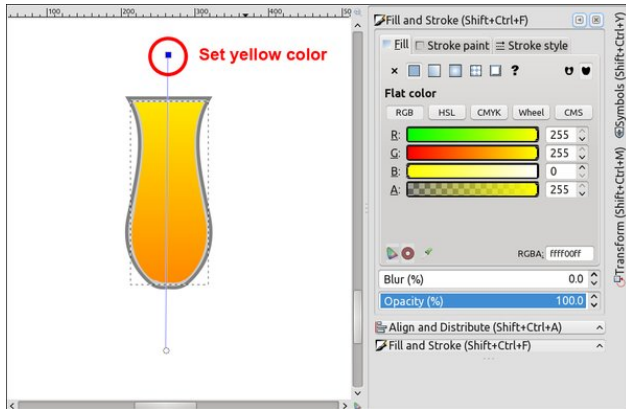
duplicate this shape (**Ctrl + D**), then flip it horizontally. Adjust the position to get the following image.



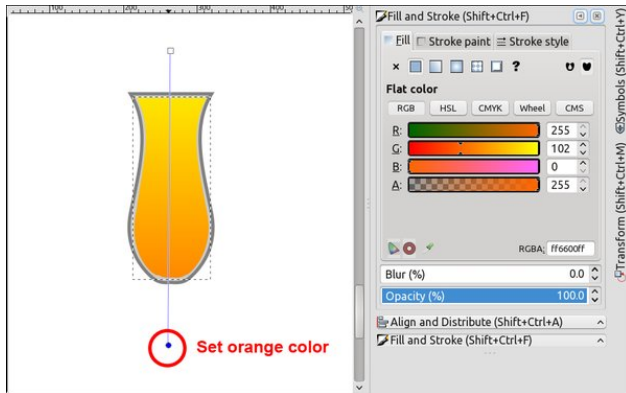
Step Five: Press **Ctrl + +** to combine the shapes so your new shape will be like this:



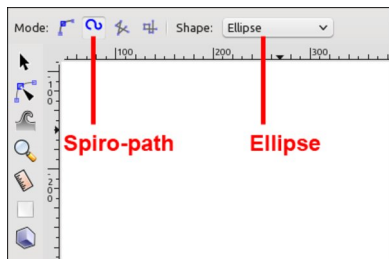
Step Six: Now, to create the juice inside of the glass, you just need to copy the glass shape, remove the stroke, and resize it. Fill it with a linear gradient, from yellow to orange. First, you need to choose the **Gradient (Ctrl + F1)** tool, then drag it from top to bottom. Click the first node (top) and choose a yellow color.



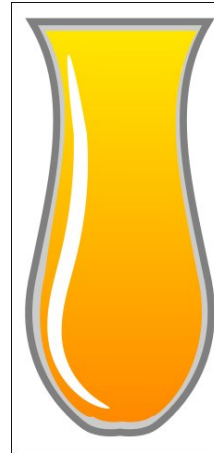
Click the second node (bottom) and choose an orange color. Look at the following color parameter through **Fill and Stroke (Shift + Ctrl + F)** panel.



Step Seven: Is the color combination satisfying for you? Now, let's create "accessories" on the glass. Grab your **Bezier (Shift + F6)** tool. Choose Spiro-path mode and Ellipse on the Shape drop down menu.



Step Eight: Draw a free curve by clicking the tool until you get the shape like this:



Step Nine: Now, fill this shape using a gradient color. So, access the **Gradient (Ctrl + F1)** tool again. Click the first node (top) and choose a white color. Click the second node (bottom) and choose a yellow color. You can always change the nodes (top or bottom) to get a better gradient.

You can draw another shape using the **Bezier (Shift + F6)** tool so that it will look like this:

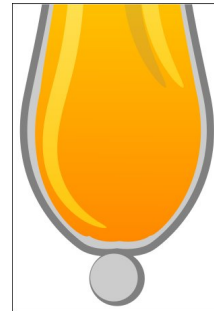


Step Ten: Now, create a circle and put it under the all shapes above. Give it a 50% gray as the color.

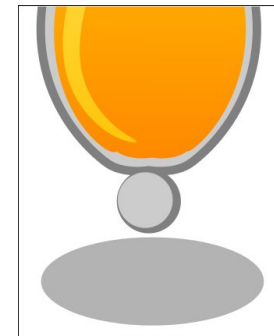
Look at the illustration below.



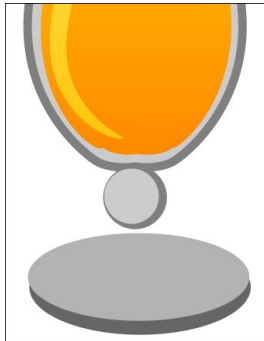
Step 11: You can also create another circle or duplicate and resize the current circle, then give it a 20% gray color. Place the second circle like this:



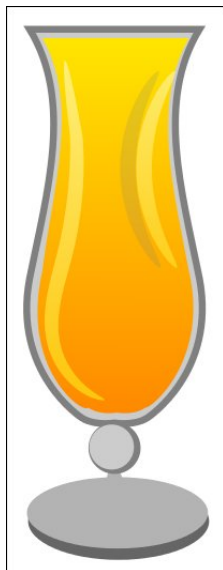
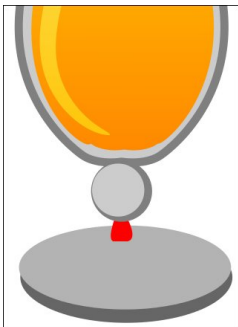
Step Twelve: Create an ellipse and put at the bottom of all the objects. You can use any proportional size. For the color, I used 30% gray color and no outline/stroke.



Step Thirteen: Duplicate (Ctrl + D) the ellipse and fill it with 60% gray color. Then, move it a little bit lower and move it a bit to right to create a shadow effect. After that, press the Page Down button on your keyboard to send the shadow ellipse to the back of all objects.



Step Fourteen: At last, you can create a “connector” between the circle and the ellipse as shown in the illustration below. The “connector” can be made of a small rectangle with a 30% gray color without stroke. On the picture below, I modified the rectangle a bit and gave it a red color so that you can spot my “connector”.



Other possible results:



PCLinuxOS Recipe Corner



Breakfast Egg Rolls

Ingredients

1 box Betty Crocker Seasoned SkilletTM hash brown potatoes
Butter or Margarine called for on box
6 eggs
2 tablespoons water
1 tablespoon butter
1/3 cup bacon pieces
1 package (1 lb) egg roll skins
2 cups shredded Cheddar cheese (8 oz)
Salt and pepper, to taste
Vegetable oil, for frying

Directions

1. In 12-inch skillet, cook hash browns as directed on box. Set aside.
2. In large bowl, beat eggs and water with whisk. In 10-inch skillet, melt butter; cook eggs in butter. After eggs are cooked, stir in hash browns and bacon pieces. Add salt and pepper.
3. Place egg roll skins on large work surface. Spoon 1/3 cup egg mixture on center of each skin. Top each with 2 tablespoons shredded cheese.

4. For each egg roll, bring up bottom corner; fold in side corners, and roll up. Bring down top corner; wet with small amount of water to stick.

5. In 8-inch skillet, heat oil until hot. Fry egg rolls in oil until crispy; shake off excess oil, and place on cooling rack to cool slightly. Serve immediately.

Options:

You can change up the flavor profile by using cooked sausage or ham instead of bacon.

Serve with a dipping sauce like salsa, a cheese sauce or even syrup, depending how adventurous you are feeling!



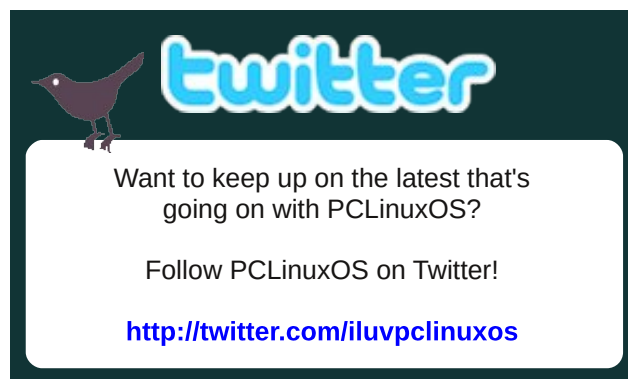
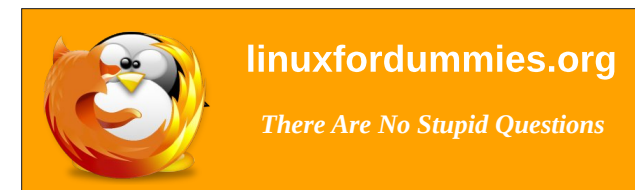
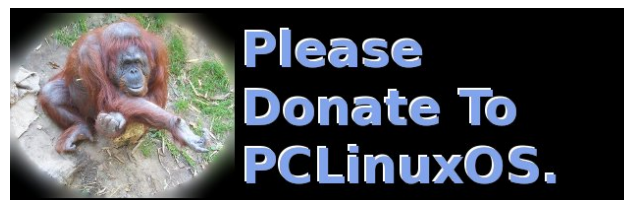
A magazine just isn't a magazine without articles to fill the pages.

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

pclinuxos.mag@gmail.com

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





Posted by Orion, on July 20, 2015, running openbox

“Do No Evil” : Has Google Lost Its Way?

by Paul Arnote (parnote)

Google. You either love them, or you love to hate them. For many modern internet consumers, it's a little of both. Without a doubt, Google has earned every ounce of love and hate that comes their way. There's love for many of their services, and there is vicious hatred that is rooted in their business practices and with whom they've chosen to hop into bed, so to speak. For some internet consumers, their hatred is so strong that they refuse to use any Google services, or services that rely on Google to provide their “backbone.”

I was reminded of how mixed the emotions/feeling are towards Google when I stumbled across an old article from 2011 about the U.S. government's investigation of Google for antitrust activities. That article led me to the [prospectus](#) that Google had filed with the SEC (the U.S. Securities Exchange Commission) prior to the IPO (initial public offering) of their stock, when Google became a publicly traded company in 2004. It was, certainly, interesting reading (especially since I had never read it before).

To gain a better understanding, we first have to look back at the founding of Google. If you're interested, you can find a blow-by-blow, in-depth accounting here (there are omissions, however, such as the “failed” Google Wave). But, in a nutshell, Larry Page (current CEO and cofounder) and Sergey Brin (co-founder and current director of special projects) met at Stanford University in 1995, and started Google in September 1998 (the domain name was registered in 1997). They start out Google in a garage in Menlo Park, California. By December, PC Magazine recognized Google as the “search engine of choice,” and in the top 100 websites in 1998. Google



Adwords launched in 2000, providing Google's primary source of cash flow.



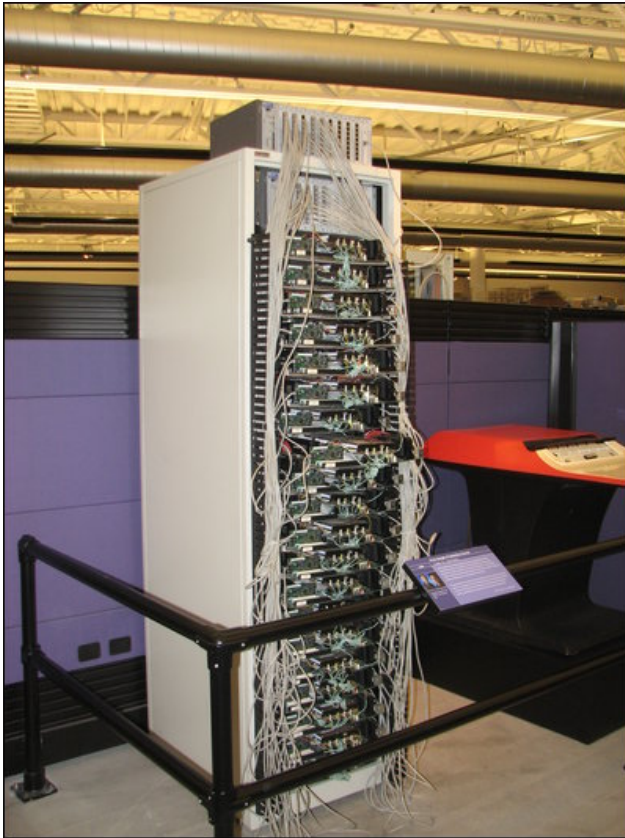
Page and Brin in their first “headquarters,” a friend's garage in Menlo Park.

In December 2000, Google Toolbar (a browser plugin) was released. Current Executive Chairman Eric Schmidt joined the company in 2001 as CEO, and Google Images (a search engine for images) was started. They also went “international” in 2001,

opening their first “international” office in Tokyo, Japan. 2002 saw the launch of Google Labs, Google News and Google Shopping (originally called Froogle until its name was changed in 2012). In 2003, they acquired Blogger, and launched Google AdSense, Google Grants, Google Code Jam, and Google Print (since renamed to Google Books). In 2004, they launched Orkut, Gmail, and acquire Picasa – all before their IPO in August 2004.

So, by the time of Google's IPO, they already had a lot of steam in their boilers, so to speak, and they were experiencing phenomenal growth. They initially offered 19,605,052 shares of Class A common stock for sale on Wall Street, with an opening price of \$85 per share. Since then, Google has introduced many other services, such as Google Maps, Google Scholar, Google+, the Chrome web browser, the Chrome OS ... and the list goes on and on and on.





Google's first server stack, 1998.

Here are a few infamous, albeit naive, excerpts from that prospectus:

On serving end users:

Sergey and I founded Google because we believed we could provide an important service to the world – instantly delivering relevant information on virtually any topic. Serving our end users is at the heart of what we do and remains our number one priority.

On long term focus:

As a private company, we have concentrated on the long term, and this has served us well. As a public company,

we will do the same. In our opinion, outside pressures too often tempt companies to sacrifice long term opportunities to meet quarterly market expectations. Sometimes this pressure has caused companies to manipulate financial results in order to “make their quarter.” In Warren Buffett’s words, “We won’t ‘smooth’ quarterly or annual results: If earnings figures are lumpy when they reach headquarters, they will be lumpy when they reach you.”

The “Don’t Be Evil” clause:

Don’t be evil. We believe strongly that in the long term, we will be better served—as shareholders and in all other ways—by a company that does good things for the world even if we forgo some short term gains. This is an important aspect of our culture and is broadly shared within the company.

