The NEW PCLinuxOS Magazine

Volume 38 March, 2010

Table Of Contents

Disclaimer

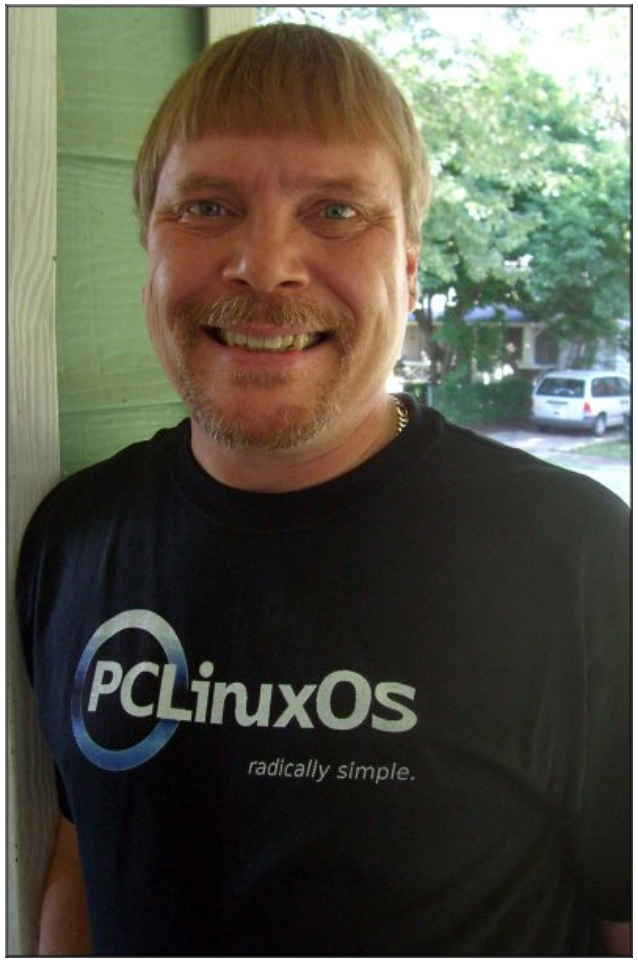
1. All the contents of the NEW PCLinuxOS Magazine are only for general information and/or use. Such contents do not constitute advice and should not be relied upon in making (or refraining from making) any decision. Any specific advice or replies to queries in any part of the magazine is/are the person opinion of such experts/consultants/persons and are not subscribed to by the NEW PCLinuxOS Magazine.
2. The information in the NEW 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 NEW PCLinuxOS Magazine and its associates shall not be liable, at any time, for damages (including, but not limited to, without limitation, damages of any kind) arising in contract, rot or otherwise, from the use of or inability to use the magazine, or any of its contents, or from any action taken (or refrained from being taken) as a result of using the magazine or any such contents or for any failure of performance, error, omission, interruption, deletion, defect, delay in operation or transmission, computer virus, communications line failure, theft or destruction or unauthorized access to, alteration of, or use of information contained on the magazine.
4. No representations, warranties or guarantees whatsoever are made as to the accuracy, adequacy, reliability, completeness, suitability, or applicability of the information to a particular situation.
5. Certain links on the magazine lead to resources located on servers maintained by third parties over whom the NEW PCLinuxOS Magazine has no control or connection, business or otherwise. These sites are external to the NEW 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 NEW 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.

Welcome From The Chief Editor

As Spring's arrival (in the Northern Hemisphere) brings a much anticipated annual renewal in nature, this Spring is also bringing the much anticipated renewal of PCLinuxOS, as users across the globe eagerly await the release of PCLinuxOS 2010. The wait is definitely getting shorter, as the **PCLinuxOS 2010 Beta 1** release was just announced by Texstar. It will go out to developers first, and then open to a public beta. For more information about the most recent Beta release, see the article on page four.



This month brings us another installment of Gary Ratliff's exploration of programming languages, with **Computer Languages A to Z: Icon**. We continue our look at those who work **Behind The Scenes** to make PCLinuxOS what it is, learning more about **travisn000** this month. With the new release of PCLinuxOS 2010 looming on the horizon, we continue our look at KDE 4 SC with three more articles highlighting what's new in KDE 4. First, **KDE 4: KRunner Grows Up** takes a look at the changes and enhancements to KRunner. Second, Andrew Strick (stricktoo) gives us a brief glimpse at **how to customize Dolphin**, the default file manager in KDE 4. Third, I explore the new document viewer in KDE 4, with **KDE 4: Okular Does More Than Just PDFs.**

I also show a way to use MP3 files to create secure passwords, in the article **Secure Passwords, Made Easy**. There's also a look at the results from the **LinuxQuestions.org 2009 Members Choice Awards**. Meemaw reviews **gbrainy**, for this month's **Game Zone** article. Galen Seaman (gseaman) reviews the **Music Notation Software in PCLinuxOS**. Andrew Huff (athaki) does a **web browser roundup**, taking a look at all the various web browsers in the PCLinuxOS repository. Peter Kelly (critter) picks up where he left off last month, and gives us **Command Line Interface Intro: Part 6**.

Additionally, all the regular features are still here, in the March 2010 issue. **Ms_meme's Nook** features another of her PCLinuxOS songs, and in keeping with an Irish theme for March, **Forum Foibles** features **Limericks**. Mark Szorady (georgetoon) gives us another installment of **Double Take**, along with another of **Mark's Quick Gimp Tip**. Plus, we feature another 10 screen shots, from the PCLinuxOS forum's Monthly Screen Shots, in **Screenshot Showcase**.

As the month goes on, pay particularly close attention to the PCLinuxOS home page and the PCLinuxOS forum. I predict that the announcements are going to come along pretty fast, as we move ever closer to the final release of PCLinuxOS 2010. Texstar and the Packaging Crew have worked hard on bringing PCLinuxOS 2010 to fruition, and continue to tweak it to live up to PCLinuxOS standards, where everything "just works" and is "Radically Simple."

Until then, I wish each of you prosperity, happiness, tranquility, and peace.

FLASH! 2010 Beta 1 Available for Download

by Paul Arnote (parnote)

By the time you are reading this, the PCLinuxOS 2010 Beta 1 will most likely be open to the public (or very soon will be). On February 27, 2010, Texstar announced that the PCLinuxOS 2010 Beta 1 was ready to go out to developers.

Keep in mind that this is **beta software**. As such, it is not intended for regular daily use, and the risk of data loss is real. So, you are warned – do NOT use this if you cannot risk the data loss.

Before installing this, or any other operating system update, it is also highly recommended that you back up your current data. You might also want to create a remaster of your current system ... *just in case*.

Of course, the painless way to give the PCLinuxOS 2010 Beta 1 a good shake-down is to install it into a virtual machine, such as VirtualBox or VMware. So, now on with the particulars.

What's included in the PCLinuxOS 2010 Beta 1?

* Linux kernel 2.6.32.8pclos4
* KDE 4.4 SC
* Updated xorg server
* Updated mylivecd
* Updated firmware packages
* Plymouth bootsplash
* Updated KMenu arrangement
* Speedboot
* and lots more updates!

The 2010 Beta 1 represents many months of continued, hard work by Texstar and the Packaging Crew, culminating in a very up-to-date PCLinuxOS that has the latest stable applications, desktop environment, and kernel. Additionally, PCLinuxOS 2010 will have a PAE kernel to allow users with more than 4 GB of memory to have access to that memory, up to 64 GB. Besides the regular kernel and the PAE kernel, PCLinuxOS will also have a legacy kernel (for older hardware), a real time kernel, and a kernel that uses BFS scheduling.

Check the PCLinuxOS home page and the PCLinuxOS forum for more details on how to download the PCLinuxOS 2010 Beta 1.

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.

The PCLinuxOS Magazine

The NEW 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: Andrew Strick (Stricktoo)
Consultants: Archie Arevalo, Tim Robinson
Artwork: Sproggy, Timeth, ms_meme

Magazine Layout: Paul Arnote, Meemaw, ms_meme
HTML Layout: Galen Seaman

Staff:
Neal Brooks
Galen Seaman
Patrick Horneker
Guy Taylor
Andrew Huff
Gary L. Ratliff, Sr.
ms_meme
Mark Szorady
Macedonio Fernandez
Meemaw
Andrew Strick
Paul Arnote

Contributors:
Peter Kelly

Behind The Scenes: travisn000

by Paul Arnote (parnote)

In the continuing "Behind The Scenes" series, we get the chance this month to learn more about Travisn000, a global moderator on the PCLinuxOS forum. Travis is also a key player behind the PCLinuxOS Wiki, and an active PCLinuxOS developer and packager.

Would you please introduce yourself ("real name," age, location of residence, marital status, children, etc.)?

On the PCLinuxOS forums my "name" is travisn000. In real life, my name is still Travis N! I am 35 years old, married for the last 8 years or so, and have three kids. I would tell you more, but I would be breaking the rules I set for them! (They have been given strict training to not divulge personally identifiable information on the internet ;))

When did you first get started with computers, and what OS did you start out with?

My first introduction to computers was on an early Apple (IIe ... I think). Because I didn't join the school music program like most kids would in the school at that time, I spend band class in the new computer lab. I think I was in 4th or 5th grade. Mostly what I remember about it was the sense of accomplishment I got by drawing things with the "turtle" on that old black and green screen. At about the same time my step-father brought home a computer that looked like an old suitcase with a 7" screen, and soon after I started middle school in a brand new school with the latest first gen Mac. I don't think too many days went by that I didn't get to school before the computer lab opened at 6:30am.

When did you make the switch to Linux, and which distro did you start with?

I started using Linux after getting interested in building a Home Theater PC (HTPC). I had come across a few articles about MythTV and decided to give it a try. After a lot of reading, and becoming more familiar with the concept of "free" software, I decided Fedora was the right distro for me. I recall downloading Fedora 3, only to see Fedora 4 was coming soon (early 2005). Fedora 4 was the first distro I installed for any significant use; it became the base for my first MythTV setup.

When did you make PCLinuxOS your home distro, and what attracted you?

My first introduction to PCLinuxOS came about a year later. I had gotten more comfortable with Linux and had started tweaking my Fedora/MythTV box, eventually tweaking it to the point where I could no longer get it to boot (oops!). I tried installing the latest Fedora and MythTV but could never get it set up and working quite right. I played with a handful of "live" distro's, including PCLinuxOS 0.93, but for a variety of reasons did not install any of them. When I came back to Linux, PCLinuxOS 2007 was newly released and I was hooked! I think it was the first distro I found that did not require endless hours to get a basic working desktop set up, and I loved the control center. I haven't left since.

Do you currently run any other OS's or Linux distro's? If so, which ones?

The only other distro that has made it on my hard drive for more than a few days is the Arch based Chakra Project (http://www.chakra-project.org/). For me it, is the only Desktop experience that comes close to the combination of simplicity, flexibility, power and speed that PCLinuxOS offers.

As a global moderator in the PCLinuxOS forum, what do you view as being the most important aspect of your role? What is the most challenging aspect of your job as global moderator? The most rewarding?

To be quite honest, I am kind of a slacker when it comes to moderating the forums. We are blessed to have one of the best communities on the internet, and because of this, it seems the need for moderation is quite rare. For the most part, my authority as a moderator is used mainly for moving, combining, and generally helping keep things organized. The thing I find most rewarding in my activities on the forums is really just the everyday aspects of helping people to make PCLinuxOS their favored desktop OS. By this, I mean those things that most all members of the community try to do: welcome new members and help them get comfortable, trouble shoot problems and post solutions/tips, etc.

Besides being a global moderator in the main PCLinuxOS forum, you are also involved with the PCLinuxOS Wiki. What has been the greatest challenge in getting the Wiki site going?

PCLinuxOS has been through a few wiki sites in last couple of years. Most of them met their demise as a result of poor organization, making them difficult to use. Because of this, a handful of members at MyPClinuxOS.com agreed to try and rebuild it once again and make it more usable; we wanted it to better reflect this great distro. At the time, I knew very little about using/making a good wiki. There was a consensus on the forum to use MediaWiki as the base and Cindy (aka Linuxera) set it up on one of her servers. As most every one seemed to have very little extra time (including myself) to devote to the project, I took it upon myself to get the basic framework for the site set up. I wanted to have a good framework that others could build upon.

How can regular users best contribute to the PCLinuxOS Wiki?

The best way that users can contribute to the wiki is to jump in with both feet! Signing up is easy: visit the wiki and click the "create an account" link at the top of the page. The wiki is set up so that any registered user can add or edit content. I think that if people from the community took a little extra time to document the steps they use in setting up various aspects of their installation and posted them to the wiki in appropriate places, we would soon have a wiki that rivals the biggest and best.

There is also much work to be done in translating the existing wiki content into other languages. Angel02_de has been working hard at adding German translations, while Nakux, Melodie, and Arl have been making great strides on French translations and content.

I do my best to make myself available to help those that need it. Send me a PM on the main forums if you need help getting started or have any questions. The Help section of the wiki also has links and information to help people get started.

Currently, what are the largest needs on the PCLinuxOS Wiki?

Wiki's are all about content. While we have a pretty good base of content going, more would be better. It would be nice to see users begin to document the steps they take to accomplish various tasks when setting up their computers. I will be trying to add, revise, and complete a few how-to topics on the wiki as I go about reinstalling with the 2010 release; perhaps it will be a good time for others to do the same?

You are also quite involved with the Advanced Users section of the forum, lending your hand to helping provide GTK+ Dialog graphical interfaces to command line scripts. What advice can you give to others who may want to learn these topics or help with providing GTK+ Dialog graphical interfaces?

Most of what I have learned about writing scripts I have learned from trial and error (and reading other existing scripts). My advice would be to just jump in. Spend time trying things out on the command line; if you have questions don't hesitate asking. Zenity is a great tool for adding basic dialog boxes to shell scripts, and there are a many good tutorials on how to use it. I think many of my scripts begin with just testing command and their options in konsole; I try to share and preserve the knowledge I gain by posting in the forums and/or wiki. Google is a great resource as well.

You are also quite involved with PCLinuxOS packaging. What advice can you give to others who may want to learn packaging, and help with the development of PCLinuxOS?

Packaging software for the repo can be alot of fun. Some software can be very challenging, while others are quite easy. Currently we don't have great documentation for PCLinuxOS when it comes to getting started packaging, as the server that had the packager's wiki is gone. Rebuilding it is another project that is on my to do list, but with some of my other personal responsibilities, I'm not sure when I will get around to doing it.

Currently the best choice is probably to browse through the various topics in the packaging section

of the main forums, and also take a look at the (archived) santa's helper section in the forums over at MyPCLinuxOS.com. Don't hesitate to ask questions, this is a community effort; your questions and answers make us all better at what we do.

How would you best describe your approach to problem-solving? Is it different than the approach to take as a global moderator in the forum? If so, how is it different?

This is a tough question to answer. I don't think I have a standardized approach. If I were to make up an answer to this question, I would say, "Backup first, then analyze the problem; think of all of the things that might be the source of the problem and try to rule them out (or in)." Sometimes this involves a bit of trial and error and a lot of online searches, but it also usually results in great educational experience.

If you had to pick one piece of music that sums up your outlook on life, what would it be?

Don't know... whatever is on the radio works for me!

How much time (per week) do you devote to your job as global moderator, and towards packaging and development in PCLinuxOS?

Again, I don't really know; there are times when I will do nothing but work on PCLinuxOS related things for weekends and many of my evenings, but as my kids are getting to be more active in various activities (and my wife volunteers me for more and more "volunteer" activites! :D), I find I have to be a little more judicious with my time. I still try to contribute what I can when I can, but I find I tend to focus more of my time in places that will enable others to contribute as well.

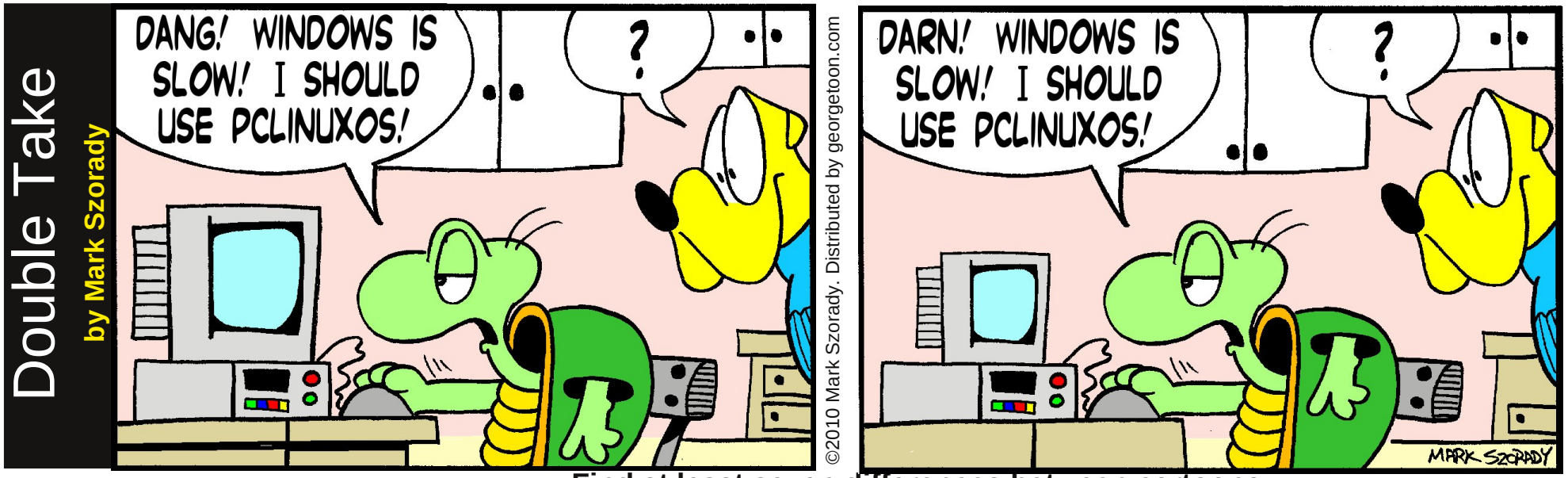
With the release of PCLinuxOS 2010 just literally right around the corner, what do you view as the biggest challenges facing users?

I'm sure there will be a few long time KDE 3.x users that will be reluctant to move on to KDE4, and there will also be users that are challenged by the process of re-installation and preserving various aspects of their setup. Luckily, PCLinuxOS has a great community of people that will help out those that ask for it. In the end, I think most will look back at the transition and find that it was not so bad as the fears and concerns that we might currently have in anticipation of it.

What words of wisdom would you like to leave us with?

Be nice, have fun, and enjoy the journey! And keep a good back-up or two!

Double Take & Mark's Quick Gimp Tip



Find at least seven differences between cartoons.

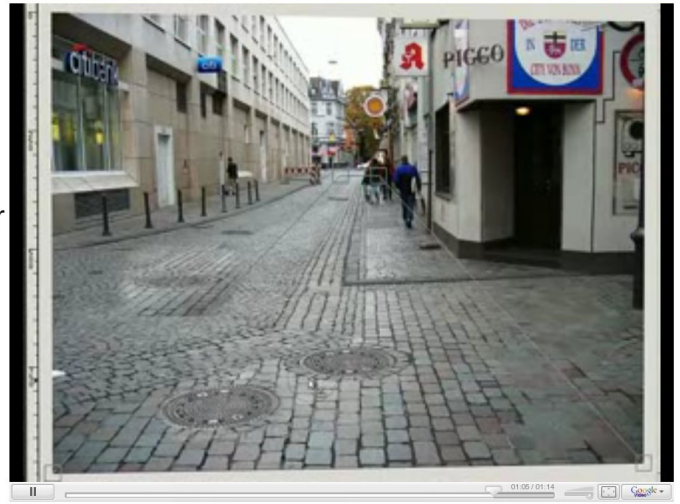
Answers on Page 12

Mark's Quick Gimp Tip

Just like commercial graphics apps, The Gimp is constantly being developed and improved. The really nice distinction, though, is with The Gimp, anyone can contribute to its development! The Gimp community welcomes and encourages input/assistance from anyone interested in helping develop this terrific graphics app! Check out http://www.gimp.org/develop/ for ways in which you can help.