Losing the shroud of innocence

Up until Google’s emergence as **the** powerhouse of the internet age, the most frequently vilified entity was Microsoft. The folks in Redmond, Washington must be somewhat happy to have someone else take some of the heat off of them in the villain department. It’s only natural to take pot shots at the “big guy” at the top, the one who’s calling the shots, especially when those below them feel that they have no say in things.

Suspicious about Google’s activities really got started when they began scanning Gmail users’ email to target “relevant” ads to Gmail users. Google claimed (claims) that the mail isn’t actually “read.” Rather, they claim, the emails are “scanned” for keywords to indicate interests of the user so ads can be offered up with greater relevance. Still, many users cried “foul!” Despite Google’s assertions, many users wondered if Google was collecting this information from their emails, what parts of it were they saving?



An image in one of Google’s top secret data centers and network (photo from Google Careers).

Next, came problems with Google automatically opting in users to various services. Instead, Google said users not interested in these various services could opt out. Users argued that users interested should be the ones to opt in, instead of forcing users to opt out if they lacked interest in the new services. This seems to be a problem that continues today, albeit to a lesser extent.

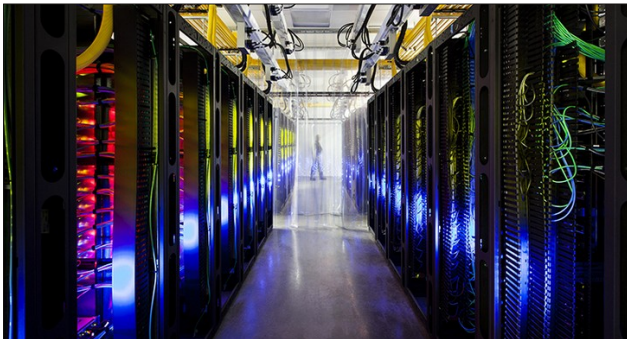
Then, there has been the problem of Google discontinuing services not long after introducing them. They would attract all these users, and then pull the plug. One prime example was Google Wave, where Google tried to reimagine email, using 21st century thinking and technology. Google pulled the plug on it barely 18 months after its launch.

Another discontinued service was Google’s Linux search page. Never mind that Google uses Linux to run its servers. Never mind that the computers/workstations in Google’s headquarters ONLY run Linux (Windows is not allowed). Many Linux users lament the fact that Google makes most – if not all – of its money off of the back of Linux, but refuses to give anything back to the Linux community.

When Google Drive was introduced, Google promised a Linux desktop client for Drive within six months after the launch of Google Drive. Here it is,

well over three years later, and there is still no Linux desktop client available – at least publicly. There are many reports of an “in house” Linux desktop client that Google uses at their offices on their homegrown custom version of Linux (and most of the reports sing praises on how well it works), but they have not released anything to the Linux community. Instead, the Linux community is forced to rely on workarounds and kludge solutions in the absence of an “official” client.

Similarly, if you were a Linux user and a user of Google Picasa, there was little to no “love” flowing from Google to Linux users. For a while, Google distributed the Windows version of the Picasa client, bundled with Wine, for Linux users to use. But then, they discontinued even that kludge. Once again, Linux users are/were forced to use homegrown solutions to find a Linux desktop solution (something Linux users have done and grown accustomed to doing over the years). Linux applications like Shotwell and others allow users to upload their image collections to Picasa. However, the only sure way to view all of your Picasa albums is via Picasa's web interface in your web browser.



A rare look inside one of Google's data centers and network (photo by Google Careers).

Then, there is the matter of Google sharing user information and communications with governments spying on their citizens. Whether the sharing of data is coerced by the governments or not is something that hasn't been fully determined. But, I find it hard to believe that you would not notice a data pipe

running from your servers to NSA servers. Somewhere along the way, someone will certainly take notice. Such activities don't occur without extra equipment (and someone hooking it all up), nor without causing a negative hit to your throughput speeds, and not without causing a CPU/memory hit on the servers.

The tales of abandonment and improprieties go on and on and on, seemingly without an end in sight. They could probably fill a bestselling book, especially when tied to stories from inside 1600 Amphitheatre Pkwy., Mountain View, CA.

Summary

I understand that Google is a business. I understand that their primary income stream is from selling smallish, minimally intrusive text-based ads. I do not lament their desire to tweak and tune their business to maximize their income stream.

What I – and other users – do dislike is how they sometimes go about that pursuit of income. Making users opt out of new services is wrong. Pulling the plug on new services before they've even had the chance to take root is wrong. Not supporting the community that makes the OS that makes your business possible is wrong. Being a participant in illegal spying by governments on their citizens is wrong.

Will I still continue to use Google's robust and helpful services? I'm certain that I will. But, at the same time, I find myself exercising a bit more discretion about the information I share via Google's services, and I'm becoming more and more discriminating by the day. There are other users – some of our friends and family in the PCLinuxOS forums, even – who have sworn off from using Google services altogether.

Given the events in the past 11 years since the Google IPO and in the 20 years since the first

meeting of Larry Page and Sergey Brin, I can only draw one conclusion: “Do No Evil” had a different meaning in 1995 than it has today.



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

PCLinuxOS Magazine's
searchable index!

The **PCLinuxOS** magazine

Does your computer run slow?

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



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

Get your PC back to good health TODAY!

Get



Download your copy today! FREE!

ms_meme's Nook: PCLOS Nights



MP3

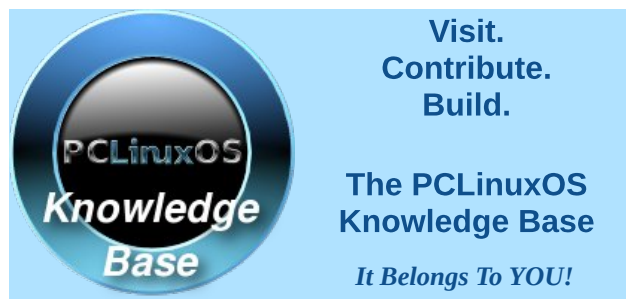
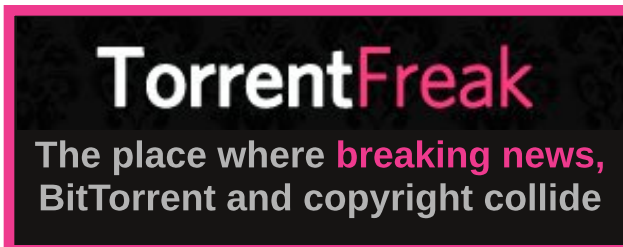
OGG

*PCLOS Nights
Always give you such delights
Free as a breeze
Tex is aiming to please
I'm telling you you'll love them so*

*PCLOS Nights
You will find you'll reach new heights
Download it now
And you will vow
There is no better way*

*Looks so good
So good it's frightening
Boots so fast just like lightening
La da da da da da la da da da da
Da da da da da da*

*You will find
All the others it leaves behind
No more 'puter blues
Let's all spread the news
About PCLOS*



Posted by trytip, on July 15, 2015, running KDE

Make Your Own Streaming Internet Radio Program

by Peter Kelly (critter)

There was a thread in the PCLinuxOS forums recently about listening to internet radio stations. I hadn't seen the thread but Paul Arnote, the magazine editor, followed the thread and was particularly interested in a little one line script posted by forum member dm+. It allowed the user to select and play a station from the impressive list of stations provided by 'Great Little Radio Player', a nice little utility that can be found, free of cost, in the PCLinuxOS repositories.

Paul thought that this would make a good basis for a magazine article. His idea was to expand on this method to provide a tray resident utility that would enable a user to build a list of favourite internet radio stations and then select one for playing but then minimise to the tray when not required. Unfortunately, Paul's busy work schedule, coupled with looking after his young son and producing the magazine, meant that he couldn't really find the time to develop the idea and so he asked me if I would like to tackle it. This is an excerpt from an e-mail he sent me:

"I kind of got the idea from the [radio station thread](#) in the forum that's currently going on. What sparked the idea was a bash command by dm+ (IIRC, it appears on the third page of the thread). He uses Zenity to select which online stream to play. So, I was thinking ... what about a bash script that created a "persistent" window on the desktop, where the user can switch between the streams in their list of "favorites" stations? Plus, when you "minimize" the window, it minimizes to the notification area. Once there, left clicking the mouse gives you the choice of redisplaying the "persistent" window, or changing the station by selecting one of the stations in your favorites list, which is then displayed in the left-click menu. Right clicking with the mouse will give you the choice of redisplaying the "persistent" window, or exiting the program. I'm not sure how problematic this would be, but think it should be "doable" from a bash script. You could get by adequately by omitting the listing of the favorites list in the left-click menu, and just redisplay the "persistent" window. From the redisplay of the window, the user could either select another station, or exit the program. The command to call to play the stream is `mplayer -playlist [URL] ...` and it works very well, at least from a command line prompt."

This article is intended to present a basic script that may be useful to some readers, and to demonstrate the use of some intermediate shell scripting

techniques. I will attempt to explain the code and my reason for including some of the features. The code is not too demanding but some basic scripting experience is presumed.

Defining the script

The first thing that I did was to make a list of features to be included.

- This should be a bash script that all users could run and modify from a standard installation of PCLinuxOS. Editor's Note: It would also be a good idea to already have GLRP (Great Little Radio Player) installed. If it's not installed, you can install it from Synaptic.
- Any tools or programs used by the script should, if not already installed, be available from the PCLinuxOS repositories.
- The script should be a tray resident application with a tooltip and icon.
- Left or right mouse clicking on the icon should be captured and produce some action.
- One of these actions should be to display a menu of available functions.
- Exiting the application should also stop the stream playback.
- The first time the application runs should be detected and the necessary files and directories created and initialized.
- Menu functions should include selecting a favorite station and adding a new favorite from the master list provided by glrp (Great Little Radio Player).
- There should be some form of display to show the name of the currently selected radio stream.

Most of the above list can be accomplished using standard shell scripting techniques, but being tray resident and detecting mouse clicks to provide a menu is not something that I have previously attempted. Being lazy and not wishing to re-invent the wheel, I turned to the PCLinuxOS repositories to search for something that would do some of these things for me. I discovered a nice little

utility called 'alltray'. I re-named the script posted by dm+ to 'radio_play' and ran this line.

First Attempts

```
alltray ./radio_play -m "Edit List:zenity --text-info --editable \
--filename=station_list"
```

While this produced promising results, it wasn't really up to the task at hand, so I looked at the original script once more. The script works just fine, but is not resident in the tray and does not provide menus. Zenity is an excellent set of routines that I have used on many occasions, but in 2009 it was forked and reappeared as yad (yet another dialog), with many new features added. Of particular interest in yad is the notification tool, which promised to solve most of the tray area problems. I decided to use yad.

Starting to write the script

Where to start? I always feel better when I have something written to disk and I always give my scripts their own folder, at least until they are complete. I created a new folder and changed to it.

```
Mkdir ~/radio_streamer; cd ~/radio_streamer
```



The application needs an icon so I searched /usr/share/icons for something suitable, copied it to my new folder and renamed it streamer.png.

Next I opened a text editor and created a file named radio_streamer.sh with the single line of text

```
#!/usr/bin/env bash
```

That line will tell the system to use the bash shell to interpret the code that follows. The contents of my folder now looked like this:

```
ls
radio_streamer.sh streamer.png
```

Now all I have to do is write the code.

A lot of the problems that people have when writing scripts comes from a lack of order in the code. I like to declare any variables right at the start and assign any strings that I might use to some of these variables. This helps to make the final code easier to read.

YAD

The key to this script is the yad notification object, so I'll explain how I implemented it in this script. A yad application is called with the name of the application preceded by a double dash. This is followed by a series of relevant double dash options to control the application. This can lead to a long and unwieldy command and so I use the backslash line continuation character to improve readability. The bash shell treats such text as a single line. There are a lot of possible options that can be applied to a yad notification, but the line for this script looks like this:

```
yad --notification \
--kill-parent \
--listen \
--image="$ICON" \
--text="$HINT" \
--command="bash -c 1_click" <&3 &
```

The first line is the one that actually starts the notification. Next is --kill-parent, which will send a signal to the parent process (bash) when the yad notification exits. The default signal is SIGTERM, so here the bash process which started yad is terminated when yad exits.

Line 3 is the clever bit --listen tells yad to listen on its standard input (stdin) for commands. We haven't yet specified a menu for a right mouse click, which could be done with the option --menu=string, but with this option we can instead send the menu string to stdin, which allows us to dynamically change the menu on the fly. The yad notification commands that can be sent to stdin are: icon, tooltip, visible, action, menu and quit. So we could, for example, change the icon according to conditions.

The icon is set with the option --image=string, and the tool-tip with the option --text=string.

In the last line, we specify what command should be executed when the tray icon is left clicked, to use file descriptor 3 for its stdin and then to run in the background returning control of the script to bash.

Using file descriptors and pipes for redirection

A file descriptor is a pointer to a file or data stream and the available pointers are numbered from 0. Normally stdin, stdout and stderr are assigned to file descriptors 0, 1 and 2. This is how the system knows where to read and write stuff. As we don't want information from other sources going to our notification object, we have to temporarily change things. Our stdin is redirected to FD3, all other process still use FD0 for their stdin.

Make Your Own Streaming Internet Radio Program

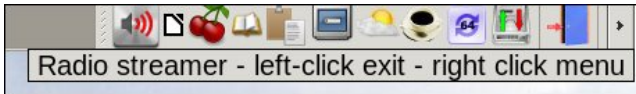
Before we can use this redirection, it needs to be set up, and to do this, we create a special type of file known as a pipe. A pipe is a 'first in first out' object. just like a pipe in real life. What is pushed in first at one end is first to emerge at the other end. These are also known as "fifo"s, and the Linux command to create one is `mkfifo filename`. Once we have our pipe file, we can use standard file redirection to associate FD3 with the pipe

```
mkfifo "$PIPE"
exec 3<> "$PIPE"
```

Declaring the variables

That's the difficult bit out of the way, but let's just back up a bit. We have referred to '*filename*', '*string*' and some variables, so these should be declared before we use them and, as I stated earlier, I like to do this at the start of the script. After the bash header line in our otherwise empty script file add

```
HINT=" Radio streamer - left-click exit - right click menu "
```



This will be the tool-tip defined in the `yad --notification` block above, and is displayed when the mouse pointer hovers over the icon.