And the community really responds. Each release of Gimp finds new improvements, tools, and features. One such tool that was introduced with Gimp 2.4, is Perspective Clone. Using this tool, you can select an object, say in the foreground, and clone it in perspective farther away in the background.

It's a terrific tool that gives the user one more trick for manipulating photos and images. For a video demonstration of how the perspective Clone tool works, visit http://video.google.com/videoplay?docid=-3077868802879051003# (At right is a snapshot from that video showing the Perspective Clone tool in action.) Get Gimp and give the Perspective Clone tool (and all Gimp's tools) a try!



-Mark Szorady is a nationally syndicated cartoonist. His work is distributed by georgetoon.com. Email Mark at georgetoon@gmail.com.

Computer Languages A to Z: Icon

by Gary L. Ratliff Sr. (eronstuc)

The Icon programming language was invented by the same gentleman who invented SNOBOL. Many people mistakenly visit its web site thinking that this is a tool to help them design icons! Dr. Ralph E. Griswold stated in his book, "The Icon Programming Language, 3rd Edition," that the word does not stand for anything in particular. However, this was designed to be a language which would make the development of computer programs easier for the programmer.

Icon was written in the C computer language, which was discussed in the Sept 2009 issue of this magazine. The language is in the public domain and is available from the University of Arizona. The web site will contain versions for Unix, BSD, Mac, Windows, 32 and 64 bit Linux, and some other computer platforms. What we shall do is create a directory in /home named Icon, and store the files which we download there. The main site for information on the Icon programming language is at the University of Arizona. Their Computer Science Department has won several awards for being one of the very best in the nation.

Unfortunately, Dr. Griswold passed away in 2006. There is a memorial web site established in his name. The information on Icon may be obtained from http://www.cs.arizona.edu/icon. These are the files which I placed in my home/gary/icon folder: linux.v943.tgz (which is reached from the Unix link on the main page.) Now go on over to http://www.cs.arizona.edu/icon/books to obtain some starting documentation to learn to program using Icon. These are all available in pdf form: The Icon Programming Language 3rd Edition, Graphics Programming in Icon, The Implementation of the Icon Programming Language, and The Icon Handbook. This should keep you busy learning some of the features of this language, but first we will need to install and make the system usable on our PCLinuxOS Systems. For this step, we will become and remain root during the next several steps.

Establishing a Working Environment for Icon

What we are going to do is follow the recommendation of the developers of the language, as found in the README file found in the linux.v943.tgz file, and this system will be installed in /opt. Once this has been done, we will create several symbolic links to an area on the default PATH so that the system will load on command. And finally, we will make it so that the supplied man pages may be loaded automatically. Now that we have established our goals, here are the commands to accomplish these items, which shall be performed from a terminal.

```
su
<enter the root password>
mv /home/gary/icon/linux.v943.tgz /opt
cd /opt
tar -zxvf linux<tab>
mv linuz.v943.tgz /home/gary/icon
mv icon.v943 icon
cd icon/man/man1
bzip2 icon.1
bzip2 icont.1
mv *bz2 /usr/share/man/man1/
cd /usr/local/bin
ln -s /opt/icon/bin/icon icon
ln -s /opt/icon/bin/icont icont
ln -s /opt/icon/bin/iconx iconx
ln -s /opt/icon/bin/vib vib
exit
```

Once the exit command has been given, you revert to being a normal user and can issue commands without worrying about destroying the system should you make a silly typing error. The files which make up the Icon language system now reside in their preferred location: **/opt/icon**, and under this system we find the directories: **/bin**, **/doc**, **/lib** and **/man**, which contain the usual items found in directories with these names. However, the manual pages are made up of bzipped files, which are located in the /usr/share/man structure. So before we performed a few operations on the files in /opt/icon/man/man1, we could have only received a no manual page exists for icon, and the same for the item icont. We are now ready to begin creating icon programs..

Interestingly enough, a file which is edited in emacs, which has an .icn extension, may be compiled from within emacs (unless it of course has user input). Just as there is a Fortran menu item for one with a .f extension, so too will there be found an icon menu item for emacs. You get the drill. The very first program to be developed is the infamous Hello World program. So from a terminal, issue the command: emacs hello.icn and enter the following code into the editing buffer:

```
procedure main()
  write("Hello World")
end
```

That's it, short and sweet. So first, save the buffer. As this requires no user input, we are ready to compile. So click on the tools menu item and select compile. Then backspace over the make -k item which will appear, and replace it with the command :

icon hello.icn

A message will appear in the compilation buffer telling you that the operation was a success, and the message "Hello World" will be displayed. Next, we will show you a simple program to compute the powers of the first twenty five integers for the radix from 1 to 5. And, we are going to display the data in a nicely formatted report. A picture is said to be worth a 1000 words, so here is a snap of the program and its output as entered and compiled in emacs:

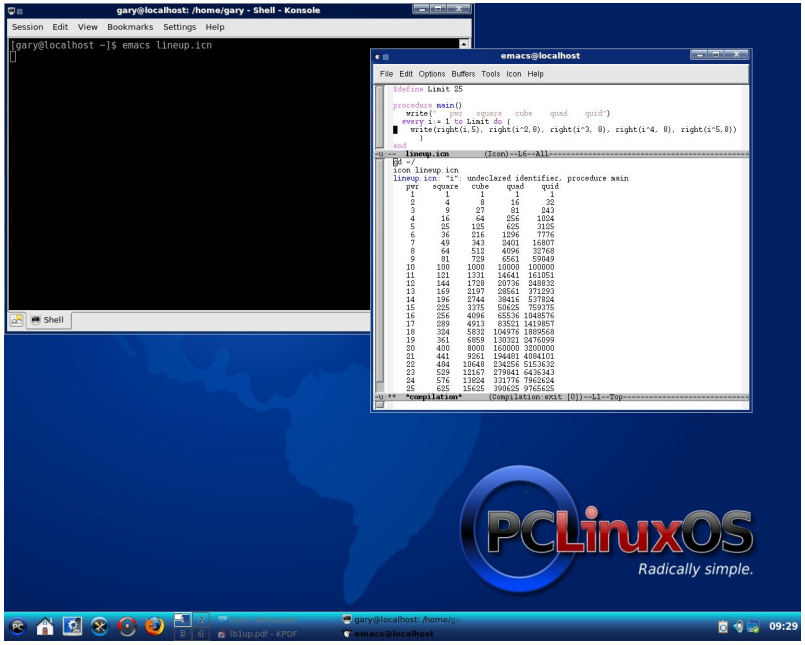
Since the picture is a little small here is the text of the program which I called lineup.icn:

```
$define Limit 25

procedure main()
  write(" pwr  square  cube  quad  quid")
  every i:= 1 to Limit do {
  write(right(i,5), right(i^2,8), right(i^3, 8),
right(i^4, 8), right(i^5,8))
   }
end
```

Features in Icon Not Found in Other Languages

Now would be a good time to introduce some of the features which seem to be unique to the Icon language. The first item, which I had never heard about, is an item called cset. This is mentioned quite a bit, however. What it means is seemingly taken for granted. Essentially, a cset is a set of all the characters within a given string. Now a set is defined as only containing one of each type item. So, if a string contains two or more of the same letter only one would be included in the cset. Also the items are arranged in order. The upper case letters are before the lower case items and numbers are before letters. So the cset of "hello" would be "ehlo." The cset of "Hello" would be "Helo," as this has an upper case H. It is easy to learn the cset of an item by casting this into a main procedure and compiling it in emacs.



For example: A := cset("My name is Gary"); write(A) would produce the output: Gmaeimnrsy. As you might expect, these raise hell with the spell checkers.

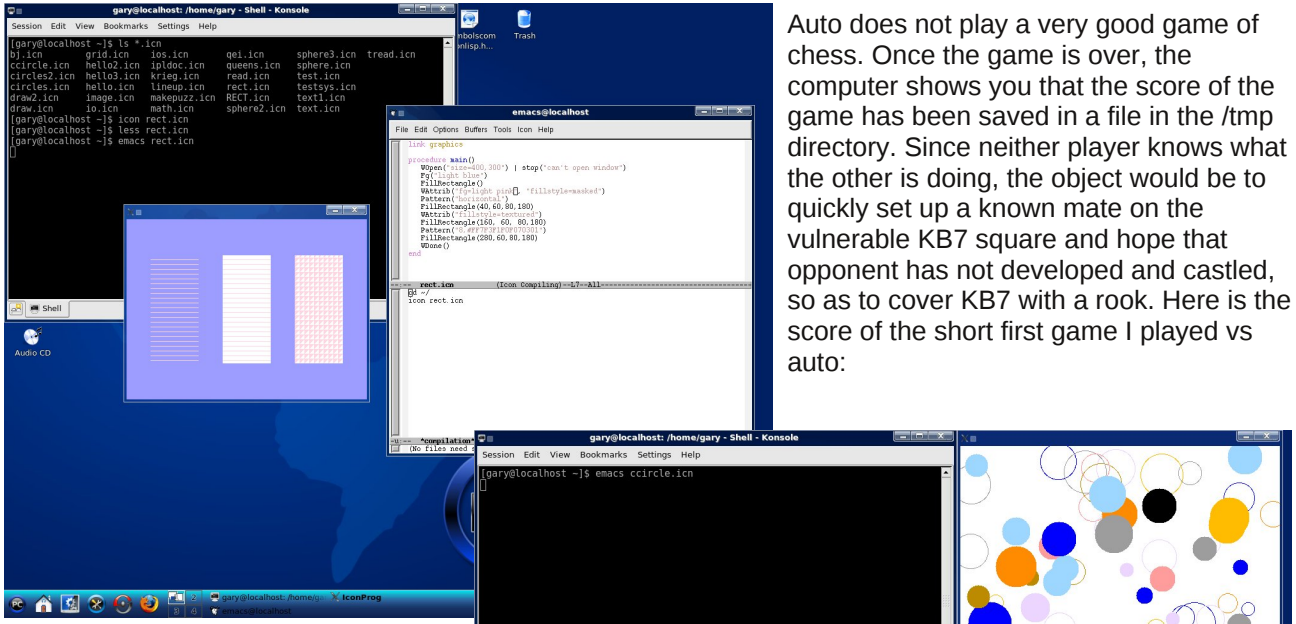
The execution of conditional statements is based on the concept of success or failure. An item which might have several answers may be suspended once the first success if found, and then continued after this to go on to find other answers. When all possible answers have been found, it would then fail.

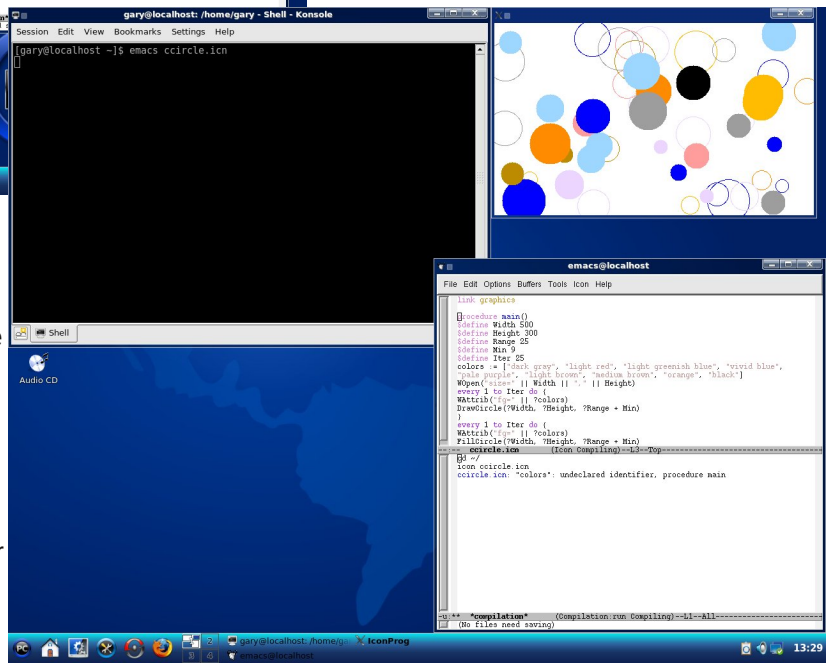
Also many of the symbols are used in an unconventional manner. This makes the reading of the manuals very important to gain any kind of understanding of what is going on in a program. A really fast introduction to the language was found in a set of slides presented as a course in Icon by Professor Mitchell.

A Little Bit of Graphics Programming

In the doc section, you should find some html pages. These may be launched from a terminal by entering firefox xyz.htm from within the doc folder. The file index.htm will present a flle for use in finding other areas of interest. In any event, there is a simple introduction to the graphics facilities. To create a window, you use: WOpen("size=400,300"). Then certain attributes of the window, such as foreground color and backgroundm may be defined. The function WDone() prevents the window from closing until the user enters Q or Control + C to terminate the program.

Here are some screen shots of the output of some of the graphics programs:





The main Icon library at the University of Arizona contains some remarkable programs, and every one I have downloaded seems to indicate from the start that these are in the public domain. There is bj.icn, which plays a game of black jack. Puzzle.icn generates word game puzzles. And for chess fans, there is krieg.icn, which plays the game Kriegspiel. This is a version of chess in which neither player sees the board. The player asks questions to learn if there are any captures available. The computer acts as the referee, and will also volunteer to play the role of an opponent named "auto."

Auto does not play a very good game of chess. Once the game is over, the computer shows you that the score of the game has been saved in a file in the /tmp directory. Since neither player knows what the other is doing, the object would be to quickly set up a known mate on the vulnerable KB7 square and hope that opponent has not developed and castled, so as to cover KB7 with a rook. Here is the score of the short first game I played vs auto:

Kriegspiel game begins Monday Sept. 7, 2009 8:45 pm
gary auto
Pe2e4 Pc7c6
Pd2d4 Pg7g5
Ng1f3 Pd7d5
Nf3e5 Pd5e4
Bf1c4 Bc8h3
Bc4f7
Result: 1-0
Kriegspiel game ends Monday Sept. 7, 2009 8:57 pm

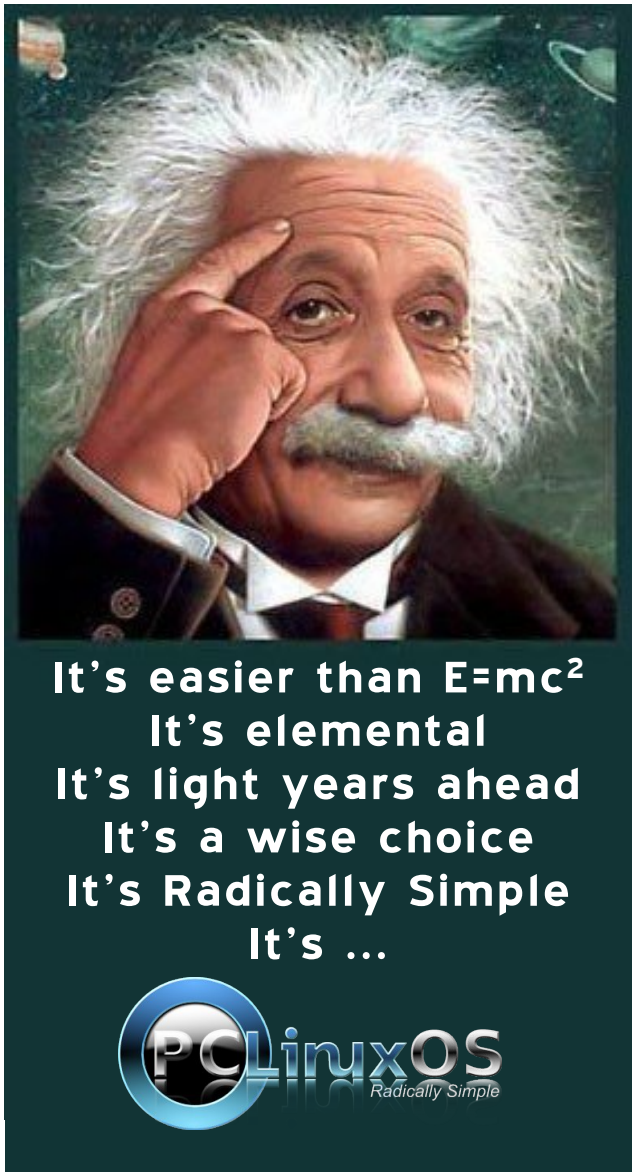
I hope that this is enough to whet your appetite for learning more about the use of the Icon Programming Language. As you have learned, there is a vast amount of material available, and most of it is in the public domain. There are also other offshoots from Icon. Noteworthy are Idol and Unicon. These are efforts to introduce the concept of OOP into icon. The Idol is an Object Oriented set to be used with Icon as a pre-processor, while Unicon is a full Object Orientated version of Icon developed by Clinton L. Jeffery.

I have also received a message from Greg Townsend, now retired, that Icon is in active use at the University of Arizona. I think that this language could easily be added to your tool kit of programming languages.

Answers to Mark Szorady's Double Take:
(1) Turtle arm higher; (2) Computer monitor smaller; (3) Seat post missing; (4) Table different; (5) “Dang” changed to “Darn”; (6) Cabinet knobs lower; (7) Drawer missing.

Screenshot Showcase

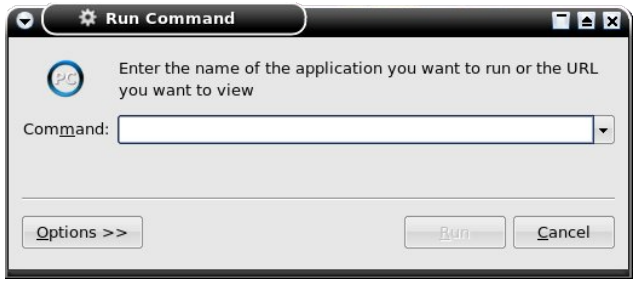
AndrzejL, running KDE 4, posted February 5, 2010



KDE 4: Krunner Grows Up

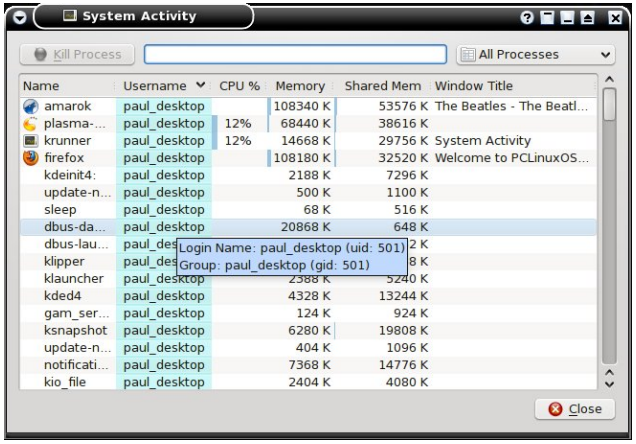
by Paul Arnote (parnote)

Of all the things that I have discovered in KDE 4 SC, Krunner represents perhaps the most understated change for KDE users, and cleverly cloaks its capabilities quite well in a very simple interface.



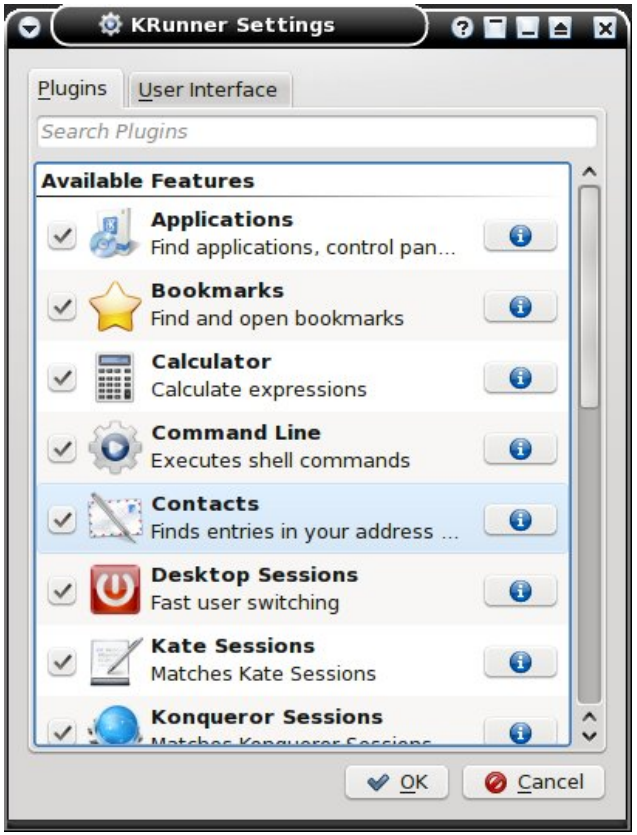
Under KDE 3.5.x, pressing Alt+F2 brought up Krunner (image above). The same thing happens in KDE 4 SC, but let's just say that it has ***really*** grown up, and now has a ton of new tricks up its sleeve. Sure, just as in KDE 3.5.x, typing in the name of a program you want to run will launch the specified program. But there is so much more that it can do, so hang on while we explore some of those added powers and abilities. It's kind of like Clark Kent – it appears to be meek and mild-mannered, but under that meek veneer lies powers untapped.

As seen in the image above, there isn't much to get excited about – at least initially. At the middle of the popup window is where you will type in the task you are wanting to perform. And under KDE 3.5.x, this was the name of the program that you wanted to run. But there are more things left to explore here. Working from right to left, there is the "X". Clicking on that will close out the Krunner window. Next, is the "?" icon. Just as you might expect, clicking on it displays help for the various functions that Krunner is capable of performing in a popup area just below the current window.



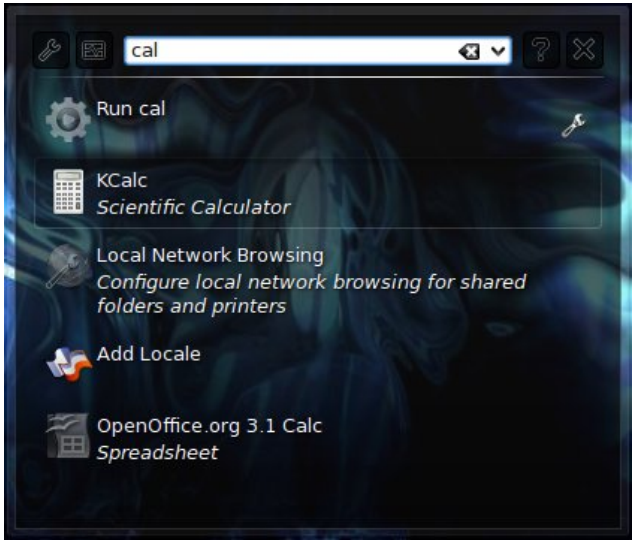
Clicking on the icon to the left of the entry box will bring up the System Activity window, where you can view all the active tasks currently running on your system. From here, you can highlight a running task, and click on the "Kill Process" button at the top left of the screen. This is helpful if you have a runaway process that's trying to bring your system to it's knees.

By clicking on the icon at the far left of the screen, you can specify which of the available plugins Krunner will use. Turn them off (or, un-check them),

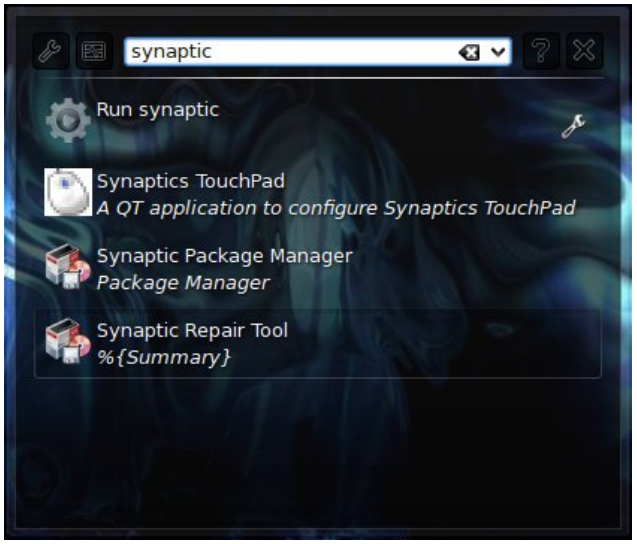


and Krunner will behave just like it's older and less capable sibling in KDE 3.5.x. Turn them all on (or, check them), and you will be able to tap into the full potential of the new features that have been added to Krunner. Understandably, there may be instances (like on a network) where you may not want users to have access to certain things, and in that case, it makes perfect sense to limit that access by limiting the functionality of Krunner by deselecting those items you want to restrict access to.

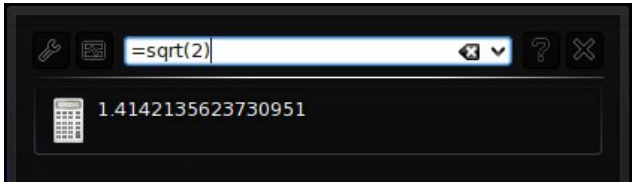
So let's explore some of the new functionality that has been added to Krunner. Starting with the basics, let's say I want to run a calculator program. I start typing in c - a - l into the text entry box in Krunner, and just as soon as I type in three letters, all the options where "cal" appears in the file names appears in the popup window below the text entry box. In my case, the options are "Run cal," "KCalc," "Local Network Browsing," "Add Locale," or "OpenOffice.org 3.1 Calc." Simply click on the one you want, or alternatively, hit the Tab key, and use the cursor arrow keys to select the item you want.



Similarly, typing in the full text of "synaptic" brings up a list of everything I can do that involves the word "synaptic," as displayed in the screen capture below. It gives me the options "Run synaptic," "Synaptics TouchPad," "Synaptic Package Manager," and "Synaptic Repair Tool."

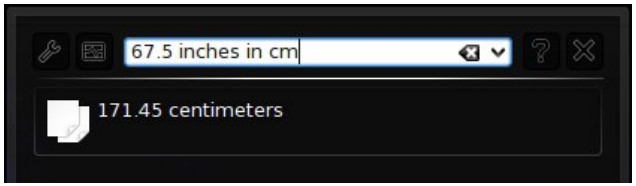


Krunner doesn't stop there. One of the plugins that is installed, and that you can select, is to use Krunner as an impromptu calculator, without having to launch a separate calculator application. Entering "=sqrt(2)" in the text entry box immediately displays the results in the popup window below, with 16 digits of precision after the decimal point. Similarly, you can put in much more complex math equations, using parenthesis, brackets, and braces to help you set the proper order of operations.



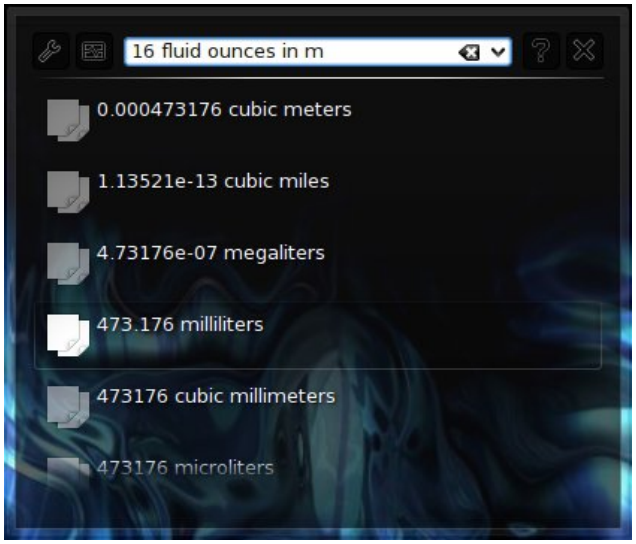
But wait! There's more! Krunner can also convert units of measure on the fly. If I type in my height in inches, Krunner immediately displays my height in meters.

If I want another measurement other than meters, all I have to do is tell Krunner what unit of measure I want it to use. In the example below, I tell Krunner to convert my height into centimeters, by entering "67.5 inches in cm" in the text entry box.



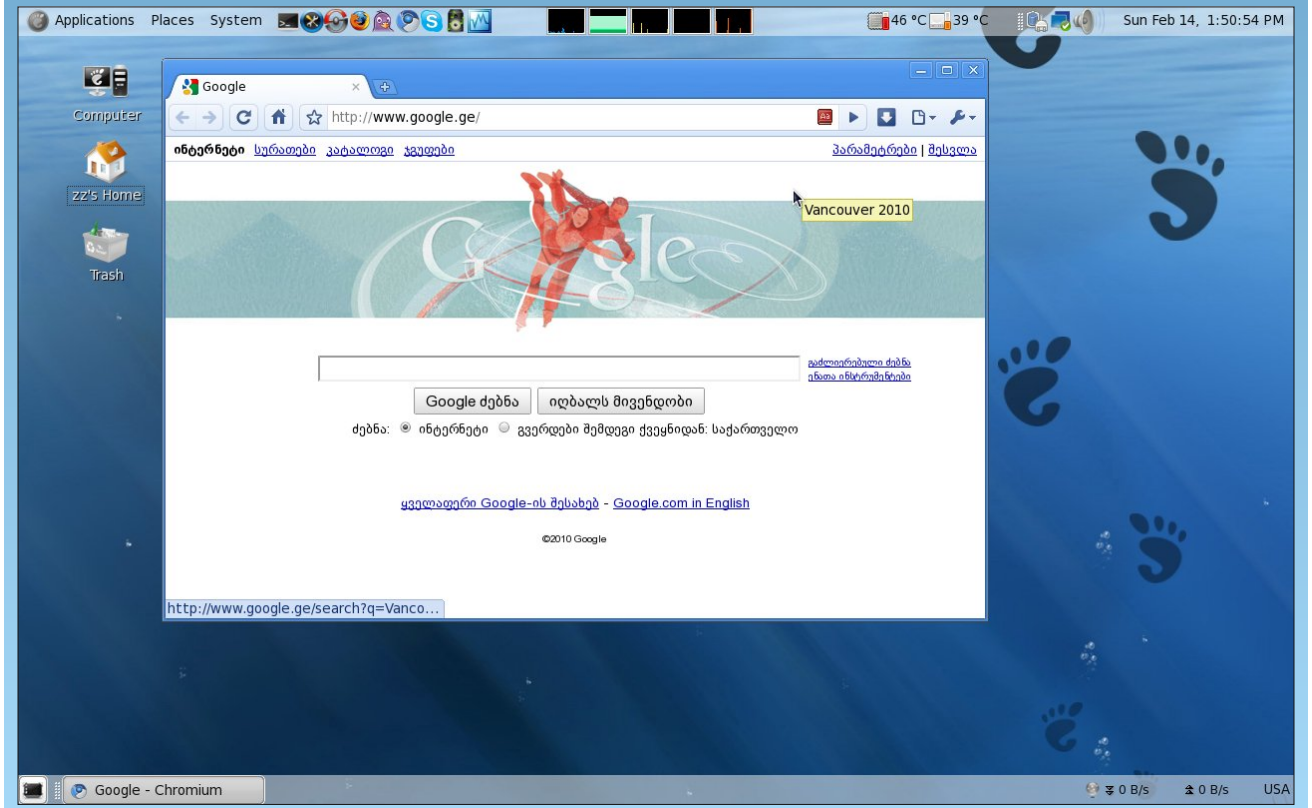
Similarly, if I type in "16 fluid ounces in m" Krunner will display ***all*** the possible measurement equivalents in all the liquid units of measure that start with "m." If I do the same type of operation with linear measurements, Krunner will display all the possible measurement equivalents in all the linear measurements that start with "m." (I found out that I am just a bit more than 0.001 miles tall!)

I have only began to barely scrape the surface of all the new functionality that the KDE developers have built into the new Krunner in KDE 4 SC. For example, you can directly enter bash commands directly into the text edit box, and they will be



executed, without you having to open a separate terminal window. Or simply enter a web address, and the specified page will open in your default web browser. You owe it to yourself to further explore the other options. But, as you can see, Krunner has grown up and has developed some new "super powers" that are easily accessible to you, the user.

Screenshot Showcase



Uncle.zz, running Gnome, posted February 14, 2010

Secure Passwords, Made Easy

by Paul Arnote (parnote)

How many times have you needed to come up with a secure password, but were stymied by the request? How many of you use the same password – or small group of passwords – over and over again? When you think about it, it really doesn't sound so "secure," now does it?

Recently, in the U.S. media, there has been lots of talk about how many people have insecure passwords. According to media accounts, some of the most "popular" passwords are "abc123," "123456," and even the word "password" being used as the password. Those who want to gain access to your data know what the most popular passwords are, so it is up to you to make access to your data difficult by picking a secure password.

In the September, 2009 issue of the PCLinuxOS Magazine, I showed you how to use openssl to help create secure passwords. But that isn't the only means available to you. I have come up with another way, one you can carry with you (in a way) on your USB flash drive, and no one would be any wiser were they to "intercept" your USB flash drive.

Many people carry USB flash drives with them. Personally, I carry multiple USB flash drives with me. One thing I like to carry are MP3 files of my favorite music and artists. I can plug it in to virtually any computer near me and listen to my favorite songs, without any difficulty. Now, I'm recommending using MP3 files as the basis for the secure password, but you can use virtually any file or file type you want on your USB flash drive.

MP3 files? For secure passwords? No, I haven't lost my mind, as you will see.

One thing you can do with those MP3 files (or virtually any other file) is create MD5 checksums, and store those MD5 checksums in files on your USB flash drive.

First of all, it's easy to create MD5 checksums. Open a terminal session, and at the command line, type the following:

md5sum [path/to/file/filename] >filename.md5

What we have done is tell the program, md5sum, to create an MD5 checksum from the file at the specified location, and to store that MD5 checksum data in the specified file, with the md5 file extension, by redirecting the output of md5sum.

Now that we have our MD5 checksum, it's time to create our "secure password." This method, by the way, works best if you have many MP3 files, along with the *.md5 files for each, stored on your USB flash drive. If all the MP3 files have corresponding MD5 files, then nothing will stand out as "weird" with only one or two MP3 files having MD5 checksum files.

Pick your favorite MP3 file from all the ones you have stored on your USB flash drive. You can just use the MD5 checksum of your favorite MP3 file, or a portion of it, as your secure password, and you will have already picked a secure password. But, you can take the security to a higher level, by applying patterns to how you apply the MD5 checksum – patterns that only you know, and that you can change at your will.

For example, you can decide that you will only use 10 characters for your secure password, and you can decide what pattern of characters to "extract" from the MD5 checksum of your favorite MP3 file. Perhaps you may decide that you want to use the first two characters, skip the third, use the fourth and fifth characters, skip the sixth character, use the seventh and eighth characters, skip the ninth character, use the tenth and eleventh character, skip the twelfth character, and use the thirteenth and fourteenth character. Now, you have a 10 character password that is always at your disposal, available to you in the event that you forget it, and one that is safe and secure – even if your USB flash drive should fall into the hands of an "unsavory" individual. All you have to do is remember the pattern.

By no means is the pattern example I give the only one. Be creative, and make up your own pattern. Here are some more pattern examples that I have thought of:

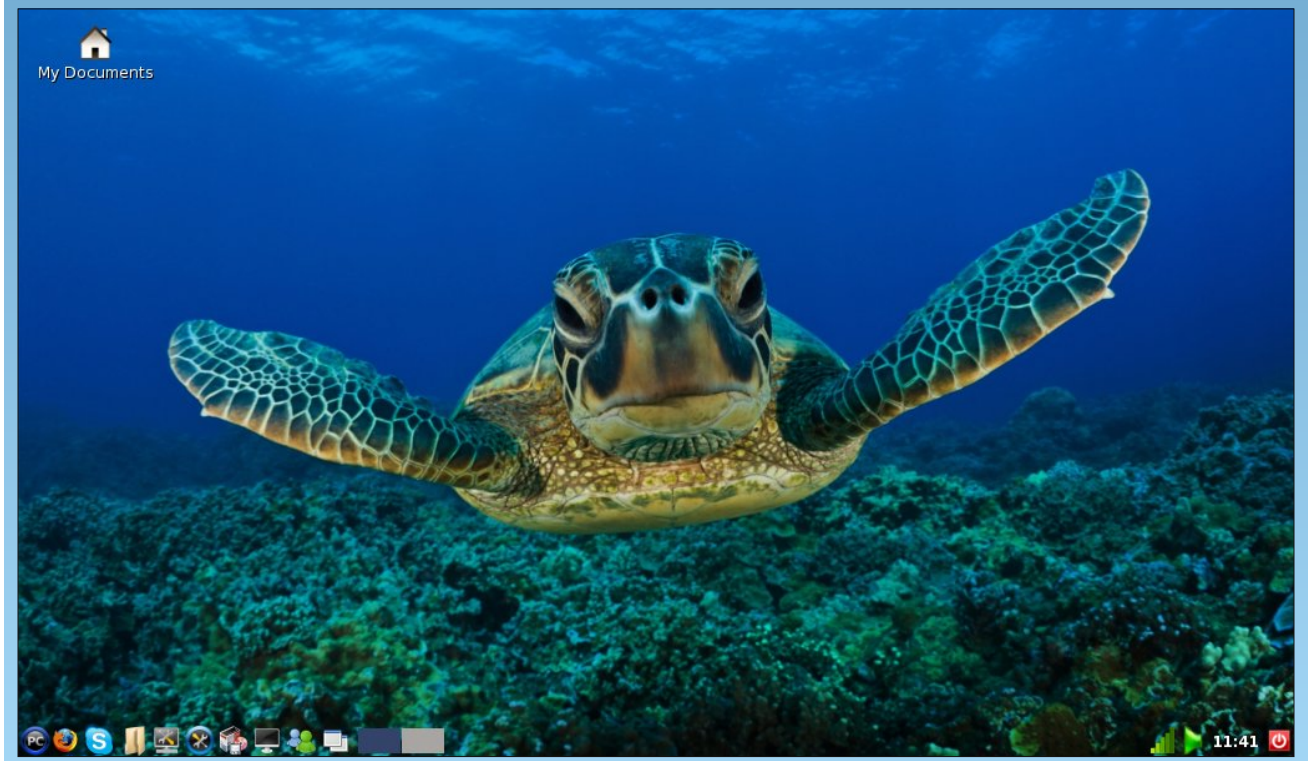
* Use the second through the eight characters of the MD5 checksum, followed by the first two characters of each word in the name of the MP3 file it represents, followed by the fourth and third characters from the end of the MD5 checksum.

* Use the first word of the title of the MP3 file, followed by the last eight characters of the MD5 checksum for that file.

* Use the abbreviation of the name of the group/singer, followed by the fourth through the eighth characters of the MD5 checksum, followed by the first letter from each word in the title of the MP3 file.

See how easy it is to create your new secure password? We all want our systems and our data to be safe, and the front line way to do so is to create a secure password that would be difficult for any hacker to hack. We want it so that if he comes across our data, the password is so secure that he just moves on to a different victim – a victim who doesn't have a secure password. And if/when "cloud computing" comes along, having a secure password is going to be even more important.

Screenshot Showcase



coffeetime, running LXDE, posted February 14, 2010

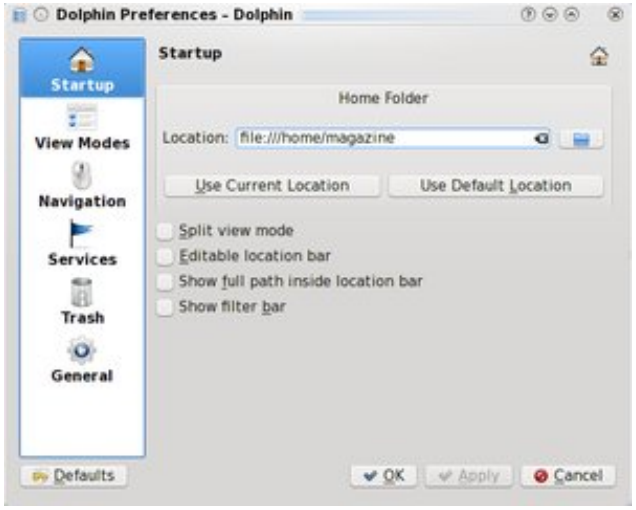
KDE 4: A Brief Look at Configuring Dolphin

by Andrew Strick (Stricktoo)

In this article, I will take a very cursory look at tinkering with the settings in Dolphin, the default file browser for KDE 4. As a caveat, I do mean *cursory*. I will not be going into much depth because most of the options are quite straight forward, which makes explaining them more confusing than helpful. If you really want to get to know Dolphin, your best bet is to simply start playing around with various settings until you're satisfied with the result. So, with that in mind, let's get started!