The icon to be used in the script is resident in the same folder as the script, and to tell bash where exactly that might be we can use the following line of code

```
$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )/
```

Now you may or may not understand how that works, but don't worry about it. We all use gadgets everyday, such as phones, microwave ovens or whatever, without necessarily understanding how they work. I see no difference in programming. This is a gadget that works to return the directory from which the script was executed, so use it and add it to your toolbox. Actually, if you take it piece by piece, it is not too difficult to understand, but it is difficult to remember, so I copy and paste it from a file of similar little 'time-savers.'

The variable holding the path and the filename of the icon then becomes

```
ICON=$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )/streamer.png
```

Add that line to the script file also.

The string that makes up the menu that is displayed on a right mouse click follows the format

```
title!command|title!command|...
```

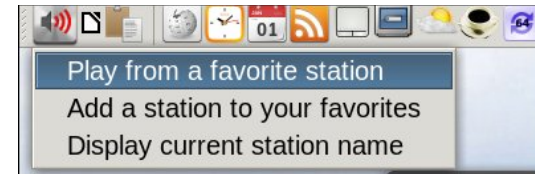
Title is a string that will be displayed on the drop down menu, the ! Signals the end of that string, command is the command that would be executed if that menu item were selected and | separates menu items. Our menu variable definition becomes

```
MENU="Play from a favorite station! bash -c play_favorite|\
Add a station to your favorites! bash -c add_station|\
Display current station name! bash -c show_name"
```

Those strange commands, 'play_favorite', 'add_station' and 'show_name', that we instruct the menu items to execute, will be explained later. The MENU string is echoed to the notification object through its stdin (FD3)

```
echo "menu:$MENU" >&3
```

Obviously this must be done in the script after the tray object has be set up.



For the pipe file, we need to use a filename that is unique, and of course there is a Linux command that will help us to do this called `mktemp`. Again it is of no real consequence to understand exactly how this works so add this line to the script file

```
PIPE=$(mktemp -u /tmp/r_streamer.XXXXXXXX)
```

We need to tell our script where to find the list of stations and this, when 'Great Little Radio Player' is installed, can be found as

```
/home/$USER/.config/glrp/stations.csv
```

Lines like that in the main code of the script do not help readability, and so we assign the string to a variable.

```
STATION_LIST="/home/$USER/.config/glrp/stations.csv"
```


Make Your Own Streaming Internet Radio Program

We also need somewhere to store our own configuration files and our list of favorites. The directory will be:

```
PREFIX="/home/$USER/.config/radio_streamer"
```

And the favorites file:

```
MY_FAVORITES="$PREFIX/favorites"
```

The functions - first_run

When the script starts, it should check if this is the first time that it has been executed, and whether the files and directories in which it will look for its files do actually exist and contain some useful data. If not, then they must be created and some data added.

To achieve this, I wrote the function named first_run. Functions are simply a block of code that is executed, when called, and may, or may not, return a value. The definition of the function must appear before it can be called. The order in which the functions are defined is unimportant, as the code is read into memory and executed as needed, making functions very efficient.

The first_run function is designed to set up the necessary file structure for the rest of the script to access, and can therefore be the first function to be defined. However, after writing the function, I found that it was dependent upon at least one other function in order to operate. This is how I defined the first_run function. The line numbers are not part of the function. They are there for reference only.

```
1.  first_run() {                # set up basic file structure
2.  if ! [ -d "$PREFIX" ]        # does the directory exist?
3.  then                          # if not
4.      mkdir "$PREFIX"          # create it
5.  fi
6.  if ! [ -s "$MY_FAVORITES" ]  # does the file exist and
                                # is not empty
7.  then                          # if not
8.      >"$MY_FAVORITES"         # create the empty file
9.  else                          # if it does exist
10.     exit 0                    # This is not the first run
11. fi
12.                             # explain what is happening with a dialog
13. yad --title "Radio Streamer - First Run" \
14.     --width=550 \
15.     --text="As this is the first time that you have
```

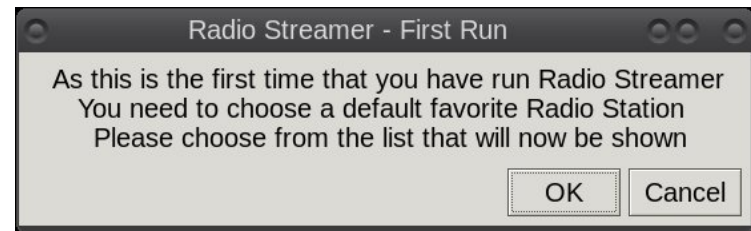
```
16.     run Radio Streamer\n \
17.     You need to choose a default favorite Radio Station\n \
18.     Please choose from the list that will now be shown" \
19.     --button="OK:1" \
20.     --button="Cancel:2"
21.     if [[ $? == "1" ]]        # the OK button was pressed
22. then                          # call the add_station function
23.     add_station                # add a station to the favorites file
24.     killall mplayer           # stop other mplayer activity
                                # and play it
25.     mplayer -nolirc -msglevel all=0 "$s_url" &
26.     echo $s_url > $PREFIX/current_url # save the url
                                # for the next run
27.     exit 0                    # leave the first_run function
28. else
29.     exit 1                    # user decided to cancel
                                # so return non zero
30. fi
31. }                            # end of first_run function
```

The comments should explain most of the workings of the function. The inclusion of comments make later modification of the code much easier.

Lines 1-5 look for the existence of the directory, and create it if necessary.

Lines 6-11 look for the favorites file. If that exists, then there is no need to continue. If not, then it is created and the function continues.

Lines 13-20 display an explanatory yad dialog. Pressing the OK button causes yad to return a value of 1, while pressing cancel returns 2.



Lines 21-30 check the value returned by the dialog in the bash variable \$?, which always contains the exit code of the last executed command. When control returns from the first_run function to the main script, the value returned by the exit statement can be checked, and if the user has pressed the cancel button, then the entire script can be abandoned. If the user pressed OK, then the script is allowed to continue.

Make Your Own Streaming Internet Radio Program

```
# is this the first run of the script?
if ! (first_run)
then
    exit # the user cancelled
fi
```

This is the code that actually executes the `first_run` function and takes the appropriate action. All other functions are executed by interaction with the tray object.

If OK was pressed, then the command `add_station` is executed. As there is no standard command of that name, then we have to create one. That is done by defining a function with that name. Having added a station to our list of favorites, we first stop any activity that mplayer is currently engaged in, play our selected stream, and save the url to the file `current_url` to be used on next startup. If the cancel button was pressed, then the function exits and the function returns a value of 1. It is important to be clear about where a returned value is from.

The `add_station` function

Now we need to define the `add_station` function which is simply

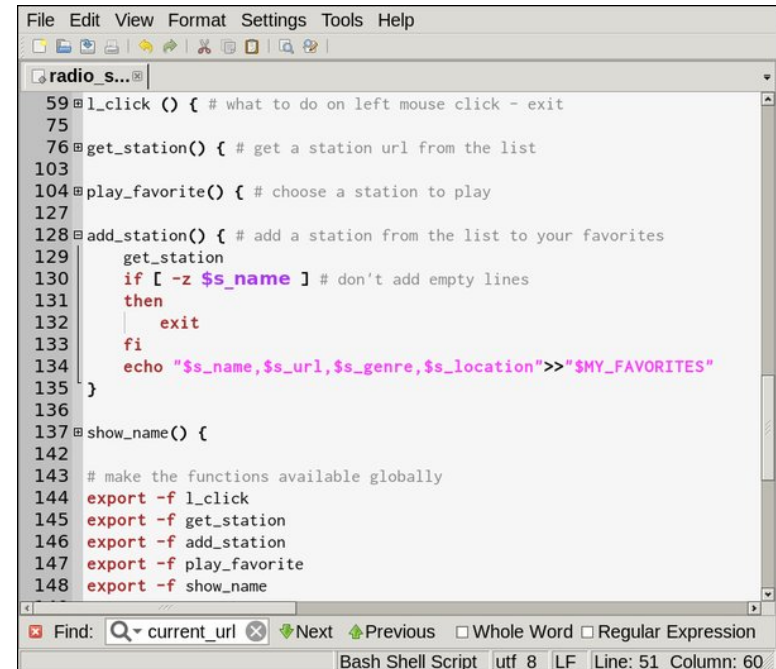
```
add_station() { # add a station from the list to your favorites
get_station
if [ -z $s_name ] # don't add empty lines
then
    exit
fi
echo "$s_name,$s_url,$s_genre,$s_location">>"$MY_FAVORITES"
}
```

The function `add_station` calls another function named `get_station`, exits if the user pressed enter without selecting a station and then it echoes the values of some variables, separated by commas, to the end of our favorites file. We haven't seen those new variables before because they were created and assigned to in the `get_station` function, which we haven't yet defined. If you look back to when we defined the MENU string you will see that `add_station` is one of the commands we requested to be executed.

Which came first?

The `first_run` function calls the function `add_station`, and `add_station` calls another function named `get_station`. This chicken and egg situation of which function to create first and of calling a function that has not yet been defined can be confusing, and so a little thought is required to determine the structure of the

script. The method that I employ is to use an editor that supports 'code folding.' There are many such editors available, from the notorious vim to graphical programmers editors such as scite and KDE's own Kwrite. Code folding allows you to write a function header followed by an explanatory comment, and only add the code once you know what you are going to put in there. You can fill the function body with more comments as you think of them, and hide away all but the header when you don't need to see it. This makes it much easier to see the structure of the script.



```
File Edit View Format Settings Tools Help
radio_s...
59 l_click() { # what to do on left mouse click - exit
75
76 get_station() { # get a station url from the list
103
104 play_favorite() { # choose a station to play
127
128 add_station() { # add a station from the list to your favorites
129     get_station
130     if [ -z $s_name ] # don't add empty lines
131     then
132         exit
133     fi
134     echo "$s_name,$s_url,$s_genre,$s_location">>"$MY_FAVORITES"
135 }
136
137 show_name() {
142
143 # make the functions available globally
144 export -f l_click
145 export -f get_station
146 export -f add_station
147 export -f play_favorite
148 export -f show_name
Find: current_url Next Previous Whole Word Regular Expression
 Bash Shell Script utf 8 LF Line: 51 Column: 60
```

The screenshot above is shown using an editor called Editra (it is in the PCLinuxOS repositories). Clicking on the plus and minus signs expands or collapses the code.

Exporting

Also in the screen shot, lines 123-126 export the functions. When a function is called, it is executed in a sub-shell which does not inherit the parent shells environment. Exporting functions and variables makes them available for use in sub-shells. Functions must be exported with `-f` option. By doing, this the function `add_station` can call the function `get_station`, which will eventually be defined in the parent shell. If the function `get_station` exports the variables `s_name`, `s_url`, etc., they will be available to the function `add_station`.

We need to export some of our variables from the main script to make them available to our functions. Add the following between the variable declarations and the function code.

```
# Make some variables available globally
export STATION_LIST
export MY_FAVORITES
export PIPE
```

Note that the exported variables are only copies of the originals. If the function `add_station` alters one of the inherited variables, the original variable in the function `get_station` is unchanged.

The `get_station` function

At first glance, this looks to be a complex function but most of the code is there for setting up a yad multicolumn list object.

```
1.  get_station() { # get a station url from the list
2.  chosen=$(cat "$STATION_LIST" | \
3.  awk -F, '{print $1"\n"$2"\n"$3"\n"$4}' | \
4.  sed 's/"/"/g' | \
5.  yad --list \
6.  --geometry=800x800 \
7.  --title='Internet Radio Stations' \
8.  --column Station_Name \
9.  --column Station_URL \
10. --column Station_Genre \
11. --column Station_Location \
12. --hide-column=2 \
13. --no-markup \
14. --ellipsize=END \
15. --expand-column=1 \
```

```
16. --print-column=0 | \
17. sed 's/|/,/g')
18. s_name=$(echo "$chosen" | awk -F, '{print $1}')
19. s_url=$(echo "$chosen" | awk -F, '{print $2}')
20. s_genre=$(echo "$chosen" | awk -F, '{print $3}')
21. s_location=$(echo "$chosen" | awk -F, '{print $4}')
22. export s_name
23. export s_url
24. export s_genre
25. export s_location
26. }
```

The list of stations in the file pointed to by the variable `STATION_LIST` is in the form of fields that are both double quoted and comma separated. Each line is of the form

`"Name", "URL", "Genre", "Location", "Favorite"`

The yad list object expects the column data to be unquoted, to be separated by newline characters, and to be supplied without the quotes. Each record of fields must be separated by a vertical bar character `|`. Lines 2-4 perform the conversion.

In line 2, the stations file is piped to the next command in line 3.

Line 3 uses `awk`, setting the field separator to be a comma. `Awk` then prints out the first four fields in each line of the file, separating each of them with a newline character, and ending the output with a vertical bar character. The fifth field is not used by us, and is therefore simply discarded.

In line 4, the `sed` utility substitutes the double quote with nothing `s/"/`. The final `"g"` means globally, so replace every occurrence in the line. The effect of this is to remove all of the quotes.

Now that we have the data in the form that yad can accept it, we can pipe it to `yad --list` in line 5.

Make Your Own Streaming Internet Radio Program

Line 6 sets the dimensions of the list dialog

Line 7 sets the text that will appear in the title bar of the dialog window

Lines 8-11 set the titles for the columns

Line 12 tells yad not to display the URL column. Although we do need it, we do not need to see it.

Yad will try to use 'pango markup' to display the text. This is useful if you want colored or bold text. Unfortunately, our import file may contain characters such as '&,' which it would attempt to interpret as markup symbols. To prevent this, we turn off the mark up feature in line 13.

Some of the data may be too long to fit in the column. The `--ellipsize` option allows us to show continuation as an ellipsis (...), and also to set the position of the ellipsis. I chose to set it at the end of the field in line 14 to ensure the start of the field is always displayed.

The column that is of most interest will probably be the station's name, and in line 15, this column is expanded to show as much of the name as possible. This is a compromise, as to some extent, the width of the columns is determined by the length of the column name and the geometry of the dialog. Expanding this column ensures that any 'spare' space is directed here.



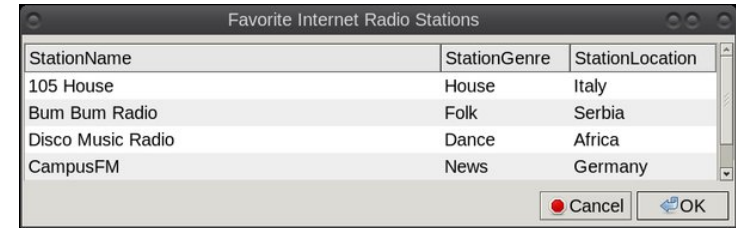
StationName	StationGenre	StationLocation
105 House	House	Italy
Accent 4	Classic	France
Aktina Radio	Top 40	Greece
Antena Radio	Folk	Serbia
Antena Zagreb	Top 40	Croatia
Athens DeeJay 95.2	Top 40	Greece
Atlantis FM	Rock	Greece
B92 uzivo	News	Serbia
Big FM	Top 40	Romania
BN Radio	News	Bosna i Herceg...
Bum Bum Radio	Folk	Serbia
Cadena Cope Sevilla	News	Spain
CampusFM	News	Germany
Candil Radio	Regional ser...	Spain
Chanquete FM	Pop	Spain
Classical 102	Classic	Germany
Coast FM	Top 40	Spain
Croozee.fm	Jazz	Belgium
Dalkas 88.2	Etno	Greece
Disco Music Radio	Dance	Africa
esRadio	News	Spain
Finam FM	Classic Hits	Russia
Flash FM	Dance	France
FM4 - Live from Austria - Vienna	Hit	Austria

Line 16 tells yad which columns of the selected data line to output. A zero here means all columns, including the URL column that we decided not to display. This ends the list dialog definition, and the output is then piped to the sed command in line 17. This sed command replaces the vertical bar characters supplied by yad with commas, which are easier to work with in the next awk statements. Lines 18-21 each assign one output field to a variable and lines 23-26 export the variables.

The play_favorite function

This function is very similar to the get_station function.

```
1. play_favorite() { # choose a station to play
2. current=$(cat "$MY_FAVORITES" | \
3.     awk '!x[$0]++' | \
4.     awk -F, '{print $1"\n"$2"\n"$3"\n"$4}' | \
5.     yad --list \
6.     --geometry=800x800 \
7.     --title='Favorite Internet Radio Stations' \
8.     --column Station_Name \
9.     --column Station_URL \
10.    --column Station_Genre \
11.    --column Station_Location \
12.    --hide-column=2 \
13.    --no-markup \
14.    --ellipsize=END \
15.    --expand-column=1 \
16.    --print-column=2 | \
17.    sed 's/|/$/'
18. )
19. echo $current > $PREFIX/current_url # save it for next run
20. killall mplayer
21. mplayer -nolirc -msglevel all=0 "$current" &
22. show_name
24. }
```



StationName	StationGenre	StationLocation
105 House	House	Italy
Bum Bum Radio	Folk	Serbia
Disco Music Radio	Dance	Africa
CampusFM	News	Germany

This is the other function called from the right click menu. The file piped to the list dialog is now our favorites file. The title has changed and mplayer is called with the selected station URL. The two options passed to mplayer remove remote control access and turn off almost all text output, as neither of these features are of any use to us.

The awk command in line 3 is interesting. I'm not going to explain it, and it is another of my little 'time-savers.' It removes duplicate lines. The favorites file may well contain duplicates. That isn't a problem in itself, but we don't need to display the duplicates here. If you want to clean up the file, use a command such as (next page)

Make Your Own Streaming Internet Radio Program

```
uniq -u <(cat favorites) > new_favorites ; mv -f new_favorites favorites
```

This line uses process substitution to filter out duplicate lines from the contents of the original and write the filtered data to a new file. The old file is then overwritten by the new file. Make a backup first!

Line 17 removes the final vertical bar character from the output produced by yad. Line 19 saves the newly selected url to the file current_url for future use. Lines 20 & 21 produce the new output, and line 22 calls the function show_name, which we haven't yet written.

The show_name function

This function is called by the play_station function, by the right click menu, and at the very beginning of the scripts operation, when the last played station is recalled from the file current_url.

For this function, I decided to use a little pop-up notification utility named notify-send, which pop up a little box in the corner of the screen for a few seconds, and then gets out of the way.

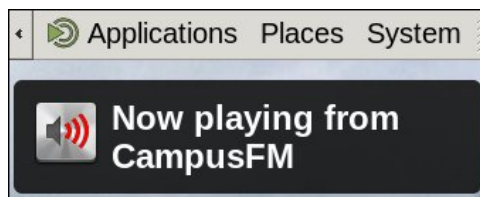
Notify-send takes the following options in this function:

```
-t      a time period in milliseconds to remain visible.
-i      in icon filename to display.
```

And a string of text to display.

```
show_name() {
current_name=$(grep $(cat $PREFIX/current_url) $PREFIX/favorites |
\
awk -F, '{print $1}')
notify-send -t 3000 -i $ICON "Now playing from $current_name"
}
```

To get the station name of the currently playing URL, we use the grep command to search for it in our favorites file, and then use awk to isolate the station name and put it in the variable current_name. The notify-send command uses \$ICON for the icon to display, a time of three seconds (3,000 milliseconds) to remain visible, and displays some text constructed from some literal text and the value held by the variable current_name.



The l_click function

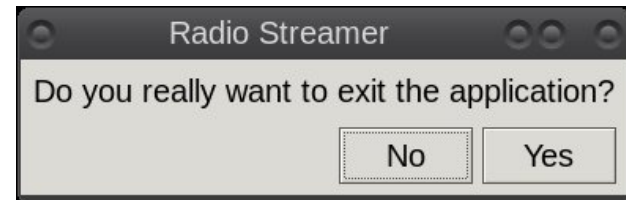
Left clicking on the objects icon gives us a means to exit and to terminate mplayer. This is too easy to do unintentionally and so we need to confirm that this is what the user really wants.

```
l_click () { # what to do on left mouse click - exit
exec 3<> "$PIPE"      # make the redirection available
                    within the function

yad --title "Radio Streamer" \
--text="Do you really want to exit the application?" \
--button="No:1" \
--button="Yes:2"

    if [[ $? = "1" ]]
    then

exit
    else
echo "quit" >&3      # send yad the quit command
killall mplayer
rm -f "$PIPE"      # remove the fifo
fi
}
```



The function is referenced in the notification setup block by the --command option. When called, the function displays a confirmation dialog, and if the user pressed No, then the function is exited and the script continues. If Yes, is selected then the 'quit' command is sent to yad via FD3, the mplayer process is terminated, and the pipe file is deleted. The script then ends.

The script structure

To recap, the basic structure of the script should look like this:

```
bash header
variable declarations
variable exports
function definitions
function exports
```

Make Your Own Streaming Internet Radio Program

call first_run function
create the pipe file
redirect data through the pipe mechanism
set up the tray mechanism and background it
echo the menu string to the tray object.

If you use an editor that offers code folding you should be able to see this structure with very little scrolling.

Bugs!

Every decent program has bugs, at least that's my experience. The first one that I noticed in this script was that on changing stations the previous output continued along with the new output. This was solved by adding the line

```
killall mplayer
```

before the line that starts playing the new stream in the play_favorite function.

The next problem was that some stations would not play even though the same station would play when called directly from the command line. This was caused by yad adding a vertical bar character to the end of the url that got passed to mplayer. A simple sed statement filters this out. The end of the play_favorite function now looks like this.

```
...
...
    --ellipsize=END \
    --expand-column=1 \
    --print-column=2 | \
    sed 's/|$//'
)
killall mplayer
mplayer -nolirc -msglevel all=0 "$current" &
}
```

More bugs will undoubtedly appear, and fixing them may introduce other, unforeseen problems. This the joy of programming.

Adding features

Mplayer is designed to play movies but is also very capable when streaming audio, as we have done here. There are many more features documented in the man pages that you may like to add. For example, mplayer is capable of

capturing the audio stream and writing it to a file. This may be useful if something that you want to listen to is being broadcast at an inconvenient time. To save the stream to a file named stream.mp3 use a line like this:

```
mplayer -dumpaudio -dumpfile stream.mp3 stream_url
```

Adding a real URL in place of stream.url. This could be launched and terminated by a tool such as cron or at. *There may be legal considerations in your locale for storing unlicensed digital works.*

The complete script

Below is the complete script. You can also download it from the magazine website, [here](#).

***Editor's Note:** It wouldn't be too much work to make this script even more "feature rich." You could add an ability to manually enter new stations, instead of relying on only those that are packaged with GLRP. You could also add in the ability to edit the "Favorites stations" list (as it is now, you will need to launch the ~/.config/radio_streamer/favorites list in a text editor to delete the stations you no longer wish to listen to from your favorites list). For these and other enhancements, we will leave as a learning exercise, should you choose to accept the challenge.*

```
#!/usr/bin/env bash
```

```
# Initialise the strings
MENU="Play from a favorite station! bash -c play_favorite|\
Add a station to your favorites! bash -c add_station|\
Display current station name! bash -c show_name"
HINT=" Radio streamer - left-click exit - right click menu "
PIPE=$(mktemp -u /tmp/r_streamer.XXXXXXX)
ICON=$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd
)/streamer.png
PREFIX="/home/$USER/.config/radio_streamer"
STATION_LIST="/home/$USER/.config/glrx/stations.csv"
MY_FAVORITES="$PREFIX/favorites"
```

```
# Make some variables available globally
export STATION_LIST
export MY_FAVORITES
export PIPE
export PREFIX
export ICON
```



```
# define functions
first_run() { # set up basic file structure
# check for and create if necessary
# directory ~/.config/radio-streamer
if ! [ -d "$PREFIX" ]
then
    mkdir "$PREFIX" # create the directory
fi

# check for and create if necessary
# the file favorites
if ! [ -s "$MY_FAVORITES" ]
then
    >"$MY_FAVORITES" # create the empty file
else
    exit 0 # This is not the first run
fi

yad --title "Radio Streamer - First Run" \
    --width=550 \
    --text=" As this is the first time that you have run Radio
Streamer\n \
    You need to choose a default favorite Radio Station\n \
    Please choose from the list that will now be shown" \
    --button="OK:1" \
    --button="Cancel:2"
if [[ $? == "1" ]]
then
    add_station # add a station to the favorites file
    killall mplayer # play it
    mplayer -nolirc -msglevel all=0 "$s_url" &
    echo $s_url > $PREFIX/current_url # save it for next run
#    show_name
    exit 0
else
    exit 1 # user decided to cancel
fi
}

l_click () { # what to do on left mouse click - exit
    exec 3<> "$PIPE" # make the redirection available within
the function
    yad --title "Radio Streamer" \
        --text=" Do you really want to exit the application? " \
        --button="No:1" \
        --button="Yes:2"

    if [[ $? = "1" ]]
    then
        exit
    fi
}
```

```
else
    echo "quit" >&3 # send yad the quit command
    killall mplayer
    rm -f "$PIPE" # remove the fifo
fi
}
```

```
get_station() { # get a station url from the list
chosen=$(cat "$STATION_LIST" | \
    awk -F, '{print $1"\n"$2"\n"$3"\n"$4}' | \
    sed 's/"/\\g' | \
    yad --list \
        --geometry=800x800 \
        --title='Internet Radio Stations' \
        --column Station_Name \
        --column Station_URL \
        --column Station_Genre \
        --column Station_Location \
        --hide-column=2 \
        --no-markup \
        --ellipsize=END \
        --expand-column=1 \
        --print-column=0 | \
    sed 's/|/,/g')
s_name=$(echo "$chosen" | awk -F, '{print $1}')
s_url=$(echo "$chosen" | awk -F, '{print $2}')
s_genre=$(echo "$chosen" | awk -F, '{print $3}')
s_location=$(echo "$chosen" | awk -F, '{print $4}')
```

```
export s_name
export s_url
export s_genre
export s_location
}
```

```
play_favorite() { # choose a station to play
current=$(cat "$MY_FAVORITES" | \
    awk '!x[$0]++' | \
    awk -F, '{print $1"\n"$2"\n"$3"\n"$4}' | \
    yad --list \
        --geometry=800x800 \
        --title='Favorite Internet Radio Stations' \
        --column Station_Name \
        --column Station_URL \
        --column Station_Genre \
        --column Station_Location \
        --hide-column=2 \
        --no-markup \
        --ellipsize=END \
        --expand-column=1 \
```

Make Your Own Streaming Internet Radio Program

```
--print-column=2 |\  
sed 's/|$//'  
)  
echo $current > $PREFIX/current_url # save it for next run  
killall mplayer  
mplayer -nolirc -msglevel all=0 "$current" &  
show_name  
}  
  
add_station() { # add a station from the list to your favorites  
    get_station  
    if [ -z $s_name ] # don't add empty lines  
    then  
        exit  
    fi  
    echo "$s_name,$s_url,$s_genre,$s_location">>"$MY_FAVORITES"  
}  
  
show_name() {  
    current_name=$(grep $(cat $PREFIX/current_url) $PREFIX/favorites |\  
    \    awk -F, '{print $1}')    notify-send -t 3000 -i $ICON "Now playing from $current_name"  
}  
  
# make the functions available globally  
export -f l_click  
export -f get_station  
export -f add_station  
export -f play_favorite  
export -f show_name  
  
# is this the first run of the script?  
if ! (first_run)  
then  
    exit # the user cancelled  
fi  
  
# set up the pipe mechanism  
mkfifo "$PIPE"  
exec 3<> "$PIPE"  
  
# set up the tray object  
yad --notification \  
    --kill-parent \  
    --listen \  
        --image="$ICON" \  
        --text="$SHINT" \  
        --command="bash -c l_click" <&3 &
```

```
# send the menu string to the tray object  
echo "menu:$MENU" >&3  
mplayer -nolirc -msglevel all=0 $(cat $PREFIX/current_url) &  
show_name
```



The PCLinuxOS
Magazine

Created with
Scribus



PCLinuxOS Full Monty ...

Everything you might want or need –
plus the kitchen sink!

Tip Top Tips: Brother Printer Driver Installer

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

This month's [tip](#) comes from PCLinuxOS forum member The Chief.

Brother continues to impress me.

They have a new *Linux-brprinter-installer*, and instructions for its use follow. By the way, it works like a champ! I used it to re-install the drivers for my two Brother printers (both network printers) and it magically cured a nagging *slow start* problem I was having with one of the printers.



Brother MFC-J870DW Wireless Color Inkjet All-In-One with Scanner, Copier and Fax Printer, available on [Amazon.com](#) for \$99.99 (U.S.)

You can choose either RPM or DEB systems when you look for the driver downloads, but it will lead you to this installer. Try it, you'll like it!

Editor's Note: You can search for your printer model [here](#), since some of the printers use a different driver. This will help insure you download the appropriate driver for your printer. Different drivers may have different procedures for installation, so be sure to check the instructions for the driver you install.