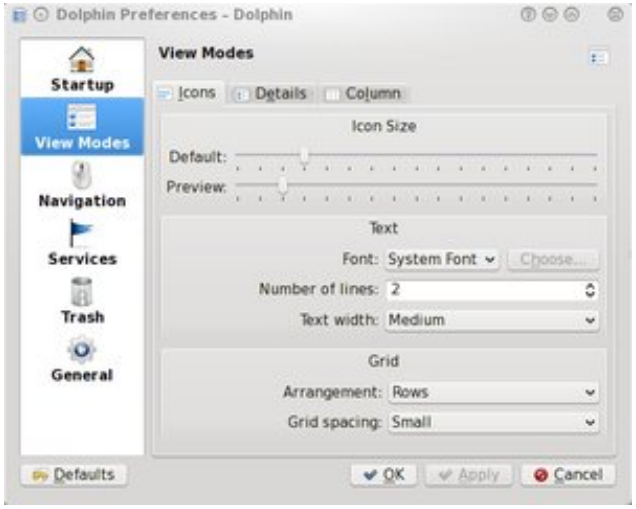
The first thing that we'll need to do is open the Settings dialog. We can do this by clicking on **Settings > Configure Dolphin** in the menu bar.

This will open the Settings dialog, specifically the Start up tab, where you can modify the way Dolphin looks when first launched. By default, Dolphin displays your Home (/home/[username]) folder. This can be changed by editing the location displayed in the **Home Folder** text box.

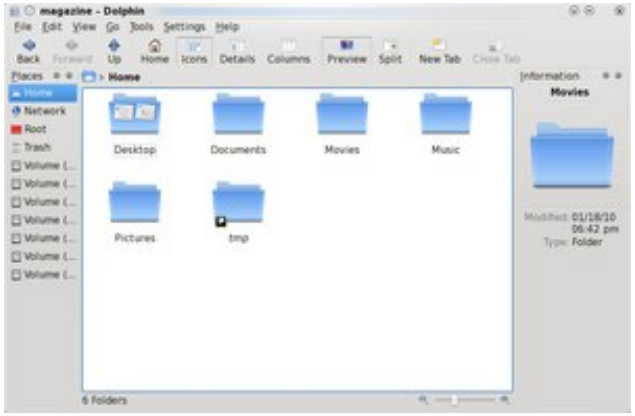


If you check **Split View Mode**, Dolphin will launch with two panes displayed. **Editable Navigation Bar** switches the navigation bar from navigation mode to the more traditional browser-style textbox. **Show full path inside location bar** causes the navigation bar (in navigation mode) to display "/»Home»[username]" instead of simply "Home."

The next option, **View Modes**, is where you can tweak the way Dolphin displays your files. The settings for each viewing mode (Icon Mode, Column Mode, and Detail Mode) are on individual tabs. Each viewing mode has its own set of **Icon Size** sliders.



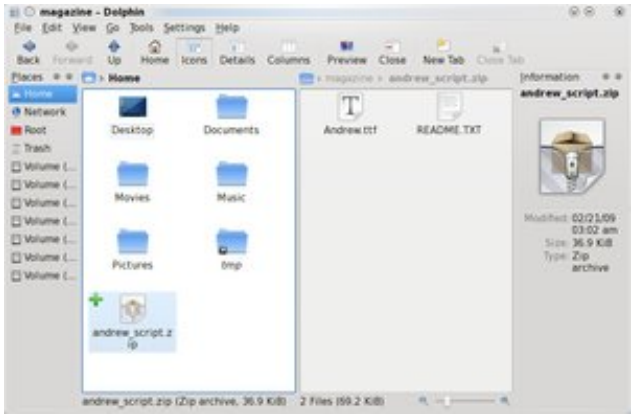
The first slider, **Default**, increases or decreases the size of normal Dolphin icons. The **Previews** slider has the same effect, but on previews. That is, you won't notice any change unless you toggle previews on via that **Previews** button on the main toolbar.

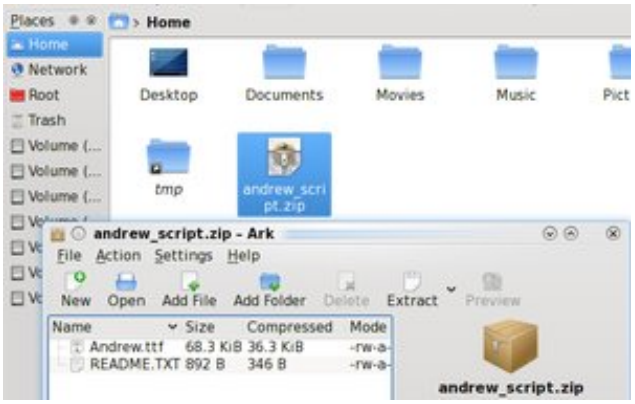


On the **Icon Mode** tab, the **Text** area contains allows you to select a specific font for icon names. **Number of lines** refers to the number of lines onto which icon names will break, if they are too long to fit in the text area. You can set the size of that area using the **Text width** option. In the **Grid** area, the arrangement setting gives you the option to have Dolphin display icons in columns instead of rows, and **Grid Spacing** controls the amount of space between icons.

The **Navigation** tab only has four options. The first two control allow you to toggle between single and double clicking to open files.

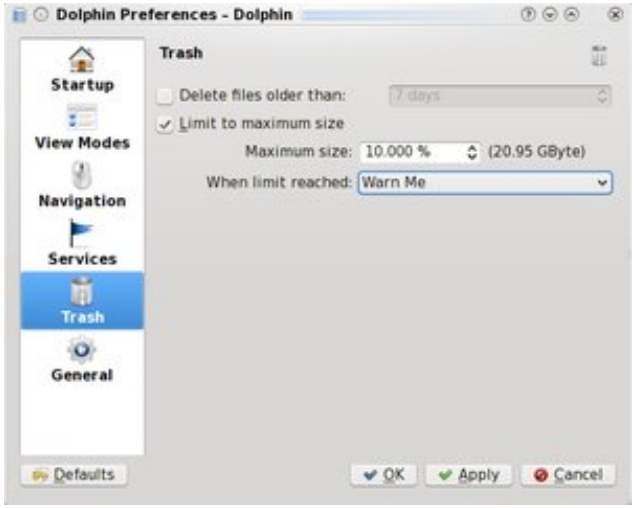
If you select **Open Archives as Folders**, clicking on an archive (files with extensions like ".tgz" or ".zip") will open that archive in Dolphin instead of Ark. As for **Open Folders During Drag Operations...** Well, I haven't been able to discover what exactly this option does.





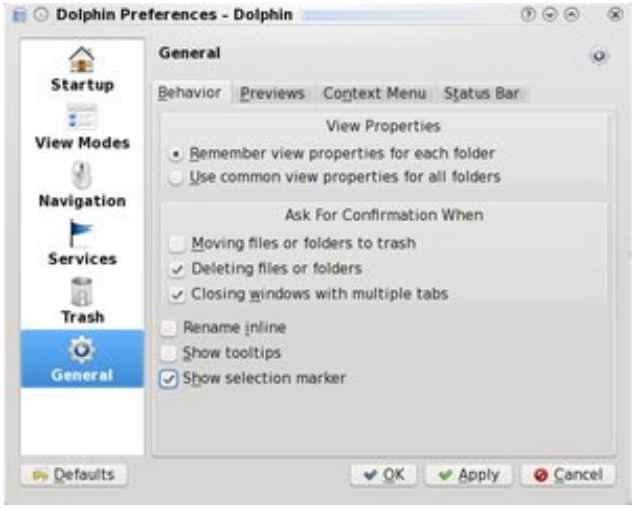
Next, the **Services** tab is essentially a list of every option that might appear in a context menu (e.g. Right Click » Open With). If you have unused options cluttering up your context menus, you can come here to deselect them. And no, Dolphin unfortunately does not (yet) have a method for adding options to this list.

The **Trash** tab contains options for controlling the size of your Trash folder. You can set Dolphin to automatically permanently delete files that are older than a specific number of days, as well as limit the size of your trash folder (as a percentage of your total /home partition size) and tell Dolphin what to do when you hit that limit.

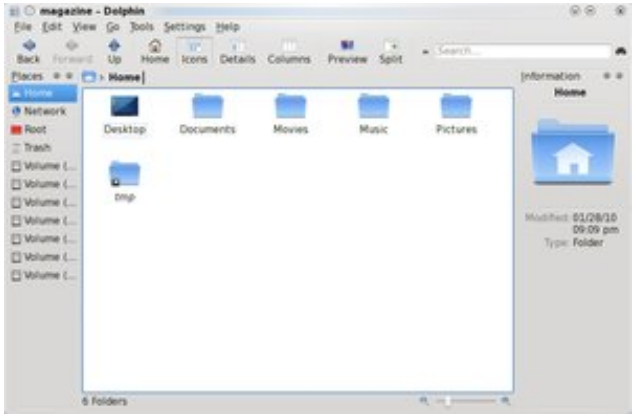


And finally, there's the **General** tab, which contains a host of miscellaneous options. Most are clearly labled, so I won't muddle things by trying to to explain them, but there are two things that I would like to point out. First, the **Rename Inline** option under the **Behavior** tab is incredibly useful. If you enable it, file extensions will no longer be automatically selected when you rename them. This is probably seems fairly inconsequential, but it does come in handy when you are attempting to rename a large number of files at once, because you can simply type out the new name instead of having to

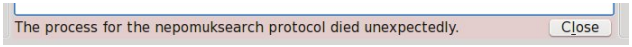
scroll to the beginning or end of the old name to avoid deleting the extension. Second, enabling a file type in the **Previews** tab will cause Dolphin to automatically enable Previews whenever you open a directory containing that type of file. For example, if you check “JPEG”, whenever you open a folder that has JPEG images in it Dolphin will toggle Previews on (and the Preview button on the main toolbar will be depressed).



One last thing that I would like to do is explain the search function in Dolphin. If you right click on the main toolbar, or go to **Settings > Toolbars**, you can elect to display the Search toolbar next to the main toolbar.



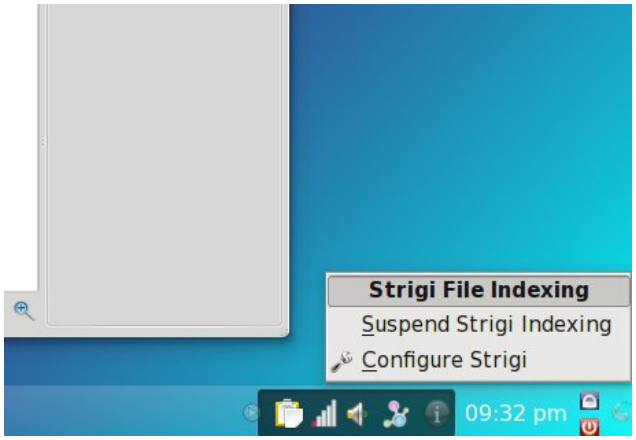
However, if you attempt to search for a file, you'll receive an error message telling you that nepomuksearch (the KDE 4 indexing service) failed.



This is because you first need to enable indexing. And to do this, you need to go to **Configure Your Desktop > Advanced User Settings > Desktop Search** and enable both Nepomuk and Strigi.

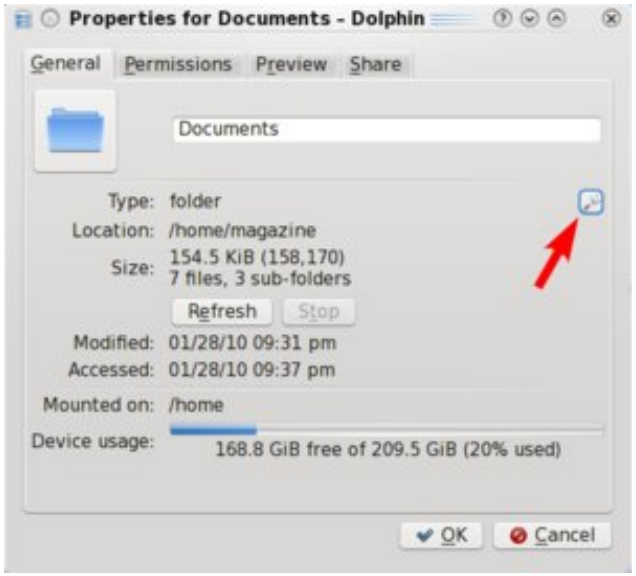
Once enabled, Nepomuk and Strigi will begin to index your home folder, and you will be able to use the search bar to locate files.

WARNING: indexing can take time. I would suggest going to the **Advanced Settings** tab and deselecting any directories that you do not need index, especially if they contain large numbers of image or video files. Also, you can pause indexing by right clicking on the Nepmuk icon in the system tray and clicking **Suspend Strigi Indexing**.

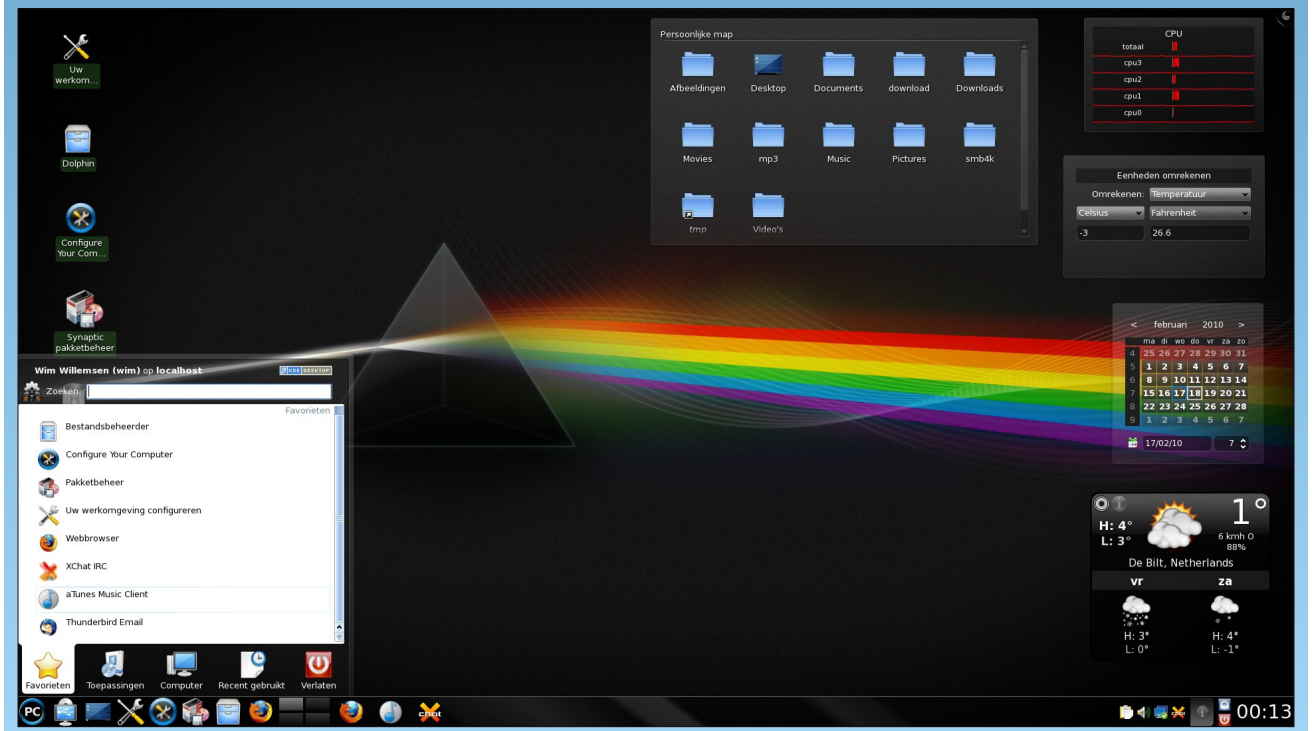


While this article was very brief, I hope that it helps to make the transition to Dolphin a little bit smoother. But if in the end you simply cannot get used to Dolphin, never fear: Konqueror can still be set as the default file browser in KDE 4. Simply right click on any folder, select “Properties” and click on the small wrench icon on the right side of the screen. This will open the file properties dialog for directories. Move Konqueror to the top of the list and hit “OK.” It's just that simple.

(See images on next page).



Screenshot Showcase



JohnW, running KDE 4, posted February 17, 2010

Forum Foibles: Limericks

Texstar wanted to make an ISO
But found he was short on cash flow
He took the risk
And made the disc
And we are glad that he did so

MeeMaw contemplating her desktop
Said my artwork is a grotesque flop
She bought a new drawing tool
Thought it so cool
And now her creations never do stop

AndrzejL kept making those tuts
Tho' everyone thought he was nuts
When he filmed KD4
Folks all asked for more
And now just look how he struts

Panote said at last I'll quit smokin'
Everyone thought he was jokin'
Not one more puff
Enough is enough
Hear me he cried I have spoken

Neal hails from the state of Tennessee
Where he's always drinking cof-fee
From the rim of his cup
He says what's up
Come along now and sip with me

There once was a Montanan so noble
A moderator of the forum so global
Members caught in the lurch
He'd tell them Go Search
You guessed it - that's our Joble

Gseaman he went to Seattle
To a conference all herded like cattle
Thought he'd learn about Tux
And a bit more Linux
Turned out just a big bunch of prattle

Old Johnboy from the Isle of Erin
Told the forum green to be wearin'
They took his advice
And said that's nice
Where's the whiskey you're sharin'

There was a grnich who could boast
Sandbox topics I've started the most
Whether a Cocoon Egg Shower
Or something more dour
About it I'm ready to post

A mod by the name of old-polack-ee
For a joke said the forum I'll hack-ee
He read the How To
And when he was through
Decided it would be kind of tack-ee

Sammy2fish obsessed with the weather
Saw snow coming down like a feather
Cloudy windy or rainy
It don't take much brainy
To know we're all in it together

There was an artist named Linuzoid
Making wallpaper he really enjoyed
Come visit my gallery
I get no salary
It's just how I'm employed

Rudge being concerned with his trash
Made a desktop folder in a flash
I'll show others the trick
I'll even post a pic
And hope their systems don't crash

Georgetoon in need of a new box
Went looking for something that rocks
Through LinPC he did dig
Finally got a new rig
Weric had one in his stocks

HootieGibbon getting out of his cage
Said I think I will go on the stage
I'll dress up like Tux
And make big bucks
Being penned up is an outrage

Coffeetime plays games day and night
A joystick is his delight
You should see my technique
It is quite unique
Gaming my passions ignite

Wildman went to the Sandbox to find
Ideas to stimulate his mind
He read and read
And then he said
I found nothing there of the kind

Scoundrel said it's time to be nice
I'll do it once - maybe twice
He tried and tried
And then he cried
It's really not worth the price

Weirdwolf remarked in his stride
Deep thoughts to the forum I'll provide
But when he came 'round
No thoughts were found
Alas he said I tried

A member by the name of Blackbird
Was rumored to be quite a nerd
When he heard those views
He said that's news
And of course it's utterly absurd

Ms_meme really does like to tease
She does it in different degrees
When commanded to quit
The more she'd commit
And said I'll do as I please

Bones113 playing on his guitar
Said I'm strumming this for Texstar
For him I can sing
Just any old thing
You know he's a bit bizarre

Sproggy was in a bit of a stew
Oh dear whatever shall I do
In all of the rush
I've lost my paint brush
My artwork is looking askew

A gentleman from old Mexico
Sat staring at the end of his toe
He gave a hoot
Said look on my boot
I do believe it's a Crow

Pirate we all would agree
Desktops a plenty has he
Now he's got old cars
It must be in the stars
Would someone explain that to me