The instructions for running the installer have been copied from the Brother web site. The Chief annotated the instructions, and his annotations appear in red text.

Step 1. If your printer is listed in the list of printers at the end of this article, you can go ahead and [download](#) the tool (`linux-brprinter-installer-*.*.*.gz`). These instructions are for this particular driver and installation tool. The tool will be downloaded into the default "Download" directory. The directory location varies depending on your Linux distribution (e.g. `/home/YourUsername/Download`).

Step 2. Open a terminal window and go to the directory you downloaded the file to in the last step. [\[Right Click/Actions/Open Terminal Here made that a lot simpler!\]](#)

Step 3. Enter this command to extract the downloaded file:

```
gunzip linux-brprinter-installer-*.*.*.gz
```

Step 4. Get superuser authorization with the "su" command.

Step 5. Run the tool, using this command: `bash linux-brprinter-installer-*.*.*.gz Brother machine name.`

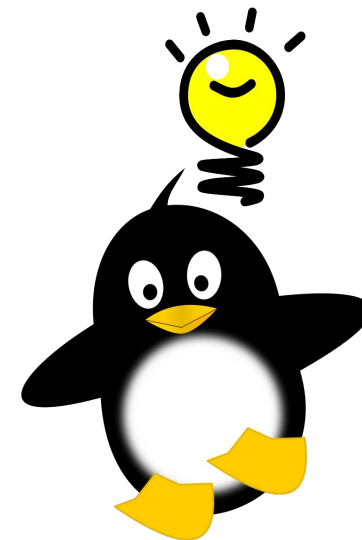
[\[Substitute your printer model for "Brother machine name," HL-2270DW, for example.\]](#)

Step 6. The driver installation will start. Follow the installation screen directions. When you see the message "Will you specify the DeviceURI?" For USB Users: Choose N(No). For Network Users: Choose Y(Yes) and DeviceURI. [\[Select the Device URI from a list.\]](#)

The install process may take some time. Please wait until it is complete. It will present you with a list of URIs, several for each networked printer. I selected the URI with the model number and "`_pdl-datastream._tcp.local/`" as shown in the example below:

```
dnssd://Brother%20HL-2270DW%20series._pdl-datastream._tcp.local/
```

You choose by selecting an entry from the list by number. You may want to try other possibilities, this worked for me.



Guys, Linux printer installation just got a whole lot easier! At least if you're smart enough to buy a Brother printer. I searched for a particular printer asked about by another poster, but it should be the same for all.

Here are the printers covered:

DCP-110C, DCP-120C, DCP-130C, DCP-1400, DCP-165C, DCP-330C, DCP-350C, DCP-375CW, DCP-385C, DCP-395CN, DCP-585CW, DCP-7020, DCP-7030, DCP-7040, DCP-7060D, DCP-7065DN, DCP-8020, DCP-8025D, DCP-8040, DCP-8045D, DCP-8060, DCP-8065DN, DCP-8080DN, DCP-8085DN, DCP-8110DN, DCP-8150DN, DCP-8155DN, DCP-9040CN, DCP-9045CDN, DCP-J125, DCP-J140W, FAX-1820C, FAX-1840C, FAX-1860C, FAX-1920CN, FAX-1940CN, FAX-1960C, FAX-2440C, FAX-2480C, FAX-2580C, FAX-2820, FAX-2840, FAX-2900, FAX-2920, FAX-2940, FAX-3800, FAX-4100/FAX-4100e, FAX-4750e, FAX-5750e, HL-1230, HL-1240, HL-1250, HL-1270N, HL-1435, HL-1440, HL-1450, HL-1470N, HL-1650, HL-1670N, HL-1850, HL-1870N, HL-2040, HL-2070N, HL-2140, HL-2170W, HL-2220, HL-2230, HL-2240, HL-2240D, HL-2270DW, HL-2275DW, HL-2280DW, HL-2460, HL-2600CN, HL-2700CN, HL-3040CN, HL-3045CN, HL-3070CW, HL-3075CW, HL-3140CW, HL-3170CDW, HL-3450CN, HL-4150CDN, HL-4570CDW, HL-4570CDWT, HL-5030, HL-5040, HL-5050, HL-5070N, HL-5140, HL-5150D, HL-5170DN, HL-5240, HL-5250DN, HL-5280DW, HL-5340D, HL-5350DN, HL-5370DW/HL-5370DWT, HL-5440D, HL-5450DN, HL-5470DW, HL-5470DWT, HL-6050D, HL-6050DN, HL-6180DW, HL-6180DWT, HL-7050, HL-7050N, HL-8050N, HL-S7000DN, MFC-210C, MFC-230C, MFC-240C, MFC-250C, MFC-255CW, MFC-290C, MFC-295CN, MFC-3220C, MFC-3240C, MFC-3320CN, MFC-3340CN, MFC-3360C, MFC-3420C, MFC-3820CN, MFC-420CN, MFC-440CN, MFC-465CN, MFC-4800, MFC-490CW, MFC-495CW, MFC-5440CN, MFC-5460CN, MFC-5490CN, MFC-5840CN, MFC-5860CN, MFC-5890CN, MFC-5895CW, MFC-620CN, MFC-640CW, MFC-6490CW, MFC-665CW, MFC-6800, MFC-

685CW, MFC-6890CDW, MFC-7220, MFC-7225N, MFC-7240, MFC-7340, MFC-7345N, MFC-7360N, MFC-7420, MFC-7440N, MFC-7460DN, MFC-7820N, MFC-7840W, MFC-7860DW, MFC-790CW, MFC-795CW, MFC-820CW, MFC-8220, MFC-8420, MFC-8440, MFC-845CW, MFC-8460N, MFC-8480DN, MFC-8500, MFC-8510DN, MFC-8640D, MFC-8660DN, MFC-8670DN, MFC-8680DN, MFC-8690DW, MFC-8710DW, MFC-8810DW, MFC-8820D, MFC-8840D, MFC-8840DN, MFC-885CW, MFC-8860DN, MFC-8870DW, MFC-8890DW, MFC-8910DW, MFC-8950DW, MFC-8950DWT, MFC-9010CN, MFC-9120CN, MFC-9125CN, MFC-9130CW, MFC-9320CW, MFC-9325CW, MFC-9330CDW, MFC-9340CDW, MFC-9420CN, MFC-9440CN, MFC-9450CDN, MFC-9460CDN, MFC-9560CDW, MFC-9700, MFC-9800, MFC-9840CDW, MFC-990CW, MFC-9970CDW, MFC-J220, MFC-J265W, MFC-J270W, MFC-J280W, MFC-J285DW, MFC-J410W, MFC-J415W, MFC-J425W, MFC-J430W, MFC-J435W, MFC-J4410DW, MFC-J450DW, MFC-J4510DW, MFC-J4610DW, MFC-J470DW, MFC-J4710DW, MFC-J475DW, MFC-J5910DW, MFC-J615W, MFC-J625DW, MFC-J630W, MFC-J650DW, MFC-J6510DW, MFC-J6710DW, MFC-J6910DW, MFC-J825DW, MFC-J835DW, MFC-J870DW, MFC-J875DW

Summary

Brother printers are popular with Linux users. Brother makes a quality printer, and provides excellent support for Linux. Furthermore, supplies for Brother printers (toner cartridges and ink cartridges) are some of the most affordable and reasonably priced of all printer manufacturers. For example, for the printer pictured in this article, the black ink cartridge is priced under \$20 (U.S.) and the three-pack of color inks is priced under \$32 (U.S.). There are several reports in the PCLinuxOS forum of everything working on the Brother All-In-One printers (printer, scanner, etc.). So, if you're looking for a new

printer, you might just want to take a look at what Brother has to offer.





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

If you have article ideas, or if you would like to contribute articles to the PCLinuxOS Magazine, send an email to:
pclinuxos.mag@gmail.com

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

Disclaimer

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

Material Submitted by Users

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

Entire Agreement

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



Screenshot Showcase



Posted by Yankee, on July 16, 2015, running openbox

Game Zone: DungeonRift

by daiashi



About The Game

Join Sir Bucket and his friends in their peaceful journey through DungeonRift! Kill everybody on the level, grab the loot and move to the portal. But what's that? Monsters are following you! They're becoming stronger and smarter! The dungeon itself transforms into a darker, creepier and more dangerous place! How long will Sir Bucket survive?

Monsters level instead of the player. Each enemy in DungeonRift has its own perks and abilities that they learn randomly, creating surprising and dangerous combinations. Every run won't be the same - different monsters choosing random perks will force Sir Bucket to adapt his strategy every time.

Destroy your foe, explore the Rift, unlock new weapons and meet new rivals in your endless struggle in DungeonRift!

Main features already developed:

- Fast and dynamic action

- Various abilities and perks for monsters

- 150+ of dungeon layouts

- Menacing traps

- Weapons with different mechanics

- Player progression system

- Two playable characters

- Dark humor

- Funny collectibles

- Sweet and tasty hand-drawn art

- Dynamic music by John Leonard French

In the past...

Dungeon Rift was born as concept-project for a contest and rapidly became an obsession. They were very happy to see people playing their alpha-version, and that's why they've decided to turn their last free version into a complete and polished product. That's what they've called the demo version - you can download it right now and get the feel of DungeonRift. Most of the aforementioned features are there. You could decide for yourself, to support our project or not, carefree.

In the future...

They're planning to fill Dungeon Rift with content and new features in upcoming months:

Content-wise - updates with weapons, characters, monsters, perks and locations planned to be frequent and regular.

Feature-wise - they are very excited to add local co-op to their game, and to make a monsters' proficiency-system, which will make them choose one of the two evolutions paths in mid-game, therefore making the gameplay even more unpredictable and interesting.

There is much much more they've planned in the future, but they don't want you to bet on their promises. Check the demo and join the early access, if you are interested in where it's going.

This is a fun game to play if you're into the whole cartoon-style genre. It reminds me a little of Gauntlet. Accept your shield and a sword. Hack and slash with its share of humour and hidden traps. One thing I would like to add is that enemies actually level and receive new weapons.

System requirements:

Fully updated PCLinuxOS and Steam

Hardware:

Recommended:

OS: PCLinuxOS

Processor: 1.4Ghz+

Memory: 512 MB RAM

Graphics: 256Mb, pixel shaders 2.0

Hard Drive: 1200 MB available space

About The Company

I didn't find much except that they are a group of Russian game developers.

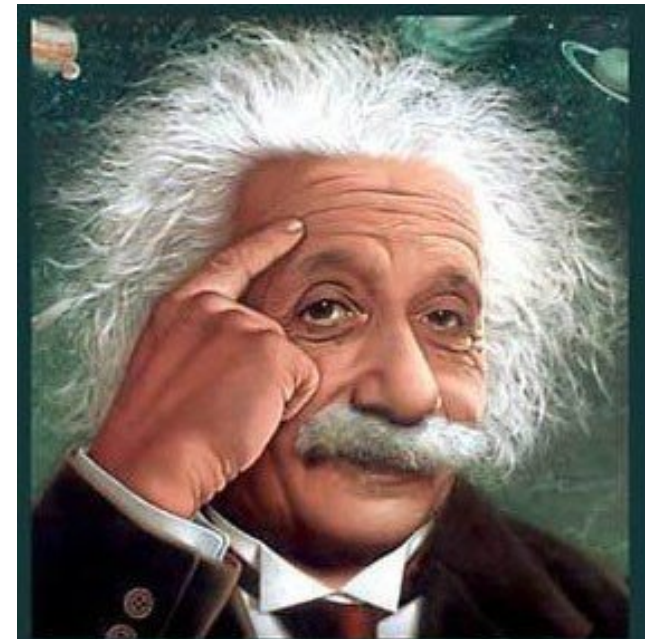
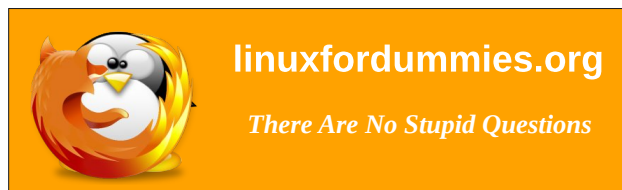
Some Gameplay Screenshots



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 it by name. Click on and download the demo. If you have updated your system, including graphics drivers, you should be good to go.

<http://store.steampowered.com/app/375550/>



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



PCLinuxOS Family Member Spotlight: BobK54

As told to Smileeb

Hi smileeb! OK, I've been shamed into volunteering! I've been hanging around here for about 7+ years and I too enjoy learning a little about others on the forum. "You don't have to be a prolific poster." Yeah, that's me. What can I say? It just works! I'll volunteer to be a victim. Kind of a daily-user type with a mostly semi-computer background. Computers have always been tools. I think if I didn't go into aviation engineering I might have gone into IT though. How about this for using tools: years ago I had a turbine engine performance algorithm supplied to me by my company on an HP41C (in RPN). I then converted that to BASIC on an AppleII and then rewrote it in Fortran on PUNCHCARDS for a missile customer. Let me know if you're interested. Thanks!!

Here is BobK54



How old are you?

At the end of March I was 61 years young.

Married, single or what?

I have been happily married to my wife for 38 years.

Children, grandchildren?

We have four children, all boys...or adult men now!

Our boys have blessed us with eight grandchildren and an additional three came to us with a daughter-in-law from a prior marriage. The grandchildren are eight boys and three girls and range in age from 18 months old to almost 16 years old. At one point, we had eight grandchildren under six years old! Yikes.

Retired or working and for how long and at what.

I'm working full time in the aerospace field for 32 years, with an eight year automotive electrical motor/actuator stint in between.

Today I'm an applications aerospace engineer/sales manager providing turbine engine oil debris monitoring systems, fluid accessories, and pressure sensors to aircraft engine and gearbox makers, as well as direct to the airframe makers like Boeing and Airbus. Our systems provide early warning of wear in engines and gearboxes, allowing scheduled maintenance instead of those annoying in-flight engine shutdowns. Our parts can be found on virtually every new aircraft flying, as well as some missiles and spacecraft, including the now retired Space Shuttle.

Helicopters are a big customer focus. They have very big, complex transmissions with lots of bearings and gears, which make all kinds of metal shaving debris which needs to be collected and sensed.