In the forum you'll find jaydot
He thinks his posts are hot
But if you inspect 'em
You'll have to reject 'em
As nothing but diddly squat

Whenever you're in a fix
You'll be answered by member T-6
Whatever's involved
It will be solved
That's how he gets his kicks

In the forum if you find nothing new
And all posts you've read twice thru
Put your boredom aside
Some HELP do provide
YouCanToo does it so can you

MeeMaw painted a heart bright pink
Archie said here's what I think
Tho' the colors are true
I prefer hearts of blue
And he gave a sly little wink

Coolbreeze we cannot deny
Does post a plentiful supply
In that thread or this
He's never remiss
To make some silly reply

Texstar said I'm on a crusade
To build the best system made
I toil day and night
To make it just right
How come I'm so underpaid

Mokmeister bragged to his spouse
I am the man of the house
She saw his avatar
And said Har Dee Har
You look more like a mouse

Timeth made a sensation
Dobie the bull is his creation
It is really quite clever
A lasting endeavor
He deserves all our admiration

I really don't mean to defy
I'm only giving a short reply
But when you say my name
If it's all the same
Please pronounce it GuynotGuy

A member by the name of davecs
While working on PCLOS projects
Said there's really nothing to it
I just have to do it
That's what Tex expects

O-P and a wall that was plastered
For lunch he decided he'd mastered
With his fork and a chew
He spit out said euww
You can't eat wall you O-P

rudge

In this forum posters should fear
The old-polack that almost lives here
If they post something crude
Or exceptionally rude
Their postings just might disappear

old-polack

Old-polack has sent me a sign
For my posts there might be a fine
My limerick was wrong
Apologies in song
So long as you never delete mine

rudge

There once was two girls and a cup
My friend said watch this close up
Then Tex chickend out
But ms_meme held stout
And ended the rhyme with a Yup

Texstar/ old-polack

Sometimes I'm not good with a rhyme
Other times I end up fine
A limerick is fun
When its properly done
But that would just take too much time

athaki

Meme sang a song or two
While singing she doodled and drew
When her pics were done
And she'd had her fun
She posted quite a few

We had fun fanfare and fright
We recognized ourselves a mite
It was us she was drawing
She had found her calling
And had us pegged alright

One song two three then four
I sat back and waited for more
In each magazine she brings
Joy as she sings
She says she has more in store

Joble

A member by the name of F.Luent
About English he was pursuant
It never makes much sense
No matter the tense
I never will become fluent

MaddogF16 is an expert on hardware
For it he has quite a flair
He gets out the quirks
And then it all works
You'll not be left in dispair

Limericks I have written a lot
Then travis I almost forgot
But I'm not to blame
Just look at his name
It ends in *N aught aught aught*

Archie told ms_meme to sing
The magazine some fun for to bring
She cackled and cluckled
The forum they chuckled
And said what an old ding-a-ling

KDE 4: Okular Does More Than Just PDFs

by Paul Arnote (parnote)

First came Xpdf, the PDF reader for X-Windows. Then, Kpdf was created for KDE, based off of the efforts of Xpdf. Then, born from Kpdf and the 2005 Google Summer of Code, came Okular, the new universal document reader for KDE 4 SC. During the process, Okular became a lot smarter, and learned how to display a lot more than just PDF files.

Replacing Kpdf, Okular can now view not only PDF files, but also most image files, OpenOffice.org Writer (*.odt) files, postscript files, compiled HTML Help files, EPub e-book files, faxes, and other formats. The chart at right, from the KDE 4 Userbase, fills in the details of supported file types that Okular can view.

Remarkably, Okular cannot currently handle displaying simple TXT files, HTML files, or Microsoft Office files. Okular is only capable of displaying OpenOffice.org Writer (*.odt) files, but currently chokes on any of the other OpenOffice.org file formats for the rest of the OpenOffice.org office suite. However, any shortcomings that Okular may currently have with displaying various file formats are likely to be filled in by developers writing additional document format handlers for Okular.

Okular keeps all the same power and abilities as Kpdf, and builds on it by increasing the number of other document files it can handle or display. The first screen shot on the next page shows Okular doing what it does best: displaying a PDF file. It can also annotate PDF files, and when you save the annotated document, any other user can open the

Document Format Handlers

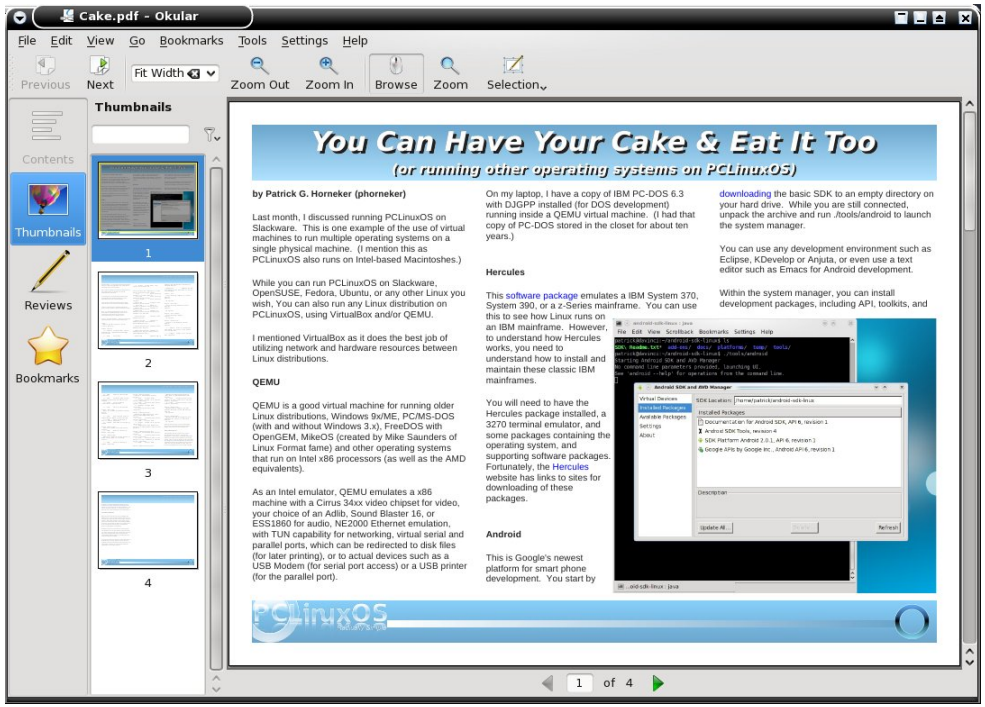
This page always refers to the stable series of Okular (currently KDE 4.2.x).

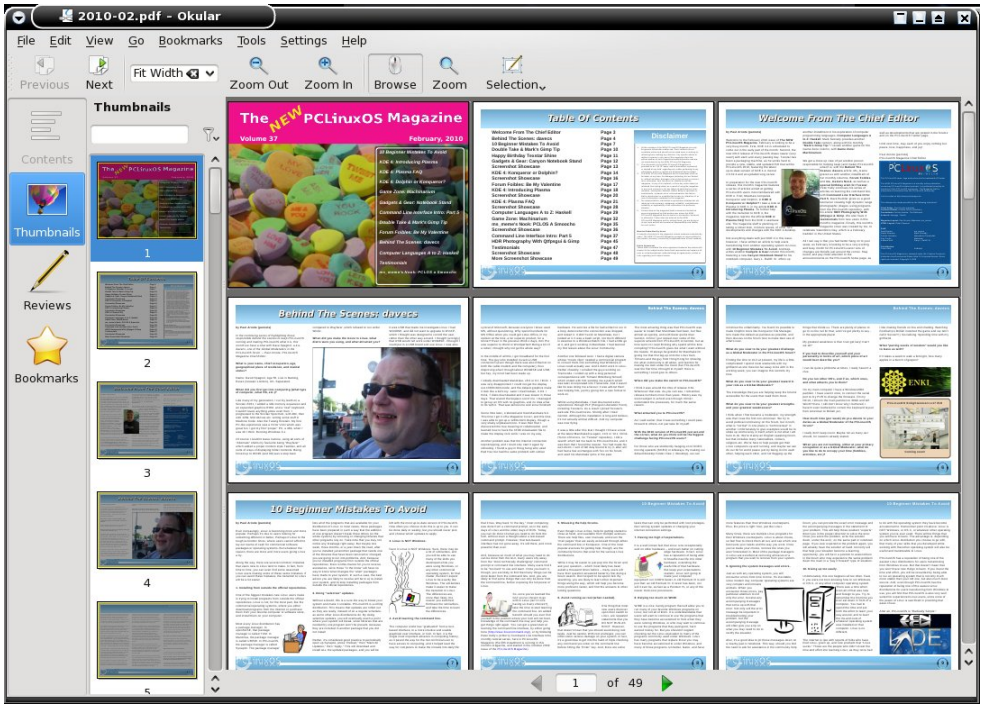
Features/Formats	PDF	PS	Tiff	CHM	DjVu	Images	DVI	XPS	ODT	Fiction Book	Comic Book	Plucker	EPub	Fax
Main library used	**poppler**	**libspectre**	**libTIFF**	**chmlib**	**DjVuLibre**								**epub**	
Loading	✪	✪	✪	✪	✪	✪	✪	✪	🕒	🕒	✪	🕒	✪	✪
Rendering	✪	✪	✪	✪	✪	✪	✪	🕒	✪	✪	✪	🕒	✪	✪
Threaded rendering	✪	✪	✪	✪	✪	✪	×	(Qt >= 4.4.2)	×	×	✪	×	×	✪
Document information	✪	✪	🕒	🕒	✪	×	✪	✪	✪	✪		🕒	✪	×
TOC	✪			✪	✪		✪	✪	✪	✪		×	✪	
Document fonts	✪						×		×	×		×	×	
Text extraction	✪			🕒	🕒		✪	✪	✪	✪		×	✪	
Links	✪			✪	✪		✪	×	✪	✪		×	✪	
Paper Size		×					×	×				×		
Printing	✪	✪	✪	×	✪	✪	✪	✪	✪	✪	✪	×	✪	✪
Text Exporting	✪			×	×		×	✪	✪	✪		✪	✪	

Other Features

Annotations	🕒				🕒			×	🕒	×		×		
Forms	🕒								×	×		×		
Inverse search	(pdfsync)						✪			×		×		
DRM	✪							×		×		×		
Embedded files	✪							×		×		×		
Sounds	✪							×	×	×		×	×	
Movies	🕒							×	×	×		×	×	

Chart Key: ✪ Working Feature 🕒 Work In Progress × Unavailable





annotated file using Okular. It can also bookmark your PDF files, which is a handy feature when reading PDF versions of e-books, allowing you to save your current place in the book and to continue reading from where you left off during a previous reading session.

Another cool feature of Okular is it's ability to display multiple pages. You can select for Okular to display "facing pages" in a PDF document, or provide an overview of all the pages (as in the screen capture above, right, of the February 2010 issue of the PCLinuxOS Magazine), to allow you to see how the whole document flows and appears together.

Overall, Okular represents a giant leap forward from Kpdf, which could only display PDF and PS files. With the additional capability of displaying more types of files, Okular is poised to become the one-stop document viewing application in not only Linux, but also every other platform supported by KDE 4 SC. Already made more powerful by it's increased capabilities, Okular will become an unstoppable force in document viewing, once other document format handlers are in place, and a greater variety of file types are supported, especially those file formats specified earlier.

Visit Us On IRC

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

Music Notation Software in PCLinuxOS

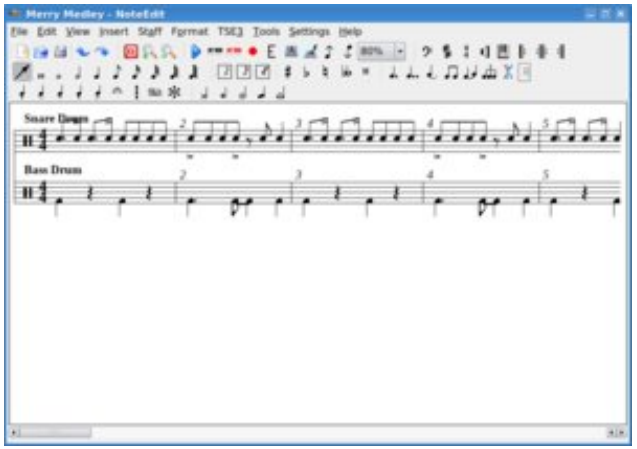
by Galen Seaman (gseaman)

History

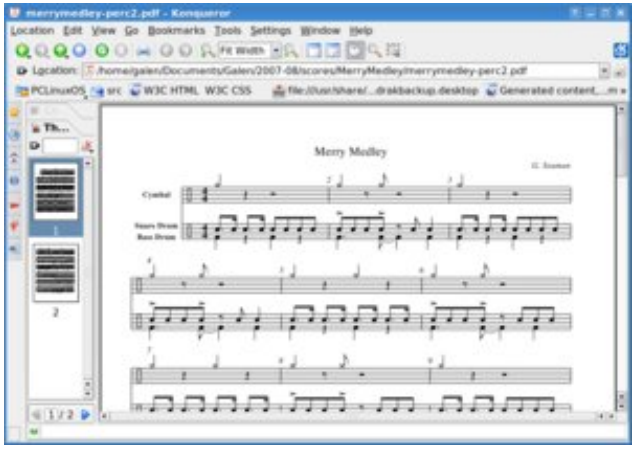
The first computer music notation program that I tried was **Finale**. It was exciting to be able to create printed notation directly with your computer, but, at that time, Finale required pages of cheat sheets to recall all of the keyboard shortcuts to use it. I know at least a few people who purchased Finale, attempted to learn it, and then decided it was much easier to write their scores by hand. Then Sibelius came along with a much easier to use interface and the competition caused Finale to make major improvements.

My first purchase of music notation software was **Sibelius**. Only one month after I purchased Sibelius for about $300 USD, a new version came out and I was not even qualified for a discounted upgrade. So to get the improved Sibelius I would have been out $600 in two months! I was just starting to notice Linux and started looking for open source alternatives. At that time, NoteEdit was very crude, but I learned to use it, hoping that it would develop into a real alternative. I could create a music part, export to the abc music format, edit the text and then print out the parts. It worked, but it was time consuming, error-prone and not a process I could ever hope to teach my students to use.

NoteEdit was the only usable graphical program on Linux for many years. The original developer, Joerg Anders, the developer of the much improved NtEd, stopped developing it when he read comments on the Linux Audio Developers' (LAD) list where a commenter requested a community drive to get a



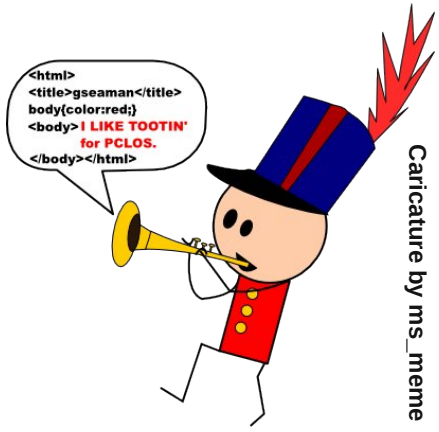
Creating a percussion part in NoteEdit.



1) Drum score created in NoteEdit, 2) exported to abc music, 3) used abcm2ps to create postscript, then 4) used ps2pdf to create the pdf file.

Linux version of one of the proprietary windows notation programs, without even acknowledging the existence or potential of NoteEdit becoming a reasonable option. Apparently, no one on the LAD list had faith that NoteEdit would ever reached maturity.

After a couple years, new developers came along and brought many improvements including direct printing through Lilypond or abcm2ps. This was a big development, bringing the possibility of doing all of your notation needs directly from the program and not having to resort to the command line for the final step. But NoteEdit had major structural flaws and lots of bugs, which made it very difficult to program. The developers decided to start over. They are now developing a program called Canorus. It looks good, but is still less capable than NoteEdit after over two years of work, and progress is slow. NoteEdit is built with the kdelibs3 so, unless it can somehow be built with static dependencies, it will soon disappear from PCLinuxOS.



Caricature by ms_meme

As a music teacher, my needs for music notation have included full band, orchestra, choral and jazz ensemble scores. I occasionally could have benefited from guitar tabs as well. Until recently, much of this was still too difficult with Linux. The good news is that music notation has matured to the point that I can do all of these

things reasonably well and without excessive effort! We now have many great programs, and with the current rate of progress, we will soon have the best programs of any OS.

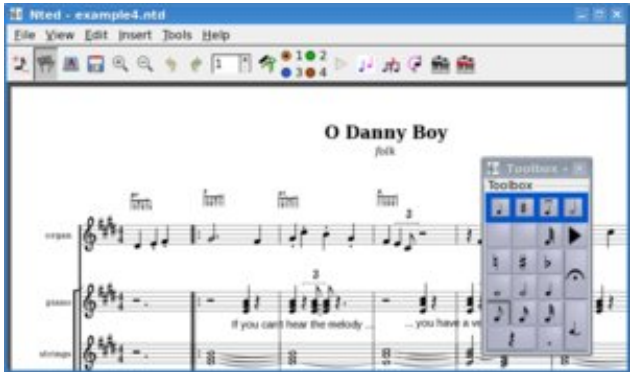
Music notation can be very complex, so I am not going to attempt to cover all of the features of these programs. Instead, I will just present the basic features and status of these programs so that readers will be aware of what is now available.

There are currently two excellent, full-featured graphical notation programs available: **MuseScore** and **NtEd**.

MuseScore is very powerful, easy to learn, has excellent midi playback, is stable, cross-platform, and the development team is making fast progress. MuseScore is released under the GPL and available for Linux, Windows and the OSX. The lead developer, Werner Schweer, also is the original author of Muse, a great midi sequencer program.

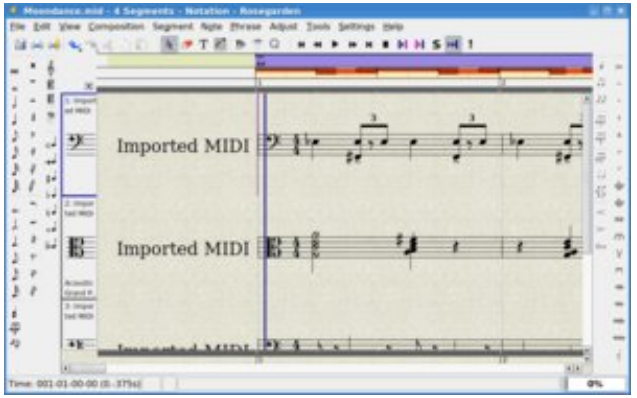
Editing a Sample score in MuseScore.

MuseScore is the one I will likely use the most, as I can teach it to students at school or have them install it at home. This is also the only notation program that currently allows manual adjustments of objects on the music. This is not needed most of the time, but it can really come in handy to clean up the look of a score. Much of my arrangements are for jazz/swing and midi playback with 'straight' eight notes really sounds bad. I just found out the next release has the option of exporting midi files with a 'swing' feel! Many other major improvements are being discussed and are likely on the way.



Editing a sample file in NtEd.

NtEd is also a very powerful notation program. Joerg Anders, original author of NoteEdit, has returned with great success with this new program. I can only assume from the speed at which Mr. Anders has added features and the compactness of the codebase, that NtEd must be very well designed. The documentation for this program is exemplary in every way. This is a first-rate program and I hope that Mr. Anders will soften his position on cross-platform development. I believe in free software and will not ever go back to using Microsoft software, but as a teacher, I am unable to tell my school and my students they must install Linux. So, I am limited in using NtEd for my own use, and not able to use it for teaching.



Rosegarden's Score Editor.

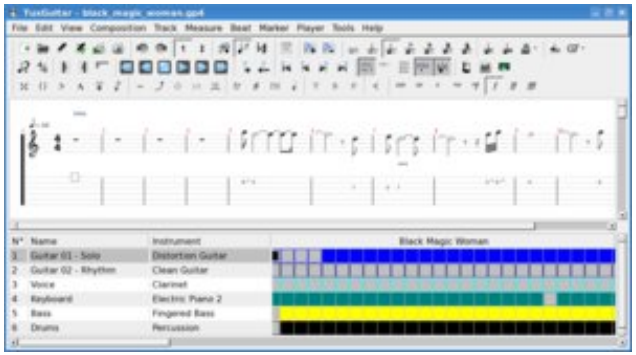
Rosegarden is a powerful tool for creating and manipulating midi files. It has the ability to be used as a music notation environment. For me, it is very difficult to get the results that you may want. It does export Lilypond files which can then be turned into .pdf files for printing. But unless you want to hand-edit very complicated files, the output is usually unsatisfactory. This is not to discredit this program, it is just designed for other purposes.

Although, **Impro-Visor** is not technically a notation program, but it can compose/generate improvised jazz solo melodies in the style a musician of your choice, which can be exported to MusicXML files and then imported into MuseScore. It will also play the improvisations with appropriate small jazz ensemble accompaniment.

Sample Impro-Visor melody.

This is an exciting new addition to PCLinuxOS. The primary goal of Impro-Visor is to teach improvisation and composition. This program has been available under the GPL for a while, but to access the source code you had to sign up for a Yahoo! mailing list account. This recently changed, and the program can be directly downloaded and now is in the PCLinuxOS repos.

TuxGuitar is the first truly successful guitar software in Linux. Previous efforts, like KGuitar, showed potential, but were limited and were never finished. TuxGuitar is a gpl'd java program that aims to be the best in class in guitar tabs and notation. Guitar players use three different ways to display music: standard notation (similar to piano music), chord diagrams (pictures of finger placement to create specific chords to be played), and tabs. Tabs are essentially the fingerings for specific notes. Well designed tabs can be a great benefit to learning new songs. Many of the tabs available online are ascii (text) files and do not show rhythm information. So, to make the most of Tuxguitar, you need to create



TuxGuitar with standard notation and tabs.

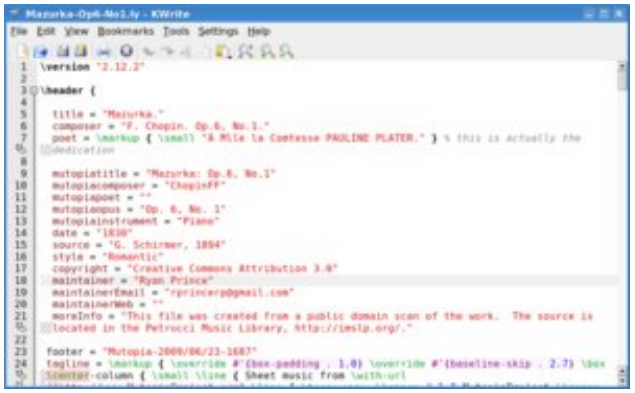
your own tabs or import PowerTab or GuitarPro tabs. (These are available in many places). Tuxguitar can include notes, chords and tabs simultaneously. It also looks and works very professionally. It is cross-platform and exports MusicXML and midi.

Denemo is designed to be a front-end to Lilypond. I don't know much about using Denemo. It was way too difficult to use for me when I first tried it, several years ago. It has had a lot of development recently, but I haven't had the motivation to tried to figure it out, with MuseScore and NtEd progressing so well.

Simple two-part Denemo score.

The power of Denemo is that it offers access to all of the features of Lilypond. For those that need the power of Lilypond, Denemo may be the best way to get started.

Lilypond is the most ambitious music notation project in existence. It attempts to automatically create the most aesthetically pleasing and accurate

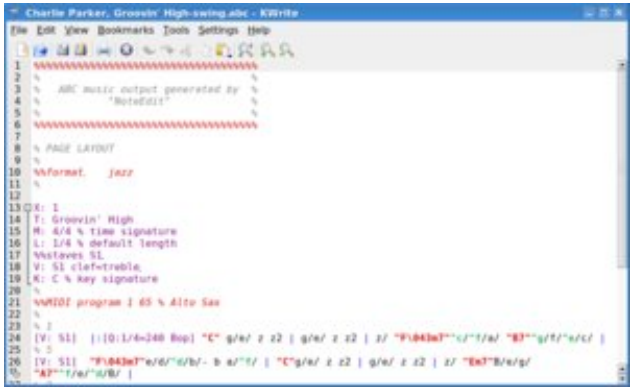


A sample lilypond text file.

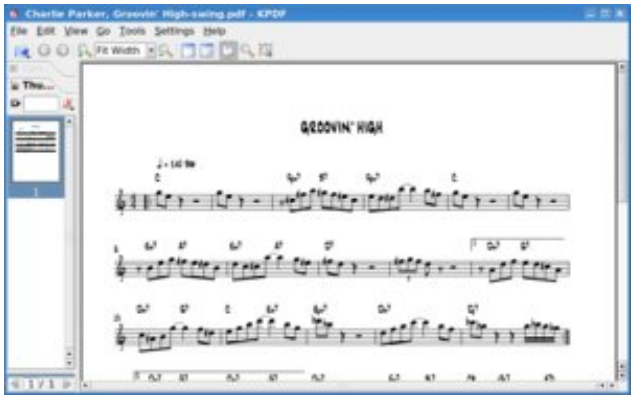
The previous text file converted to PDF using Lilypond.

music scores in every style and musical period, without any manual placement needed. Many public domain scores have been made in the lilypond format and can be downloaded from http://www.mutopiaproject.org.

abcm2ps, which stands for abc music to postscript, is the best pop/jazz music production software that I know. It is much simpler than Lilypond and because the Lilypond authors are focused on classical notation, abcm2ps often looks better with popular and jazz styles. The file format is relatively easy to read and edit directly if you want more control of the output. Unfortunately, at this time, only NoteEdit can create abcm2ps files. Although, you can create music in MuseScore, export to MusicXML files, import into NoteEdit, then export to abc music. This is only worth it if the quality of the output is significantly better the MuseScore can create by itself.

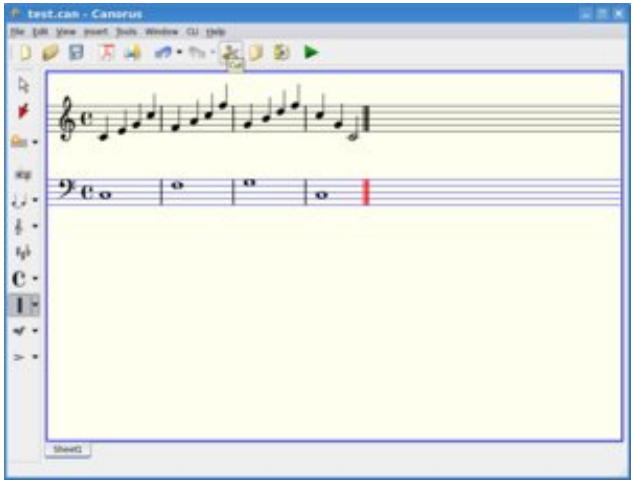


Because only abcm2ps allows you to select the font for the actual notes, it has the ability to create music that looks like professionally created jazz scores. It



A slightly modified abcm2ps file exported from NoteEdit (left). Created by abcm2ps using a jazz stylesheet, with jazz note and text fonts (above).

takes a bit of work to set it up, but once it's done, it's not to hard to repeat.



Simple example in Canorus.

Finally, **Canorus** is from the development team that kept NoteEdit alive and made some major improvements. Canorus looks good, and seems to have a sound basis, but it is not usable for anything yet. Development is very slow compared to MuseScore and NtEd. (I will not contribute a package for the PCLinuxOS repos until it is stable.)

Galen Seaman

Command Line Interface Intro: Part 6

by Peter Kelly (critter)

Globbing

What? It's an unusual word that has it's roots in the way command line interpreters used to handle things in the early days of Unix.

The bash shell recognizes certain characters as 'wild cards' that can be used to specify unknown or multiple occurrences of characters when specifying file names. We've already met some of these 'wild card' characters back in chapter 3, **'*'** and **'?'**. These are the most common ones, but groups and classes of characters can also be used. When the shell encounters these 'wild cards' in a file name, it substitutes the *meaning* of the wild card at that position, a process known variously as **"file name expansion"**, **"path name expansion"**, **"shell expansion"** or **"globbing"**, depending on how pedantic or geeky you want to be.

The following can be used:

***** means match at this position *any one or more* characters

? means match at this position *exactly one* character

Individual characters can be grouped in square brackets:

[a,f] matches either **a** or **f**
[a-m,w-z] matches only a character in the ranges **a-m** or **w-z**
[a-z,0-9] matches any lowercase letter or any digit
[!a-c] matches any character that is **not** in the range **a-c**
[^a-c] same as above

```
jane@daisy > ~ $ ls -d [mM]*
Movies/  Music/  musicfiles  mydir/  mydir1/
```

Another way of specifying groups of characters is to use **pre-defined classes**. These are groups of characters defined in the POSIX standard and the syntax is **[:class:]**.

The defined classes include:

[:alnum:] any alphanumeric character [0-9,a-z, A-Z]
[:alpha:] alphabetical characters [a-z,A-Z]
[:blank:] characters that don't print anything like spaces and tabs (also known as whitespace).
[:digit:] numeric digits [0-9]
[:punct:] punctuation characters
[:lower:] lowercase characters [a-z]
[:upper:] uppercase characters [A-Z]
[:xdigit:] any character that may form a part of a hexadecimal number [0-9,a-f,A-F]

So, **ls -d [[:upper:]]*** will find all files and directories that start with an uppercase letter.

```
jane@daisy > ~ $ ls -d *[[:digit:]]*
contacts2  mydir1/  newfile2
```

Note that two pairs of square braces are required here, one pair to define the start and end of a range and one pair to contain the class.

All of these methods may be combined to provide the file set that exactly matches your requirements. This method of file name expansion may reduce or increase the number of files to which your commands are applied.

The dot character **'.'** is not included in these expansions. Type **ls -al** in your home directory and you will see quite a few files that begin with this character. These are so called hidden files which is why we needed the **-a** option to the **ls** command to display them. The first two file names (in any directory) are the names of directories, **'.'** and **'..'**, and these refer respectively to this directory and to the parent of this directory. Why do we need these? One reason is to be able to refer to a directory without specifying its name.

cd .. takes you up a level
cd ../.. takes you up two levels and so on.

If you write a script and want to execute it, then it has to be on your **PATH** (a list of directories to be searched for executable files), or you have to supply the full absolute address for the file. Typing **./myscript** {this directory/myscript} is easier than **/home/jane/myscripts/myscript**.

For security reasons, it is inadvisable to add your home directory to your **PATH**.

If then, the dot character is not included in these expansions, how do we include hidden files if we want them to be a part of our list of files to operate on, but we don't want the two directory shortcuts **'.'** and **'..'**?

Suppose we want to rename all files in a directory with the extension '.bak', but some of these files are hidden **'dot'** files? If we try to rename all files, including those beginning with a dot, then we will include **.** and **..**, which we didn't intend (you will probably get an error).

We could make two lists of files, one of normal files and one of dot files with the two unwanted files filtered out, or we could could get the shell to do our dirty work for us. The bash shell has a lot of options and is started by default with those options set that your distribution settled on as being most useful for everyday use.

For a list of those options that are set (on) type **shopt -s**. The **shopt** command is used to display the status of **sh**ell **opt**ions. **shopt -u** lists those that are unset.

The one that we are looking for here is called **dotglob**.

Setting this option on expands the dots that signify a hidden file, but ignore the two directory shortcuts. **shopt -s dotglob** turns it on, while **shopt -u dotglob** turns it off again. Don't forget to do this or you will remain in unfamiliar territory.

To rename all of the files, we need to use a loop. I will explain the mechanism of this when we get to shell scripting – the real power of the shell. For now just follow along.

ls -al shows 2 unwanted directory shortcuts and 5 files, 2 of them hidden dot files.

```
jane@daisy > bak $ ls -al
total 8
drwxr-xr-x  2 jane jane 4096 Jan 12 13:10 ./
drwx------ 30 jane jane 4096 Jan 12 13:40 ../
-rw-r--r--  1 jane jane    0 Jan 12 13:09 a1
-rw-r--r--  1 jane jane    0 Jan 12 13:10 a2
-rw-r--r--  1 jane jane    0 Jan 12 13:10 .a3
-rw-r--r--  1 jane jane    0 Jan 12 13:10 .a4
-rw-r--r--  1 jane jane    0 Jan 12 13:10 a5
```

```
jane@daisy > bak $ ls *
a1  a2  a5
```

The **'*'** does not expand to show the hidden dot files or the directory shortcuts.

```
jane@daisy > bak $ shopt -s dotglob
jane@daisy > bak $ ls *
a1  a2  .a3  .a4  a5
```

By turning on the shell option **dotglob**, the wild card **'*'** expands into all of the files but ignores the directory shortcuts.

We now can run our loop, and a check reveals that all files have been renamed:

```
jane@daisy > bak $ for f in *; do mv $f $f.new; done
jane@daisy > bak $ ls *
a1.new  a2.new  .a3.new  .a4.new  a5.new
```

Turning off the option reverts to the normal mode of dot files being unexpanded.

```
jane@daisy > bak $ shopt -u dotglob
jane@daisy > bak $ ls *
a1.new  a2.new  a5.new
```

The bash shell expansion is not limited to file names. There are six levels of expansion recognized by bash. They are, in alphabetical order:

Arithmetic expansion.
Brace expansion.
File name expansion.
History expansion
Shell parameter expansion.
Tilde expansion.

If you are not at least aware of these, then you may find yourself inadvertently using them and then wondering why you are getting such weird results.

Arithmetic expansion. The shell can do limited integer-only arithmetic, its operators all have have different meanings under different circumstances in the shell. They are:

- **+** Addition
- **-** Subtraction
- ***** Multiplication
- **/** Division
- ****** Exponentiation (The exponentiation operator is more usually '^' however this is used for negation in the shell expansion rules.)
- **%** Modulo (remainder

The syntax is **$((expression))**. Again, note the two sets of braces. Expressions may be nested within the first pair of braces and standard operator precedence is observed. Whitespace (e.g., spaces and tabs) has no meaning within the expression.

```
jane@daisy > ~ $ echo $((3+4))
7
jane@daisy > ~ $ echo $(($((3+4))/2))
3
```

3 + 4 = 7
7 / 2 = 3 Integers only!

Tilde expansion. we may as well get this one out of the way now as the only way we are likely to use it is very simple.

We can use the tilde **'~'** as a shorthand way of referring to our home directory by using it as the first letter of an expression.

cd ~ change to our home directory

cd ~/mydir1 change to the sub-directory mydir in my home directory.

If we follow the tilde with the login name of another user then the command is relative to that users home directory. **cd ~john** change to johns home directory. No forward slash is required between the tilde and the users login name but you obviously still need the correct permissions to enter the directory if you are not the owner or a member of the directories group.

So is that all there is to tilde expansion. Of course not, this is Linux!

A system administrator might use it to assign expansions to commands by manipulating the directory stack – but you really didn't want to hear that did you?

Brace expansion is particularly good for a situation where a sequence of files or directories need to be created or listed.

The syntax is **{a,b,c}** or **{A..J}** or **{1..8}**.

For example, to create a set of directories to hold a years notes

```
jane@daisy > notes $ mkdir notes_{jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nove,dec}_2010
jane@daisy > notes $ ll
total 48
drwxr-xr-x 2 pete pete 4096 Jan 12 16:51 notes_apr_2010/
drwxr-xr-x 2 pete pete 4096 Jan 12 16:51 notes_aug_2010/
drwxr-xr-x 2 pete pete 4096 Jan 12 16:51 notes_dec_2010/
drwxr-xr-x 2 pete pete 4096 Jan 12 16:51 notes_feb_2010/
drwxr-xr-x 2 pete pete 4096 Jan 12 16:51 notes_jan_2010/
drwxr-xr-x 2 pete pete 4096 Jan 12 16:51 notes_jul_2010/
drwxr-xr-x 2 pete pete 4096 Jan 12 16:51 notes_jun_2010/
drwxr-xr-x 2 pete pete 4096 Jan 12 16:51 notes_mar_2010/
drwxr-xr-x 2 pete pete 4096 Jan 12 16:51 notes_may_2010/
drwxr-xr-x 2 pete pete 4096 Jan 12 16:51 notes_nove_2010/
drwxr-xr-x 2 pete pete 4096 Jan 12 16:51 notes_oct_2010/
drwxr-xr-x 2 pete pete 4096 Jan 12 16:51 notes_sep_2010/
```

You can use brace expansion wherever you need to put a set of items into a common expression.

History expansion. When you type the command **history**, you are presented with a numbered list of previously typed commands. Entering **!number** (where **number** is the number in the list) executes that command, this is history expansion at work. There is more to it than that, but it shouldn't bother us for the moment.

Shell parameter expansion is at its most powerful when used in shell scripts, but in its simplest form, it takes whatever follows a **'$'** and expands it into its fullest form. What follows the **$** is often an **environment variable**, which may optionally be enclosed in braces i.e. **${var}**, but it can be much more.

Environment variables are names given to things that need to be remembered in your current working environment. An example of this is where you currently are in the file system. To find this out you might issue the command **pwd**. This information is stored in the environment variable named **PWD** (uppercase is customary and does help to distinguish variables from other text. Whatever the case of the name you will have to use it as Linux is case sensitive).

Typing the command **echo $PWD** uses parameter expansion to the expand the variable name **PWD** to the full path of your current directory **before** passing it to the command echo.

To see a list of environment variables and their contents that are currently set up in your environment, type the command **env**.

Some you will recognize, and you can always add your own by use of the **export** command in your **.bashrc** file.

export SCRIPTS_DIR=”/home/jane/scripts”

After that, the command **cp my_new_script $SCRIPTS_DIR** will make sure that it goes to the correct place, providing of course that it exists..

With all these different ways that the shell can interpret what you type, we need some method of controlling what gets seen, as what and when. After all, although the bash shell is very powerful, we do want to be remain in control.

Control of shell expansion is exercised through the use of quotes. In Linux you will find four different kinds of quotes in use:

1. “ ” The decorative 66 – 99 style used in word processors – these are of no interest to us.
2. ” Standard double quotes
3. ' Single quotes
4. ` Back ticks, also known as the grave accent

The last three all produce different results in the bash shell.

Double quotes tell the shell to ignore the special meaning of any characters encountered between them and to treat them exactly literally, with the exception of **$**, ` and ****. This is useful if we want to pass the shell a file name containing an unusual character or space. However, because the **$** is still interpreted, variables and arithmetic will still be expanded. But **after** expansion, everything between the quotes will be passed to the command as a single word with every character and all white space intact. The interpretation of the back tick you will see in a moment, just remember that here it is preserved. The backslash allows us to escape the **$** or ` so that we may pass expressions such as “You owe me \$100”.

```
jane@daisy > ~ $ echo "you owe me \$100"
you owe me $100
jane@daisy > ~ $ echo "you owe me $100"
you owe me 00
```

See the difference?

Single quotes are the strongest form of quoting and suppress **all** expansion. Whatever you put between these gets passed on verbatim. No changes whatsoever.

```
jane@daisy > ~ $ echo 'You owe me $`expr 4 \* 25`
You owe me $100
```

Back ticks evaluate and execute the contents and then pass the result to the command. Here **'You owe me $'** is passed literally as it is enclosed in single quotes. The next part, **`expr 4 \* 25`**, evaluates the expression 4 * 25 to be 100 before passing it to the echo command. The backslash is needed before the asterisk to escape its alternative meaning as a wild card.

All of this globbing and wild card stuff should not be confused with **regular expressions** (often abbreviated to **regexp**), even though they do share some common features.

regular expressions are used to manipulate and scan data for a particular pattern and is something much bigger.

You've already used regular expressions when we used the **grep** command. The command grep (**g**lobal **r**egular **e**xpression **p**rint, from the original line editor **ed** which used the command **g/re/p!** to achieve the same thing.) has two brothers known as **egrep** and **fgrep** (there's another brother known as **rgrep** but we don't see much of him).

We use the grep command to find a matching pattern of characters in a file, set of files or in a stream of data passed to the command. The general syntax is **grep {options} {pattern} {files}**.

It can be used directly as a command, or used as a filter to the output from some other command.