By far, the most interesting part of my career was for the several years that I worked with Burt Rutan, who designed the Voyager. Round-the-World Non-stop/Non-refueled aircraft; the Spaceship One/Whiteknight One X-Prize aircraft/spacecraft; and the Virgin Galactic SpaceshipTwo/WhiteKnight Two aircraft/spacecraft. During the time I worked with him we built the NASA AD-1 Oblique Wing test aircraft (scissor wing), the PARLC jet powered scale

model ship for the Navy, as well as the Fairchild NGT 62% scale model aircraft, which actually helped Fairchild win the T-46A trainer contract in the early 80's. I wrote the Feasibility Study for the Beech Starship in the mid-80's, an all-composite canard wing biz aircraft which eventually died an ugly death when metal-minded structures committees and the FAA couldn't cope with this new-fangled composite structure stuff. Sad. The eventual aircraft was way too heavy and slow, not what was originally designed. I'm still a few years away from retirement, and I'm looking forward to working on the next generation of aircraft and aircraft engines.

What is the area you live in like. Weather, Quietness, Scenery.

I'm an ex-Long Island New Yorker living in the small town of Bardstown Kentucky for the past 28 years. Bardstown is the "Bourbon Capital of the World." They say 85% of all the bourbon made comes from a 50 mile radius. The weather here is hot and humid in the summer and can get significantly cold in the winter. Growing up on an island, I was more used to the surrounding water moderating the temperatures in the summer and winter. That's not the case in the landlocked midwest US. The winters tend to be pretty grey as well, with not a lot of sunshine. Kentucky is a great place to live, we have mountains, lots of lakes, and lots of horse farms too. It's really green in the summer, and a pretty place to live most of the year.

Are you handy with your hands and have any hobbies.

I'm pretty good with my hands. At one time I actually built and tested turbine engines full time, and worked as an auto mechanic part time, so my skills grew at diagnosing and fixing mechanical "stuff". If it breaks,

I'll try to fix it. "Try" being the key. Sometimes I actually have to call an expert.

Hobbies are few. I actually enjoy home repairs and working on friends' and families' computers. I enjoy making things. I can work with wood, plumbing, and electrical pretty easily. Other than parts changing in a computer box, "electronics" is a bit of a mystery, so I stay away from anything which requires you to keep the "magic smoke" inside the box.

What is your education level?

I have a two-year associate college degree in engineering and subsequent classes in program management and sales. I once considered getting into the solar energy field, and took some HVAC classes as an entry point. I made more money with aircraft engines, so I never went into alternative energy. I still have an interest in those technologies though! Windmills and solar arrays still fascinate me.

Do you like to travel, go camping?

I have traveled a bunch on business. It's funny. You go on a business trip and never really stop to visit the places you go. It's get in, get out, go home. I've been to 47 of the 50 states, Canada, Mexico, Japan, England, and France.

I do like to travel! We did some camping when my kids were young, but my wife prefers air conditioning/heating and a private bathroom these days. Although I'm definitely not athletic, all of my boys are and enjoy hiking, rock climbing, camping, and boating. We tag along sometimes. I may get to visit Key West in a few weeks. We're looking forward to getting out of the cold for a change.

What caused you to try Linux and join this forum.

I think I'm a bit of a closet IT guy. If I didn't go into engineering I probably would have gone into the world of computers. I've always had an interest in

computers. I had a Timex-Sinclair ZX81 at first, and learned BASIC commands on that beast. I then graduated to a VIC20, and then a C64. I would type in programs from Compute magazine regularly and save them on the cassette recorder, and later a 5 1/4" floppy drive. My first "real" computer was a Trash 80 without a hard drive. Tons of fun there. My first IBM clone was an AST Advantage Multi-Media computer running a smoking 486DX2 33MHz processor with a 10MB hard drive. I STILL can't believe that thing cost \$2,100 USD in the mid-90's.

During this time I was working full time and working on my engineering degree part time, and was introduced to Fortran as part of my studies. Fortran on PUNCHCARDS, folks. The good old days.

One of my biggest achievements at that time was converting a turbine engine performance program written in RPN on an old HP41C programmable calculator into a BASIC program on an Apple II and then further converting it from BASIC to Fortran, once again on punchcards, and shared that with our customers using our engines. Good times! By default, I became the sysop for a Baby System 36 running COBOL for my company. Nobody else had ever touched a computer. I never got into COBOL though. I only knew how to turn it on, update it, and turn it off! All of this "user" and semi-sysop stuff kept my interest in computing alive while I made my money in the aerospace industry.

Finally, around the end of 2006 I had an old computer in the basement, and had read some articles about this new freedom-loving thing called Linux. I started some distro hopping and ran a few other distros in the short term before discovering PCLinuxOS Big Daddy 0.94. I was in love. After playing with Big Daddy for about 6 months I cobbled together an old Pentium III 450 system and permanently installed PCLinuxOS 2007 as a single boot. That was it. I was hooked. I joined the forum to sort out a few little problems and made some standard newbie mistakes. I once blamed PCLinuxOS for a problem which was really a KDE3

problem. Tex and O-P were kind, even though they knew I was being a total idiot.

I discovered the error of my ways and learned a lot. Over the years I've had some nVidia driver issues, which were quickly fixed (thanks Terry). But mostly, IT JUST WORKS. I have become a quiet evangelist, and converted my 82 year old mother's XP machine to PCLinuxOS early last year. She loves it. It doesn't break. It doesn't blue screen on her. She's happy. Even though I don't have a high post count, I do lurk a lot and read a lot and try to help if and when I can.

If you want, send some pictures of you and area of interest.

I attached a picture of myself and my wife on a trip to Las Vegas two years ago. Amazing I don't have many picture of myself!



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.

Visit Us On IRC

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

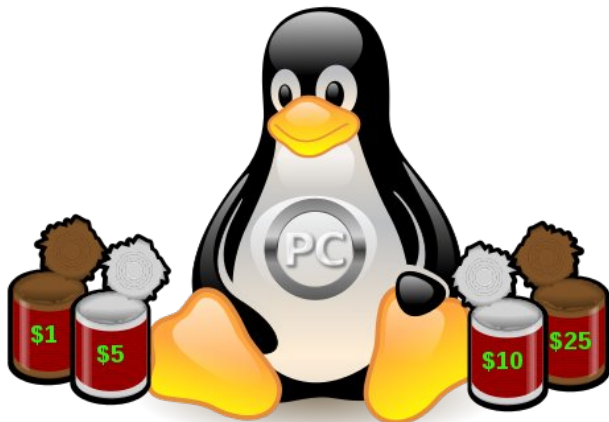


Donate To PCLinuxOS

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

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

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



Screenshot Showcase



Posted by zerocool, on July 14, 2015, running KDE

Playing Angry Birds On PCLinuxOS

by Alessandro Ebersol (Agent Smith)

Oh, but you will say, it's already possible to play Angry Birds on PCLinuxOS ... with Google Chrome. Angry Birds is an extension and it's ready. However, it's not the real thing. Now, with this tutorial, you'll play the real thing, the real Angry Birds, and offline, the same as Android.

But how? There's a way, and in fact, there is more than one way to run Android games on PCLinuxOS.

In order to understand how it works, we have to take a look at Google, and the Chrome OS.

The Chrome OS is the operating system of Chromebooks, and its characteristic is to function as a terminal that runs Google applications in the cloud. It was this feature that was criticized: after all, one uses an operating system for its applications, (the apps, and the killer apps) and with no local applications, the Chromebook would be a hard sell. Therefore, Google decided it would enable Android apps to run on the Chrome OS, Chromebooks and... in the Chrome Browser. The Chrome OS and the Chrome browser share many similarities, so what runs on ChromeOS can also run in the Chrome browser.



Understand the process

Natively, Android applications, APK's, can not run on the Chrome browser. To enable the Chrome browser to run Android APK's as Chrome applications, we need one extension, Archon Runtime, that allows Android apps to be installed as Chrome extensions. We also need an Android application, called the Archon Packager, which can package the Android APK, to convert it to a format that is "understood" by Google Chrome as an extension. Both programs are required for the Chrome browser, to make the Android APK's run as Chrome applications.

So get ready for the ingredients. You're gonna need :

Google Chrome browser – In PCLinuxOS repos (install via Synaptic).

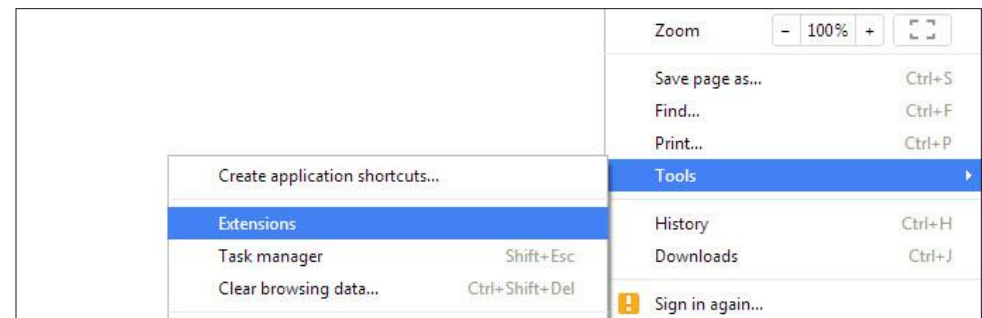
Application ARChon Packager (which must be installed in the Android device):
<https://play.google.com/store/apps/details?id=me.bpear.archonpackager>

ARChon Runtime for Chrome
<https://archon-runtime.github.io/>

In ARC's case, download the IntelX86 Chrome version 64 bit/64 bit OS Chrome. The latest version is 2.1.0.

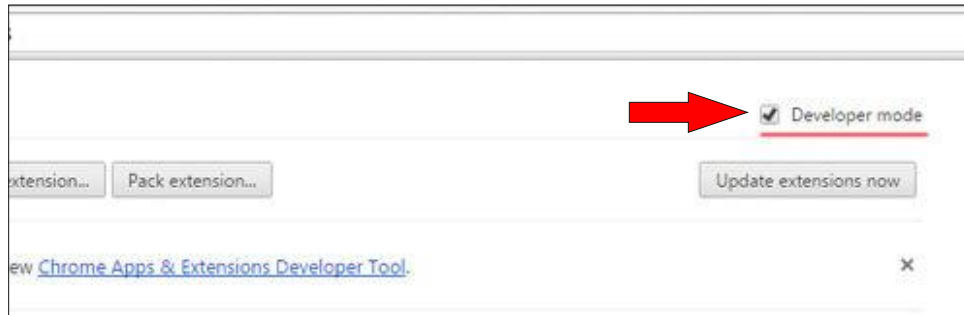
Angry Birds Android, <https://play.google.com/store/apps/details?id=com.rovio.angrybirds>. It can be downloaded from the Google Play store. The included link should take you directly there. Be sure to download one of the versions **prior** to the current 5.0.x version. You can also check this [alternative](#) download site, if you are having difficulty finding one of the older versions.

Now follow the steps below.



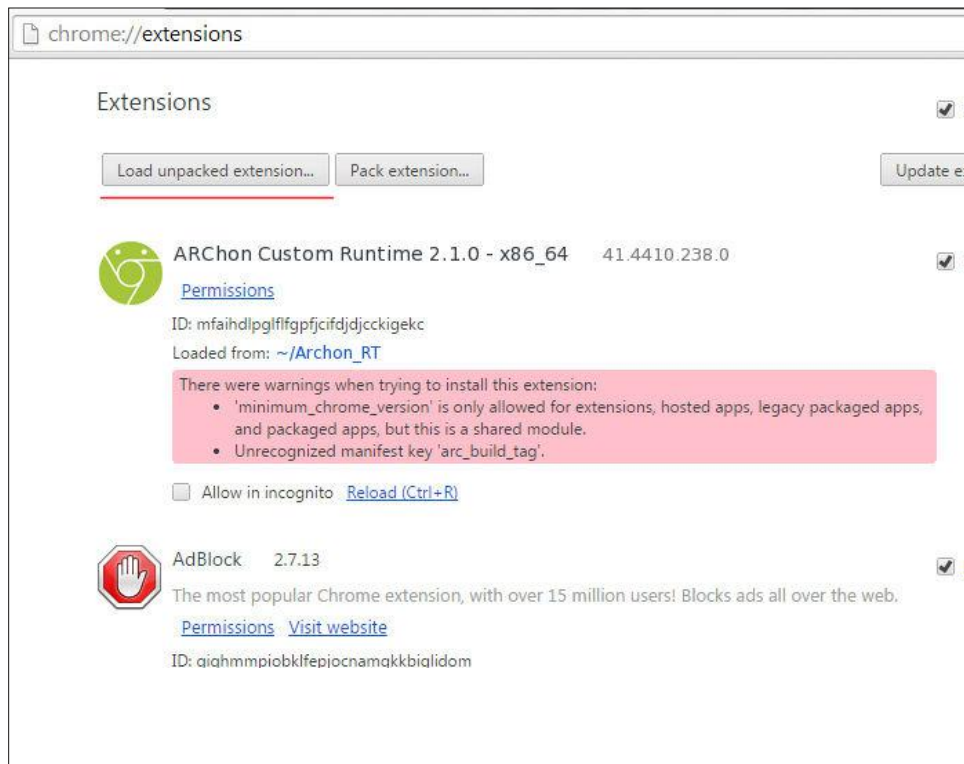
Step One: In Google Chrome, go to tools, extensions.

Step Two: In extensions, enable “Developer mode.”



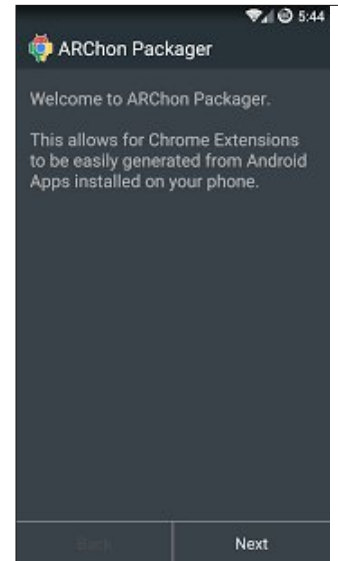
Step Three: Now download the ARChon Runtime for Chrome. It will be a compressed package.

Step Four: Create a folder in /home, and uncompress the ARChon Runtime for Chrome. I named it ARCHON_RT, but you can name it whatever you want.



Step Five: Click on “Load unpacked extension.” Ignore the error messages.