To find janes entry in the /etc/passwd file we could use either **grep jane /etc/passwd** or **cat /etc/passwd | grep jane**

```
jane@daisy > ~ $ cat /etc/passwd | grep jane
jane:x:500:500:jane:/home/jane:/bin/bash
```

In the above examples, jane is a pattern that grep tries to match.

Regular expressions are sequences of characters that the software uses to find a **particular pattern** in a **particular position** within the target data.

Why bother with regular expressions at all?

Linux uses plain text files for most of its configuration, for the output from commands and scripts and for reporting on system activity, security and potential problems. That's an awful lot of text, and to be able to search accurately for some particular information is one of the most important skills a command line user can master.

To use regular expressions we use three main tools:

grep is used to search for a pattern

sed is a stream editor used to filter and manipulate data streams. This enables us to pass pre-processed data directly on to the next command, to a file or to **stdout/stderr**.

awk is a scripting/programming language used to automate pattern searches and data processing.

Many other commands, such as **tr** (translate), use regular expressions, but if you can get a grip on these three tools, then you are on your way to a higher level of command line usage.

Before we go on to discuss **grep** and **sed** (I'll leave **awk** until we have done some programming in the bash scripting section), we need a good basic understanding of regular expressions. Regular expressions are used by line based commands and will not match patterns spread over two or more lines. I just thought that you ought to know that.

In regular expressions certain characters have a special meaning and these are known as **meta characters**. For any meta character to be taken literally, and to lose its special meaning, it has to **'escaped'**, usually by preceding it with a backslash character, as we have done previously with wild card characters. The following are meta characters, but not all are recognized by all applications.

- **.** The dot character matches any single characters
- ***** The asterisk matches zero or more occurrences of the preceding character.
- **^** The caret is a tricky one, it has two meanings. Outside of square brackets, it means match the pattern only when it occurs at the beginning of the line, and this is known as an **ancho**r mark. As the first character inside a pair of brackets it negates the match i.e. match anything except what follows.
- **$** Another **anchor** mark this time meaning to only match the pattern at the end of a line.
- **\< \>** More **anchor** marks. They match a pattern at the beginning \« or the end \» of a word.
- **** The backslash is known as the escape or quoting character and is used to remove the special meaning of the character that immediately follows it.
- **[ ]** Brackets are used to hold groups, ranges and classes of characters. Here a group can be an individual character.
- **\{n\}** Match **n** occurrences of the preceding character or regular expression. Note that **n** may be a single number **\{2\}**, a range **\{2,4\}** or a minimum number **\{2,\}** meaning at least two occurrences.
- **\(\)** Any matched text between **\(** and **\)** is stored in a special temporary buffer. Up to nine such sequences can be saved and then later inserted using the commands **\1** to **\9**. An example will make this clearer.
- **+** Match at least one instance of the preceding character/regexp. This is an extended regexp (ERE) – see later.
- **?** Match zero or more instances of the preceding character/regexp. This is an extended regexp (ERE) – see later.
- **|** Match the preceding or following character/regexp. This is an extended regexp (ERE) – see later.
- **( )** Used for grouping regular expressions in complex statements

With these we can find just about any pattern at any position of a line. Some of these meta characters do take on a different meaning when used to **replace** patterns.

grep

First off **egrep** is exactly the same as **grep -E** and **fgrep** is exactly the same as **grep -F** so whats the difference?

fgrep uses fixed strings to match patterns, no regular expressions at all, and so it does actually work faster.

egrep is actually a more complete version of **grep**. There are two sets of meta characters recognized by regular expressions, known as **BRE** and **ERE**, or Basic regular expressions and Extended regular expressions. BRE is a subset of ERE. BRE is used by **grep**, and **egrep** uses ERE. BRE does not recognize the meta characters **+ ?** and **|**, and requires the **( ) { }** meta characters to be escaped with a backslash. For now, we'll stick to plain old grep. Just to set the record straight, **rgrep**, the other brother, is just grep that will re-curse down through directories to find a pattern match.

The **grep** command comes with a set of options that would make any Linux command proud. Here I'll go only through those options that normal people might use.

-A -B & -C followed by a number, print that number of lines of context After, Before or around the match – this helps to recognize things in a long text file.

-c Only output a count of the number of occurrences of a match for each file scanned.

-E Use the extended set of regular expressions (ERE). The same as using the command **egrep**.

-F Don't use regular expressions – treat all pattern characters literally. The same as **fgrep**.

-f filename Use each line of the named file as a pattern to match.

-h Don't output the filename when searching multiple files.
-i Ignore case
-n Output the line numbers of lines containing a match.
-r Recurse through sub-directories.
-s Suppress error messages
-v Invert the match to select only non-matching files
-w Match only complete words. A word is a contiguous block of letters, numbers and underscores.

That's enough theory for now, so let's go visit the family grep and do a few examples.

If we were unsure how jane spelled her name (jane or jayne), then to search for her name in the /etc/passwd file we may be tempted to use the **'*'** wild card with **grep j*ne /etc/passwd** but this would fail, as the shell would expand **j*ne** before passing it to **grep**, which uses regular expressions to match the search pattern.

We could use **grep ja[n,y] /etc/passwd**.

```
jane@daisy > ~ $ grep ja[n,y] /etc/passwd
jane:x:500:500:jane:/home/jane:/bin/bash
janet:x:502:502:janet:/home/janet:/bin/bash
```

This would however also match names such as janet.

To get around this, we could use the extended set of regular expressions available with the **-E** option or the **egrep** command. **grep -Ew 'jane|jayne' /etc/passwd** match **either** jane **or** jayne. The **-w** option matches only complete words. The quotes are needed to prevent the shell expanding the vertical bar symbol into a pipe.

```
jane@daisy > ~ $ grep -Ew 'jane|jayne' /etc/passwd
jane:x:500:500:jane:/home/jane:/bin/bash
jane@daisy > ~ $
```

How many users have bash as their default shell?

grep -c '/bin/bash' /etc/passwd

```
jane@daisy > ~ $ grep -c '/bin/bash' /etc/passwd
4
jane@daisy > ~ $
```

To search for files in your home directory that contain a match, burrowing down through subdirectories and discarding warnings about inaccessible files, we could use a command such as **grep -rs glenn ~/***

```
jane@daisy > ~ $ grep -rs glenn ~/*
/home/jane/contacts-link:glenn
/home/jane/mydir/personal/mycontacts/contacts:glenn
```

Multiple use of the grep command can simplify complex searches. To search for directories that begin with an uppercase or lowercase 'm', use **ls -l | grep ^\d | grep [M,m]**. This matches all lines output from a long directory listing that begin with a 'd' (i.e., are directories). The output from this is then piped through another **grep** command to get the final result.

```
jane@daisy > ~ $ ll | grep ^\d | grep ' [M,m]'
drwxr-xr-x 2 jane jane 4096 Feb 26  2007 Movies/
drwxr-xr-x 2 jane jane 4096 Feb 25  2007 Music/
drwxr-xr-x 3 jane jane 4096 Nov 10 20:45 mydir/
drwxr-xr-x 2 jane jane 4096 Nov 10 12:06 mydir1/
jane@daisy > ~ $
```

As you can see from the examples, grep is a powerful tool to find the information that you want from files or from a commands output. It is especially useful if you don't know where that information is, or even whether it exists at all, in the places that you are looking.

Sometimes you know exactly what information is available but you want only certain parts of it. Linux has a command that will help you to get exactly what you want.

cut

The cut command is only really useful when you have tabulated data but as so many commands output data in that format it is a tool that is really worth knowing about and it is really simple to use.

When you examine a file of tabulated data, you'll see groups of characters separated by a common character, often a space, comma or colon. This common character is known as a delimiter, and the groups of characters are known as fields. The cut command searches each line of text looking for the delimiter, and numbering the fields as it does so. When it reaches the end of the line, it outputs only those fields that have been requested. The general form of the **cut** command is:

cut {options}{file}

The most useful options are:

-c list Select only characters in these positions

-d Specify the delimiter. If omitted, the default is tab. Spaces need to be quoted -d" "
-f list Select the following fields
-s Suppress lines without delimiters

List is a sequence of numbers separated by a comma or, To specify a range, by a hyphen.

If we look at a line of data from the /etc/passwd file, we will notice that the various **'fields'** are delimited by a colon **:**.

```
jane@daisy > ~ $ cat /etc/passwd | grep jane
jane:x:500:500:jane:/home/jane:/bin/bash
```

The first field contains the users login name and the fifth field, known for historical reasons as the **gecos** field (from **G**eneral **E**lectric **C**omprehensive **O**perating **S**ystem), and contains the users real name and sometimes some optional location information, although PCLinuxOS doesn't use this additional information.

To extract the users login names and real names we use the command like this: **cut -d: -f1,5 /etc/passwd**.

This tells the command to look for a colon as a delimiter and to output fields 1 and 5.

All of this is fine for nicely ordered data as we find in the /etc/passwd file, but in the real world things don't always work out like that. Take for example the **ls -l** command. This outputs a nicely formatted long directory listing. The catch here is that to make the output look nice and neat, the **ls** command pads the output with extra spaces. When our **cut** command scans the line using a space as the delimiter it increases the field count each time it encounters a space and the output is, at best, unpredictable. Many Linux commands pad their output in this way. The **ps** commands output is another example of this.

```
jane@daisy > ~ $ ls -l
total 28
-rw-r--r-- 1 jane jane    0 Feb  7 05:34 contacts
drwxr-xr-x 3 jane jane 4096 Feb  7 05:25 Desktop/
drwxr-xr-x 2 jane jane 4096 Feb  7  2010 Documents/
drwxr-xr-x 2 jane jane 4096 Feb  7  2010 Movies/
```

If I wanted to extract the owner, size and file names from this listing it would be reasonable to assume that I needed fields 3,5 & 9 and that the delimiter is a space.

```
jane@daisy > ~ $ ls -l | cut -d" " -f3,5,9

jane  Feb
jane 4096 05:25
jane 4096
```

As you can see, the output is not as expected. We could try the **-c** option, ignoring fields and counting the characters from the start of the line.

```
jane@daisy > ~ $ ls -l | cut -c14-18,24-27,41-

jane    0 contacts
jane 4096 Desktop/
jane 4096 Documents/
```

But apart from being tedious and error prone, if the directory listing changes slightly then the numbers will be different and the code is not re-usable, we would have to start over.

```
jane@daisy > ~ $ ls -l
total 32
-rw-r--r-- 1 jane jane        0 Feb  7 05:34 contacts
drwxr-xr-x 3 jane jane     4096 Feb  7 05:25 Desktop/
drwxr-xr-x 2 jane jane     4096 Feb  7  2010 Documents/
drwxr-xr-x 2 jane contacts 4096 Feb  7 05:58 Friends/
```

```
jane@daisy > ~ $ ls -l | cut -c14-18,24-27,41-

jane     5:34 contacts
jane     5:25 Desktop/
jane     2010 Documents/
jane cts 5:58 Friends/
```

To work around this, we need to prepare the output from the **ls** command by filtering out the extra spaces. We can do this by using a command we have met once before. The **tr** command translates data, and we used it previously to change text from lowercase to uppercase. If we use the **tr** command with the **-s** option followed by a space, it **sqeezes** repeated spaces out of the file or data stream that we feed it.

```
jane@daisy > ~ $ ls -l | tr -s " "
total 32
-rw-r--r-- 1 jane jane 0 Feb 7 05:34 contacts
drwxr-xr-x 3 jane jane 4096 Feb 7 05:25 Desktop/
drwxr-xr-x 2 jane jane 4096 Feb 7 2010 Documents/
drwxr-xr-x 2 jane contacts 4096 Feb 7 05:58 Friends/
```

We can now cut out exactly the data the we want.

```
jane@daisy > ~ $ ls -l | tr -s " " | cut -d" " -f3,5,9

jane 0 contacts
jane 4096 Desktop/
jane 4096 Documents/
jane 4096 Friends/
```

Two other commands, rather simple but occasionally useful, so worth mentioning, are **paste** and **join**. Typing the command name followed by **--help** will give you enough information to use these commands, but a simple example may better show their usefulness and their differences.

Suppose we have two files containing different data about common things such as these:

```
jane@daisy > Friends $ cat friends1
jane    12 High Street  Newtown
john    12 High Street  Newtown
janet   1432 Long Avenue        Old town
jane@daisy > Friends $ cat friends2
jane    age 22  height 5'-4"
john    age 19  height 5'-10"
janet   age 28  height 5'-5"
```

```
jane@daisy > Friends $ join friends1 friends2
jane 12 High Street Newtown age 22 height 5'-4"
john 12 High Street Newtown age 19 height 5'-10"
janet 1432 Long Avenue Old town age 28 height 5'-5"
```

The names in these files are common but the data is different. We can merge the data from both files with **join**.

With **paste** we can add data we cut from one file to the end of another file on a line by line basis

```
jane@daisy > Friends $ cut -f2-4 friends2 > friends3
jane@daisy > Friends $ paste friends1 friends3
jane    12 High Street  Newtown age 22  height 5'-4"
john    12 High Street  Newtown age 19  height 5'-10"
janet   1432 Long Avenue        Old town        age 28  height 5'-5"
```

sort

When you have found the data that you want, cut out all but the required information, and joined or pasted the results, it may not be in the order that you want. Here, Linux has an exceptionally powerful & quick utility to do just that.

The syntax of the **sort** command is **sort {options} {file(s)}**. If more than one file is supplied then the **combined** contents of the files will be sorted and output.

The options available for the sort command make for a rather comprehensive utility. These are the most useful ones:

-b Ignore leading blanks in the sort field
-c Only check whether the data is sorted but do not sort
-d Sort in dictionary order, consider only blanks and alphanumeric characters
-f Treat upper and lowercase as equal
-i Consider only printable characters. This is ignored if the **-d** option is specified.
-k Specify the sort field
-n Numeric sort
-r Reverse the sort order
-t Specify the field separator
-u Output only the first of one or more equal lines. If the **-c** option is specified, check that no lines are equal

A couple of these options need further explanation.

sort doesn't have the hangups about field delimiters that commands like **cut** have. Fields, as far as **sort** is concerned, are separated by the blank space between them, even if these blank spaces contain multiple non-printing characters. This is generally a good idea, but occasions arise when this causes problems, as in /etc/passwd, which has no blank spaces. In these cases, the field separator can be specified with the **-t** option. **sort -t: -k5 /etc/passwd** would sort the file on the 5th (users real name), using the colon to decide where fields start and end.

Specifying the sort field used to be a strange affair, but with the **-k** option it is now reasonably straight forward.

-k {number} Specifies the field in position **{number}**.

Numbering starts at 1. More complex sort field arguments may be specified, such as which character within the sort field to start or end sorting on. I like to take a 'learn it if you need it' approach to these things, as I find that I rarely need such features and I don't like to clutter my poor brain unnecessarily.

ls -l | sort -k9 Sorts a directory listing in dictionary order on the 9th field (file name).

```
jane@daisy > ~ $ ls -l | sort -k9
total 44
-rw-rw---- 1 jane jane   44 Nov 10 12:37 contacts2
-rw-r--r-- 2 jane jane   39 Nov 10 12:33 contacts-link
drwxr-xr-x 3 jane jane 4096 Nov 16 08:34 Desktop/
drwxr-xr-x 2 jane jane 4096 Feb  6 04:23 Documents/
drwxr-xr-x 2 jane jane 4096 Feb 26  2007 Movies/
drwxr-xr-x 2 jane jane 4096 Feb 25  2007 Music/
drwxr-xr-x 3 jane jane 4096 Nov 10 20:45 mydir/
drwxr-xr-x 2 jane jane 4096 Nov 10 12:06 mydir1/
-rw-r--r-- 1 jane jane   69 Nov 10 12:10 newfile
-rw-r--r-- 1 jane jane 2174 Nov 10 12:12 newfile2
drwxr-xr-x 2 jane jane 4096 Feb  8 07:54 Pictures/
lrwxrwxrwx 1 jane jane    5 Nov 12 12:39 tmp -> //tmp/
```

ls -l | sort -nrk5 Sorts the listing by the 5th field (file size) in **r**everse **n**umerical order.

```
jane@daisy > ~ $ ls -l | sort -nrk5
drwxr-xr-x 3 jane jane 4096 Nov 16 08:34 Desktop/
drwxr-xr-x 3 jane jane 4096 Nov 10 20:45 mydir/
drwxr-xr-x 2 jane jane 4096 Nov 10 12:06 mydir1/
drwxr-xr-x 2 jane jane 4096 Feb  8 10:40 Pictures/
drwxr-xr-x 2 jane jane 4096 Feb  6 04:23 Documents/
drwxr-xr-x 2 jane jane 4096 Feb 26  2007 Movies/
drwxr-xr-x 2 jane jane 4096 Feb 25  2007 Music/
-rw-r--r-- 1 jane jane 2174 Nov 10 12:12 newfile2
-rw-r--r-- 1 jane jane   69 Nov 10 12:10 newfile
-rw-rw---- 1 jane jane   44 Nov 10 12:37 contacts2
-rw-r--r-- 2 jane jane   39 Nov 10 12:33 contacts-link
lrwxrwxrwx 1 jane jane    5 Nov 12 12:39 tmp -> //tmp/
total 44
```

An awful lot can be done with these few commands and a little practice. If you want to do more, then of course you can. This is where the **sed** stream editor excels, enabling search and replace, insertion, deletion, substitution and translation to multiple files at the same time if that is what you want. **sed** can be a simple substitution tool or as complex as you like. We'll be meeting **sed** very soon.

Screenshot Showcase

ramchu, running KDE 4, posted February 4, 2010

Game Zone: gbrainy

by MeeMaw

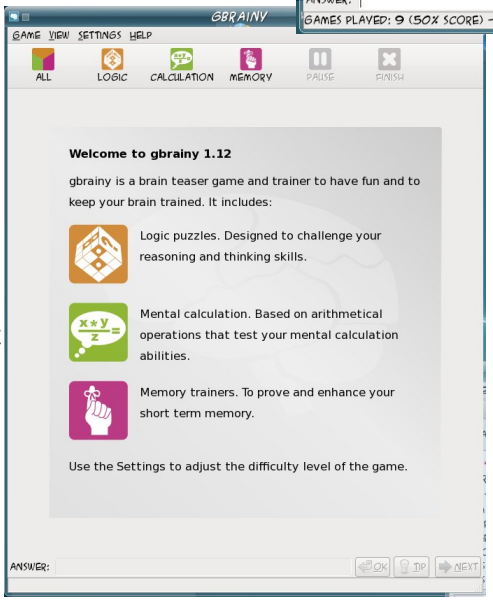
As one gets older, the mind sometimes weakens..... I'm thinking that at my age, I need some exercises to keep my mind nimble. I like some of the crossword puzzles and word games. Is there anything in the way of computer games that might help?

Well, sure, there are hundreds of computer games! However, we've played many of them so much now that it doesn't take much brainpower to succeed. Let's try something new.

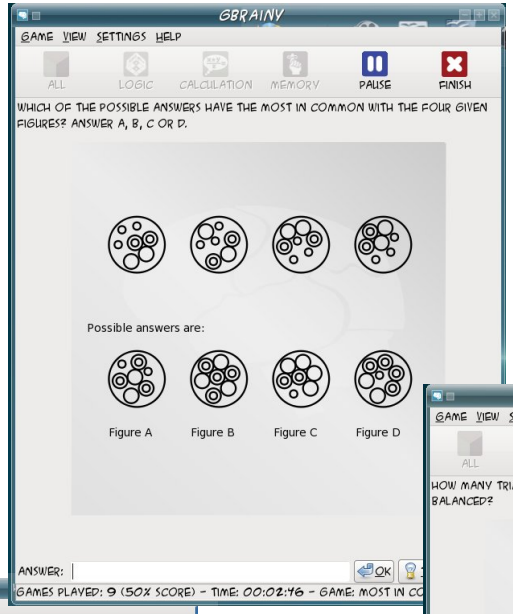
GBrainy is a “brain teaser game and trainer to have fun and to keep your brain trained.”

It is in our repository. Simply go to Synaptic, mark GBrainy for installation and click Apply.