Step Six: Now, in the Android device, run Archon Packager. It should look like this:

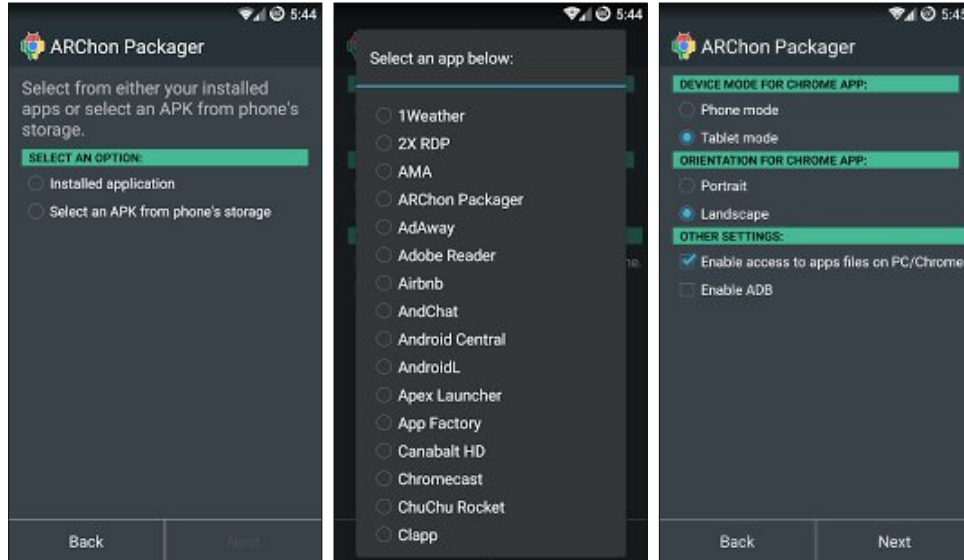


Step Seven: Click Next.

Step Eight: Archon Packager will ask to select an installed app or a stored app in the Android device:

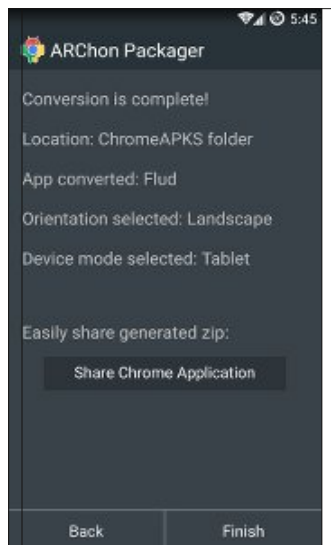


Step Nine: Now choose Angry Birds. Archon packager will list all the applications, the installed ones and the ones stored in the device's SD card.



Step Ten: Archon packager will ask for the settings for the converted APK package. It's straight forward in the settings. For Angry Birds, mark "Tablet mode," "Landscape," and unmark "Enable access to apps files or PC/Chrome."

Step Eleven: Press Next. The conversion will end, and the following screen will show:



Step Twelve: Click Finish, and the converted APK will be saved, packaged as a compressed ZIP file, in a folder previously chosen, in Archon packager's settings.

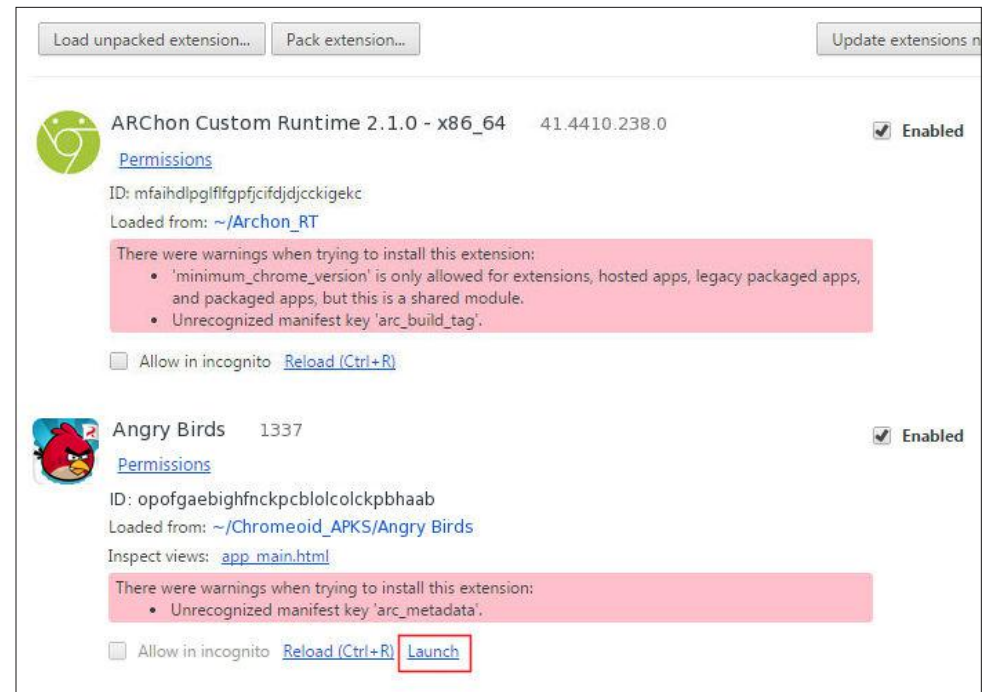
Step Thirteen: Transfer the compressed packaged APK to your PCLinuxOS computer (might be via Air Droid, USB cable, etc...)

Step Fourteen: I created a folder in my /home, Chromeoid_APKS, where I place the converted Android APK's. You can create and name the folder for the converted APK's as you like.

Step Fifteen: Now, in the folder you chose for the converted APK's, uncompress the AngryBirds.zip file.

Step Sixteen: In Google Chrome browser, click settings, extensions and click load unpacked extension.

Step Seventeen: Point to the Angry Birds folder, and, voilá, Android Angry Birds will be installed as an offline Google Chrome application.



Clicking **Launch** will run the program. Google Chrome will make an entry in the applications menu.

This tutorial works with Angry Birds versions prior to the most recent one, 5.1.0.

There are several groups of Internet users researching which Android applications run in the Chrome browser. In the link below, there's a list of applications that run smoothly, others that run with small issues and others that do not run at all:

https://docs.google.com/spreadsheets/d/1ilbxaftAu_ho5rv9fUIXSLTzwU6MbKOldsWXyrYio8/htmlview?usp=sharing&sle=true

There is also a group on Reddit about Android applications on Chrome: <http://www.reddit.com/r/chromeapk>. Note that all applications that depend on Google Play Services will not work.

In another article, I will write about another method to run Android applications on the Chrome browser. Until then, peace, health and a good time playing Android games in PCLinuxOS!



The PCLinuxOS Magazine Special Editions!

Get Your Free Copies Today!

Inkscape Tutorial: Abstract Wallpaper 2

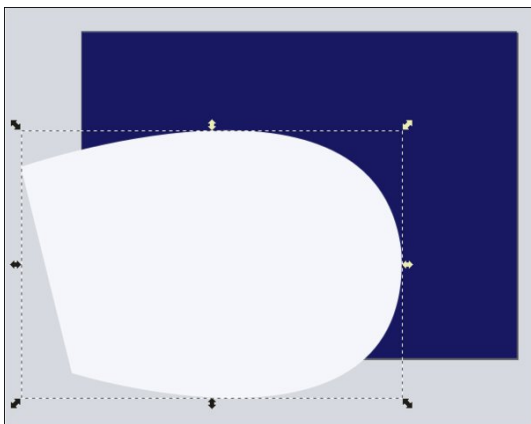
by Meemaw

We have already done a wallpaper using tiled clones. Now, let's just use lines, shapes and gradients. This project is actually pretty easy.

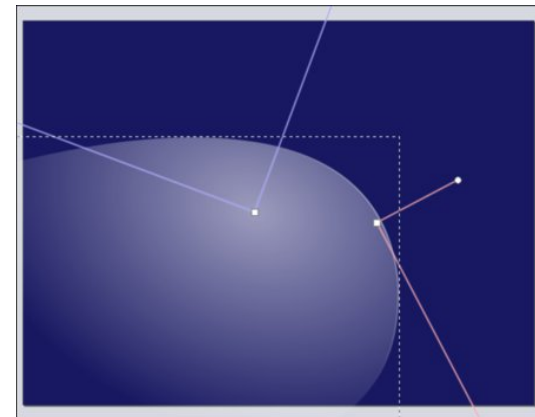
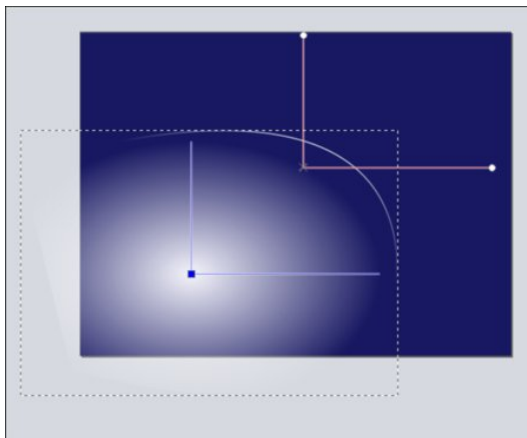
Open Inkscape and edit your document properties to reflect your desired wallpaper size. My desktop monitor is 1024x768, so I chose that size. You can make it any size you wish.

Draw a rectangle the same size as your document. I am partial to blue backgrounds so I used a dark blue, but you can use any color you wish as well. We will probably change it later, anyway.

Now for the gradients. Let's use the bezier tool first. Draw any figure, making sure you close the curve so it will fill. In your **Fill & Stroke** window, make the Fill white, and the Stroke a 3 px white line.



Go back to Fill and choose a radial gradient, then to Stroke and choose radial gradient there as well. Clicking your **Gradient** tool, you should see two gradient handles, one for the fill and one for the stroke. Grab the one for the fill first and move it where the figure you made looks like a highlight, then move the stroke handle so it emphasizes just a part of your stroke. You can also set the Opacity down on your color. I set mine down to about 60%.

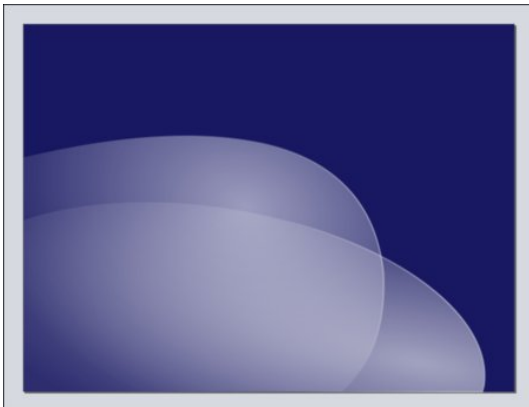
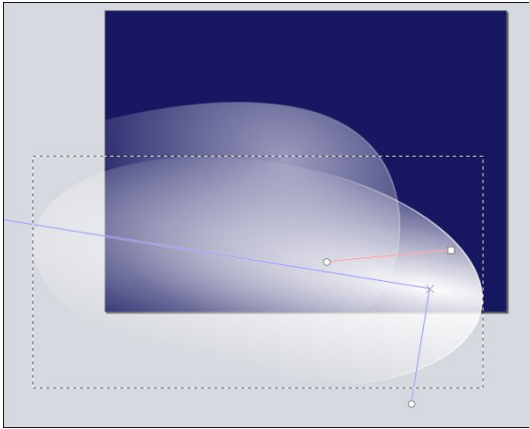


One thing that perplexed me was that the gradient handles seemed to jump together so when you wanted to move one, both of them moved. The way to remedy that is to hold down your Shift key while moving one of them. It will move away from the other one and then you can manipulate them one at a time.

To keep from having so much of your artwork “off the page”, you can also duplicate your background and then, holding the **Shift** key, choose your object. Click on **Path > Intersection** and the only part of your gradient you will see is that part on the page. You will still be able to move and resize it.

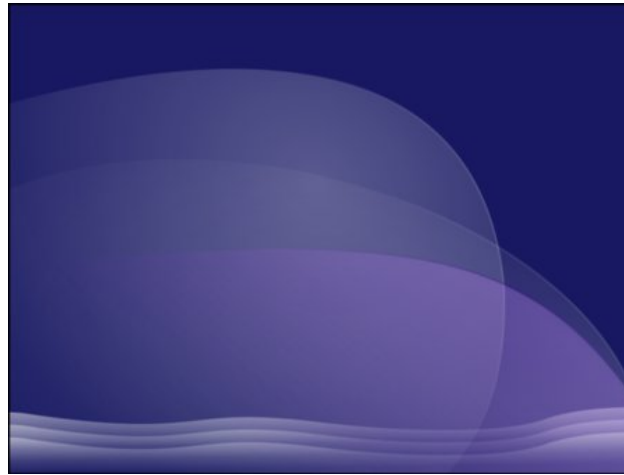
We can also use the **Ellipse** tool to do this. The procedure is the same. Simply draw an ellipse, change it to white, and configure the gradients the same way. You can use a linear gradient as well, depending on how you want your drawing to look (next page, top left).

You can put in as many of these figures as you want, arranging them any way that looks pleasing to you. You can also use your bezier tool to put in more



varied objects with different curves.... whatever looks good to you. After all, it IS your creation. I put a little purple in one of my gradients for a bit of variety (center, top).

Did you decide you wanted a different color background? Choose the background, change the color, and export it again (center). Easy!



Reach Us On The Web

PCLinuxOS Magazine Mailing List:
<http://groups.google.com/group/pclinuxos-magazine>

PCLinuxOS Magazine Web Site:
<http://pclosmag.com/>

PCLinuxOS Magazine Forums:
<http://www.pclinuxos.com/forum/index.php?board=34.0>



Support PCLinuxOS! Get Your Official

PCLinuxOS
 Merchandise Today!

PCLinuxOS



The PCLinuxOS Magazine

Created with Scribus



Does your computer run slow?

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



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

Get your PC back to good health TODAY!

Get



Download your copy today! FREE!

PCLinuxOS All your
Connect connections in one
convenient location!

LINUX
FORUM

Screenshot Showcase



Posted by tuxlink, on July 4, 2015, running Mate

Setting Up Your Own Chat Room In PCLOS-Talk Using Pidgin

by YouCanToo

You can set up your own chat room in PCLOS-Talk by using Pidgin. Let's get started!

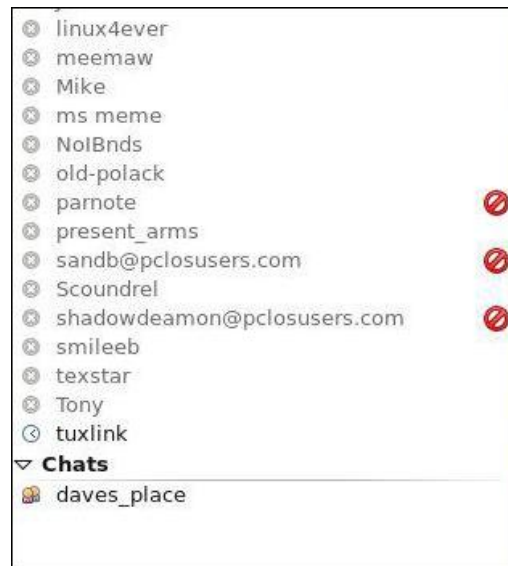
From your Buddy List, click on the Buddies tab, and then click "Add Chat."

This will open up a "Add Chat" window. Now click in the "Room:" and add the name that you want to use in the box to the right. Then click "ADD"

NOTE: You cannot have spaces or things like " or ' in your chat room name. Instead of spaces, use the dash or underscore instead.



Your chat room will now appear in your Buddy List (center, top).

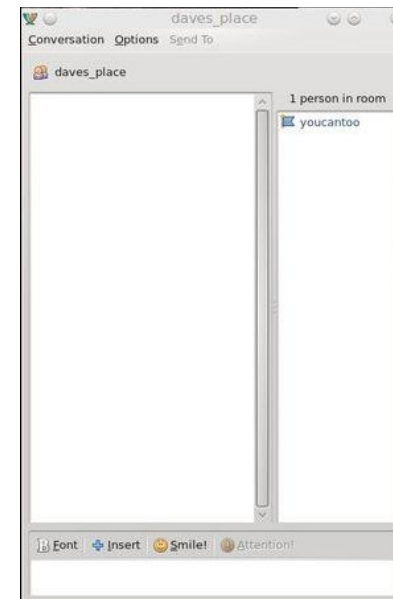


Now, double click on your new chat room. It will open two new windows. First, select the "Create New Room" window.



In this window you can set the behavior of your chat room. If you make changes, don't forget to save them. If you want to just accept the default configuration for your chat room, just click on "Accept Defaults." The "Create New Room" window will close, leaving your room window (right, top).

Now, to use your personal Chat Room ...



There are two ways for other people to find your new Chat Room:

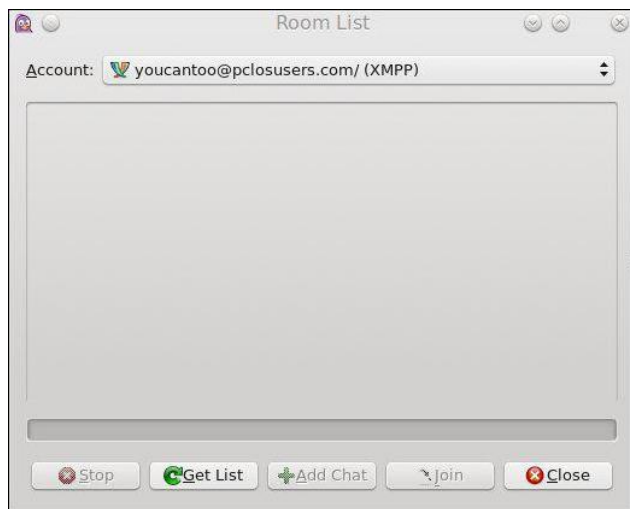
1. Invitation
2. Using the Room List from the Buddy List.

You can invite any of your buddies to your new room by using the invite button. In your room, click on "Conversation" and then select "Invite." This will open an Invite Buddy Into Chat Room window. Just enter your Buddy's username and fill in your Message (optional). Now, just press the "Invite" button and wait for your buddy to join the room.

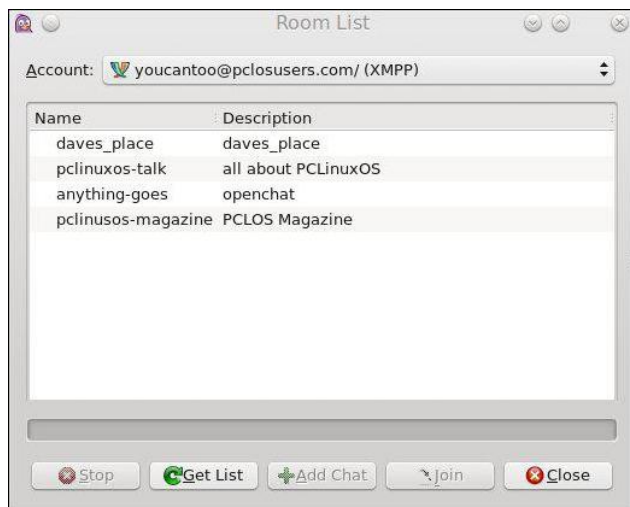


The other way is for your buddies to search for your room. To search for a list of all available rooms, click "Tools" from your Buddy List and select "Room List". This will open the Room List window (next page, top left).

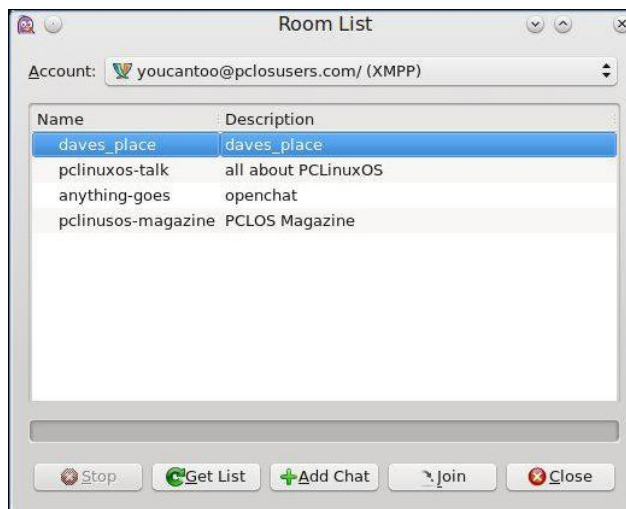
Setting Up Your Own Chat Room In PCLOS-Talk Using Pidgin



Now click on the "Get List" button and in the new window click "Find Rooms". A list of all available rooms will be listed. The only rooms that will not be listed are PRIVATE rooms. Private rooms require an invitation only.



Highlight the room you want to enter and then click the "Join" button (center, top).



You're now in the chat room. You will see a listing of all other users that are in the chat room.



When the last person leaves the chat room it disappears. The chat room will be no more, unless the person who first created it sets it to "persistent."



Does your computer run slow?

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

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

Get your PC back to good health TODAY!

Get



Download your copy today! FREE!

PCLinuxOS Puzzled Partitions

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

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

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

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

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



SCRAPPLER RULES:

1. Follow the rules of Scrabble®. You can view them [here](#). You have seven (7) letter tiles with which to make as long of a word as you possibly can. Words are based on the English language. Non-English language words are NOT allowed.
2. Red letters are scored double points. Green letters are scored triple points.
3. Add up the score of all the letters that you used. Unused letters are not scored. For red or green letters, apply the multiplier when tallying up your score. Next, apply any additional scoring multipliers, such as double or triple word score.
4. An additional 50 points is added for using all seven (7) of your tiles in a set to make your word. You will not necessarily be able to use all seven (7) of the letters in your set to form a "legal" word.
5. In case you are having difficulty seeing the point value on the letter tiles, here is a list of how they are scored:
 0 points: 2 blank tiles
 1 point: E, A, I, O, N, R, T, L, S, U
 2 points: D, G
 3 points: B, C, M, P
 4 points: F, H, V, W, Y
 5 points: K
 8 points: J, X
 10 points: Q, Z
6. Optionally, a time limit of 60 minutes should apply to the game, averaging to 12 minutes per letter tile set.
7. Have fun! It's only a game!



Double Word



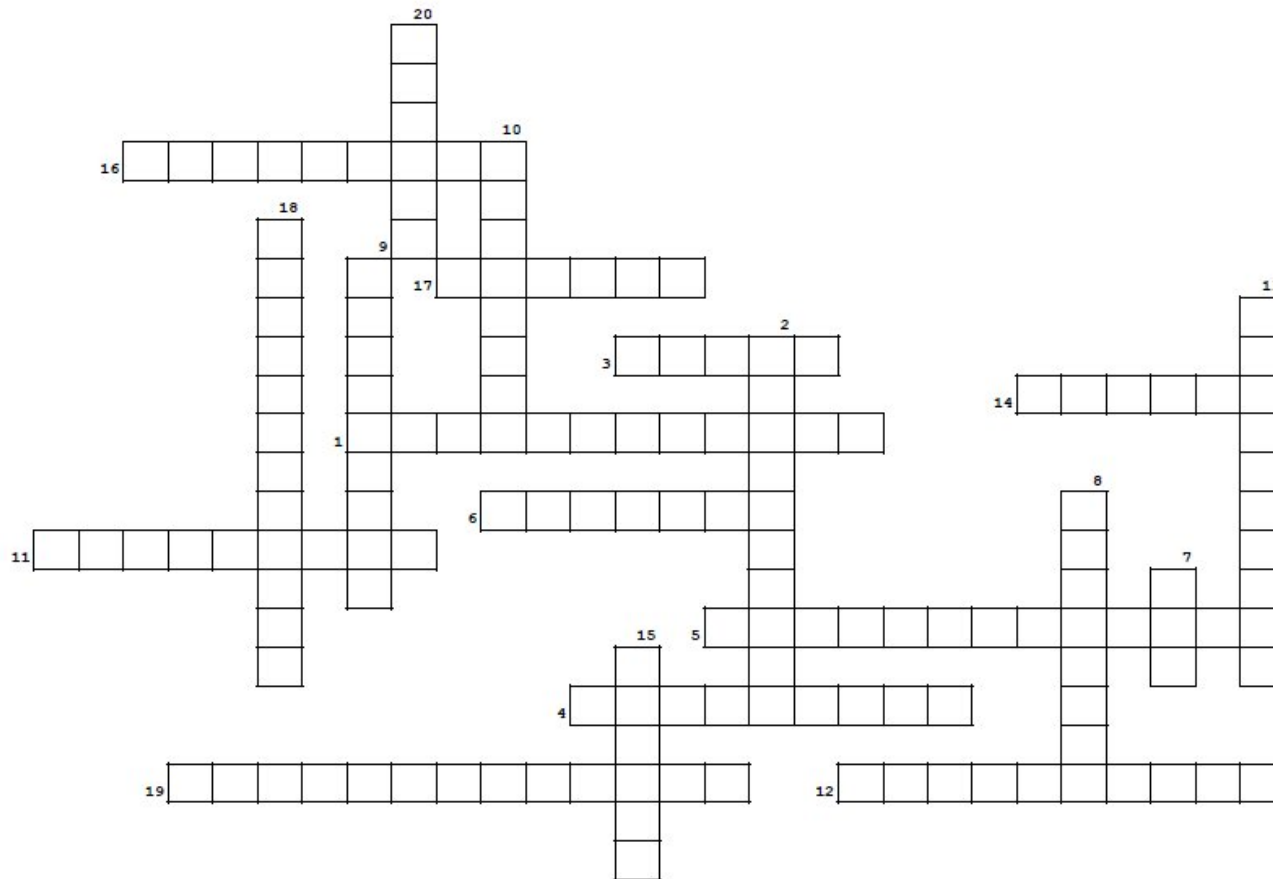
Triple Word



Possible score 219, average score 153.

Download Puzzle Solutions Here

PCLinuxOS Crossword Puzzle: August 2015 Vacation



1. You are sure to get wet even though you are in a boat!!!
2. Best to have a guide if you want to learn a lot
3. Another name for hotel?
4. Beautiful sights
5. That small basket flies high
6. Make sure your harness is secure
7. See animals you don't see every day!
8. Nice and warm after a day on the slopes
9. The water slide is fun!
10. Good for lying on the beach, relaxing
11. Most US states have them
12. Lots to do on that huge boat!
13. Taking it easy around the fire
14. The Smithsonian has at least 7 in Washington, DC
15. Out in the jungle
16. Nice, beautiful scenery, and usually cooler up there
17. Nice to visit if you like the ocean all around you
18. Yosemite and Grand Canyon, for example
19. Riding the ferris wheel is fun
20. Don't gamble your money away!

[Download Puzzle Solutions Here](#)

Vacation Word Find

```

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

```

airplane
 amusement park
 baggage
 beach
 cabin
 campground
 cruise ship
 ferry
 flight
 guided tour
 hotel
 inn
 island
 lake
 lodge
 mountains
 museum
 national park
 ocean liner
 passport
 safari
 seashore
 ski lodge
 state park
 tourist
 voyage
 waterfall
 water park
 yacht
 zoo

[Download Puzzle Solutions Here](#)

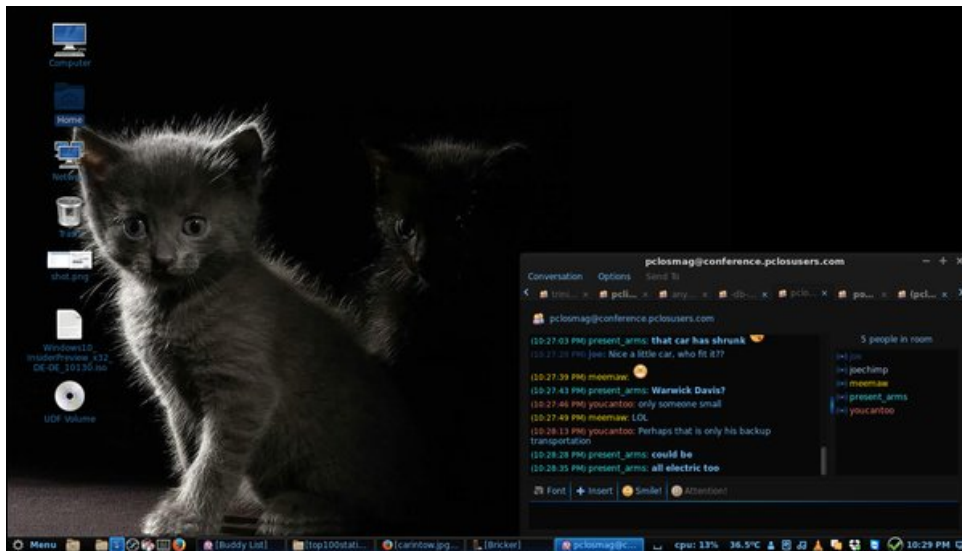
More Screenshot Showcase



Posted by Agent Smith, on July 23, 2015, running Trinity



Posted by francesco_bat, on July 5, 2015, running KDE



Posted by Joe, on July 9, 2015, running Mate



Posted by Meemaw, on July 22, 2015, running Xfce