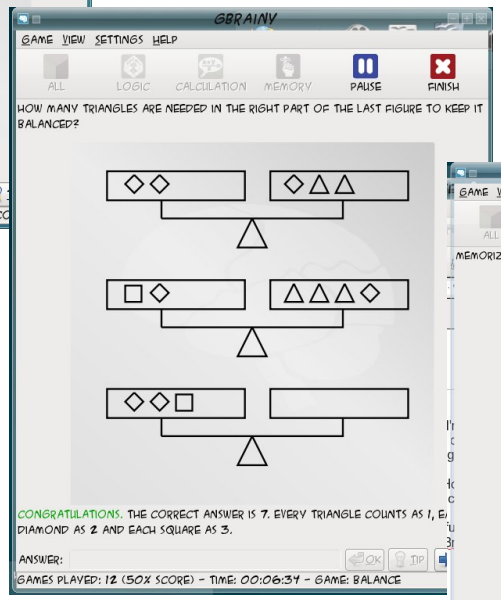
When you open it, you get the screen shown here;



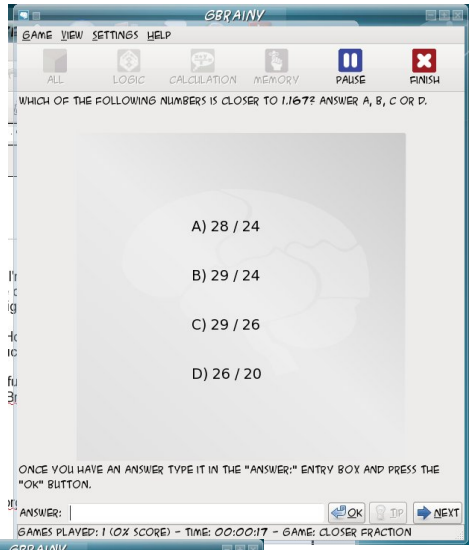
As you can see, three types of games are there for you to try. You may stick to one type or try a combination of two or all three types.

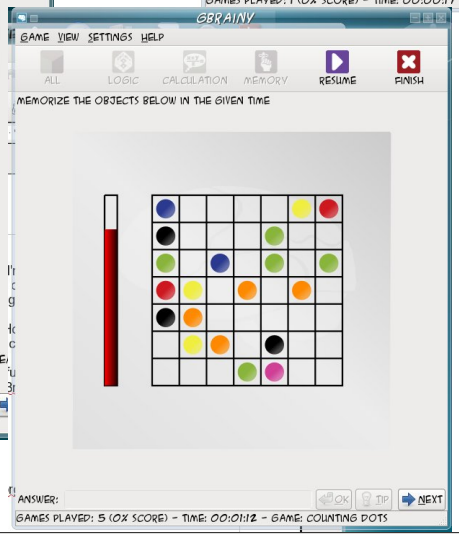


Logic Puzzles – In one type, you are asked to pick the design from a group which has the most in common with four other designs. In another type you are asked to fill in the blank to make the scales balance. In another type, you are given a series of numbers which follow a pattern and asked what the next number is.



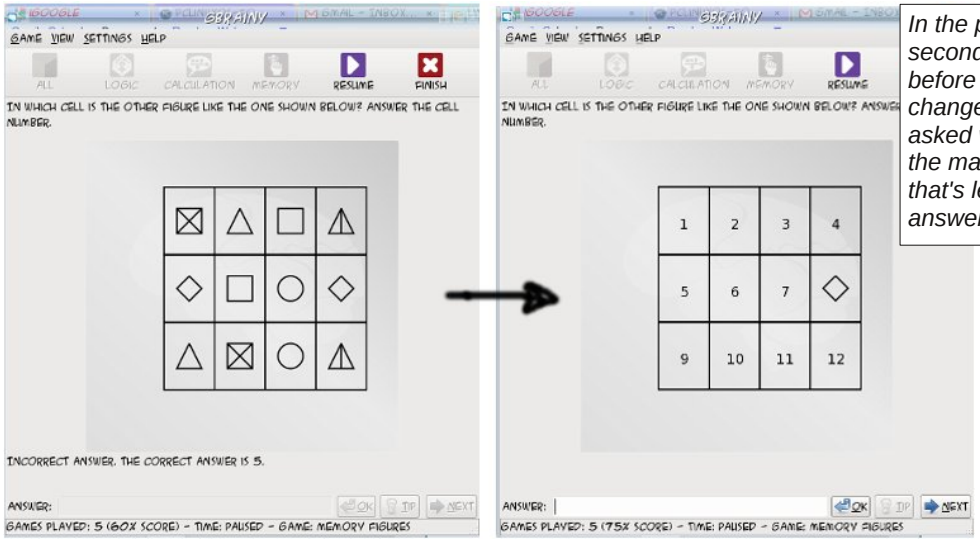
Mental Calculation – In one type, you are given three numbers and an answer and you need to provide the operations necessary to complete the calculation. In another type, you are presented with four fractions and asked which one is closest in value to a given number written in decimal form. There are also calculation problems. You may be tempted to get out your calculator, but you are also timed.





When the bar fills, your time is up, and the grid disappears. Then you are asked how many circles of a certain color there were in the grid.

Memory Puzzles - In one type, you are given a grid of symbols to look at for a certain length of time, then all symbols in the grid except one are replaced by numbers and you are asked to remember where the other matching symbol is located. In another

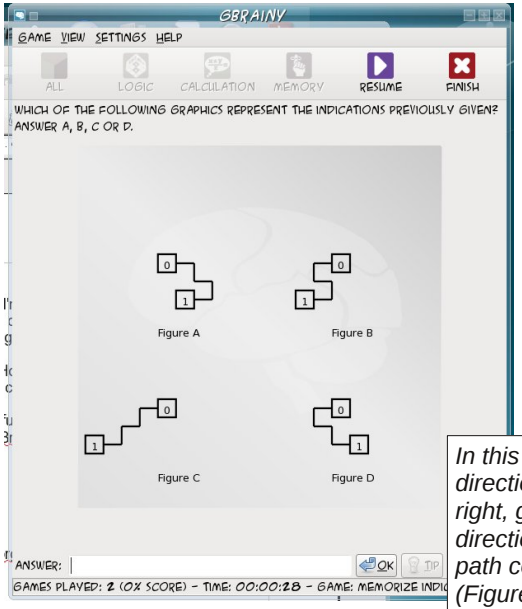


In the puzzle at left, you have 5 seconds to memorize the above before all symbols but one are changed to numbers - then you are asked what numbered square has the matching symbol to the one that's left. In the example above, the answer is square 5.

type you are given a certain length of time to memorize an arrangement of dots on a grid, a group of words, a group of symbols or a sequence of directions, then asked a question about it later after it has been removed.

You can stop at any time by clicking on 'Finish'. When you do, you will get a score telling the total percentage of questions you got right. If you used more than one type of puzzle it will be broken down into types as well. I'm not going to tell you it's all easy... after all, we're trying to "sharpen".

The game can be configured for difficulty (Easy, Medium or Master), including how long you get to study something before it disappears (the default is 4 seconds) and whether or not you see a countdown (3, 2, 1 or a bar that fills with color.)

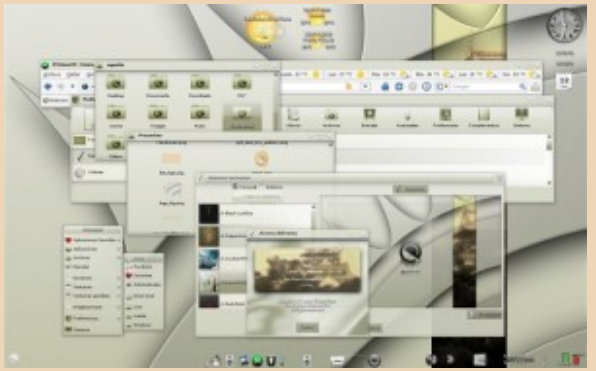


It is also available for Windows users (from Sourceforge.net) In the Windows version, a Verbal section is also present which does word analogies (Fish is to Aquarium as Monkey is to ?) and other word exercises.

Have fun sharpening your brain!!!!

In this puzzle you have been shown a set of directions like "Start at 0, go right, go down, go right, go down, go left, end at 1" - then the directions disappear and you are asked which path corresponds to the directions you saw (Figure A).

PCLinuxOS Enlightenment e17 ISO



Coming soon!

2009 LinuxQuestions.org Members Choice Awards

by Paul Arnote (parnote)

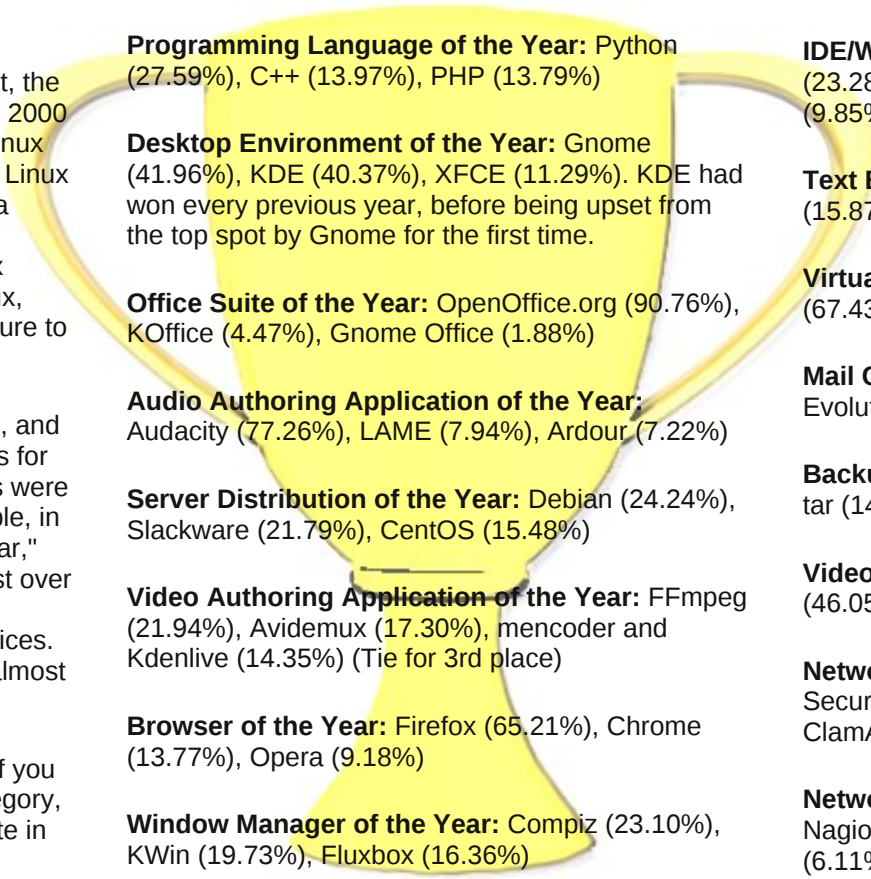
From January 8, 2010 to February 9, 2010, members of the LinuxQuestions.org forum were given the chance to vote for their favorite Linux software of 2009. This was the 9th consecutive year for the poll.

For those of you who may not be familiar with it, the LinuxQuestions.org forum was started in June, 2000 as a general forum where users can discuss Linux issues. There are forum categories for specific Linux distros, as well as a Linux Beginner's section, a Linux Programming section, a Linux Software category, which also contains a separate Linux Games category for running games under Linux, and a whole host of other categories that are sure to capture your interests.

So, software was divided up into 27 categories, and members had a solid month to cast their ballots for their favorite Linux programs. Some categories were as predictable as you might expect. For example, in the category of "Desktop Distribution of the Year," Ubuntu was the runaway favorite, capturing just over 30% of the vote, with Slackware (16.74%) and Fedora (9.78%) rounding out the top three choices. (Yes, I voted for PCLinuxOS, which captured almost 3% of the vote).

Voting was not required in every category, so if you had no knowledge about the software in a category, you could skip voting in that area. Only one vote in each category per registered forum user was allowed.

Below is a roundup of the top three choices in the 26 other categories. Feel free to visit the official poll results to see how your favorite Linux program fared. You do not need to be a member to read the posts there. You do, however, need to be a member to post there. Registration is free.



Programming Language of the Year: Python (27.59%), C++ (13.97%), PHP (13.79%)

Desktop Environment of the Year: Gnome (41.96%), KDE (40.37%), XFCE (11.29%). KDE had won every previous year, before being upset from the top spot by Gnome for the first time.

Office Suite of the Year: OpenOffice.org (90.76%), KOffice (4.47%), Gnome Office (1.88%)

Audio Authoring Application of the Year: Audacity (77.26%), LAME (7.94%), Ardour (7.22%)

Server Distribution of the Year: Debian (24.24%), Slackware (21.79%), CentOS (15.48%)

Video Authoring Application of the Year: FFmpeg (21.94%), Avidemux (17.30%), mencoder and Kdenlive (14.35%) (Tie for 3rd place)

Browser of the Year: Firefox (65.21%), Chrome (13.77%), Opera (9.18%)

Window Manager of the Year: Compiz (23.10%), KWin (19.73%), Fluxbox (16.36%)

Database of the Year: MySQL (60.81%), PostgreSQL (22.65%), sqlite (8.40%)

Open Source Game of the Year: Battle for Wesnoth (15.45%), Open Arena (9.27%), Nexuiz (8.99%)

IDE/Web Development Editor of the Year: Eclipse (23.28%), Netbeans (15.52%), Bluefish and Geany (9.85%) (Tie for 3rd place)

Text Editor of the Year: vim (35.29%), gedit (15.87%), Kate (10.40%)

Virtualization Product of the Year: VirtualBox (67.43%), VMware (15.24%), KVM (6.67%)

Mail Client of the Year: Thunderbird (53.48%), Evolution (14.77%), Kmail (11.38%)

Backup Application of the Year: rsync (48.99%), tar (14.41%), Clonezilla (8.93%)

Video Media Player Application of the Year: VLC (46.05%), mplayer (36.28%), xine (6.36%)

Network Security Application of the Year: Nmap Security Scanner (29.85%), Wireshark (23.13%), ClamAv (10.82%)

Network Monitoring Application of the Year: Nagios (51.11%), OpenNMS (15.00%), Zenoss (6.11%)

Open Source CMS/Blogging Platform of the Year: WordPress (45.20%), Drupal (23.60%), Joomla! (22.80%)

Graphics Application of the Year: GIMP (66.48%), Inkscape (12.66%), Blender (5.77%)

Audio Media Player Application of the Year: Amarok (38.81%), Rhythmbox (16.83%), Audacious (9.37%)

File Manager of the Year: Nautilus (24.92%), Dolphin (19.66%), Konqueror (18.81%)

Security/Forensic/Rescue Distribution of the Year: BackTrack (43.48%), SystemRescueCd (31.88%), Trinity Rescue Kit (4.35%)

Multimedia Utility of the Year: GStreamer (32.84%), digiKam (26.04%), XBMC (14.79%)

Messaging Application of the Year: Pidgin (48.74%), Skype (12.18%), Kopete (11.76%)

Host Security Application of the Year: SELinux (39.26%), chkrootkit (16.56%), AppArmor (12.27%)

Screenshot Showcase

ef2000, running e17, February 16, 2010

ms_meme's Nook: PCLOS You Light Up My Screen

MP3

OGG

So many nights I'd sit using Windows
Waiting for hours till it would boot
In so many dreams I kept doing scans
Alone in the dark and then I found root

You light up my screen now I can sing and carry on
My days are filled with hope no more worms

Rolling a drift with all the trojans
Could it be finally I'm turning for home
Finally a chance to say hey I love you
From PCLOS I never will roam

'Cause you light up my screen now I can sing and carry on
My days are filled with hope no more worms
I can't be wrong you're a Linux dream
'Cause you you light up my screen

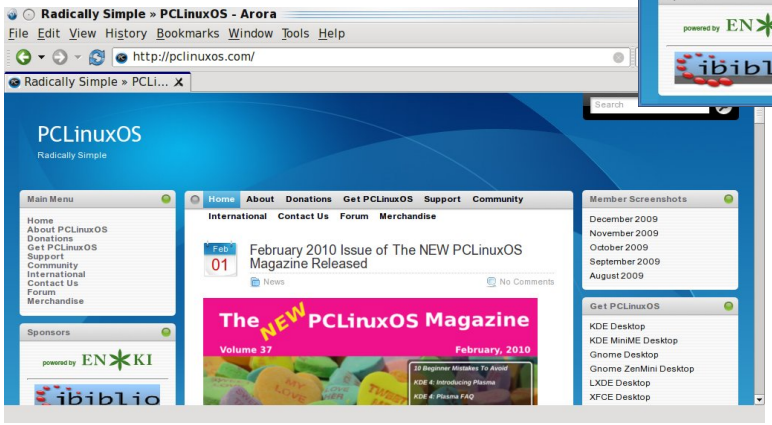
PCLOS You Light Up My Screen

Web Browser Roundup

by Andrew Huff (athaki)

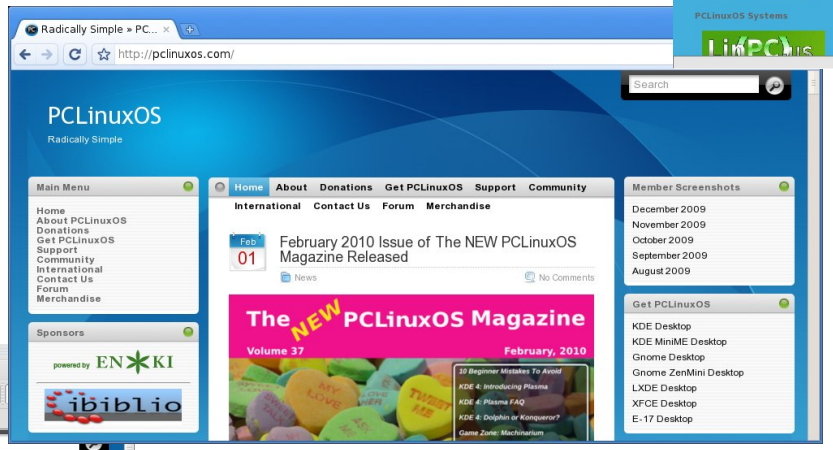
In this article, I'll be showcasing the various web browsers that are in the PCLinuxOS repository. All versions reviewed are, as of Feb 5th, the most current version in the repository. These include: Arora, Chromium, Dillo, Elinks, Epiphany, Firefox, Flock, Galeon, Konqueror, Lynx, Midori, Netscape, Opera and SeaMonkey. The test machine is my Acer Aspire One AOA 150 with 1GB ram and 160GB HDD running KDE 4 on kernel 2.6.26.8.tex3. I've also measured their ram usage against the PCLinuxOS.com homepage, javascript handling ability by accessing my Gmail and video playback via this website: http://www.youtube.com/watch?v=MDhMBxAHGYE (Rick Roll Geek edition). To measure the ram, I used the system monitor application in KDE 4 and looked at the memory the application used (not the shared memory). Let's get started!

Arora



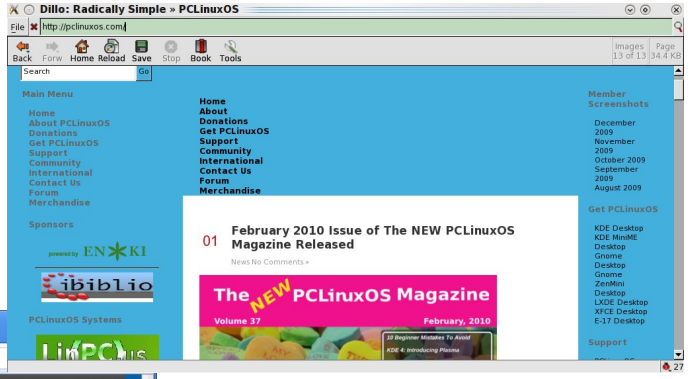
Arora is a lightweight, webkit based browser and also happens to use the most RAM out of all the browsers tested at a whopping 47.9MB on our test page. It has a common window layout which would make newcomers to the browser feel at home. When accessing javascript heavy websites such as Gmail, Arora struggles, leaving a less than to be desired user experience. Arora doesn't seem to have plugin capabilities, though it does have a private browsing feature. Playing the video from youtube was flawless.

Chromium



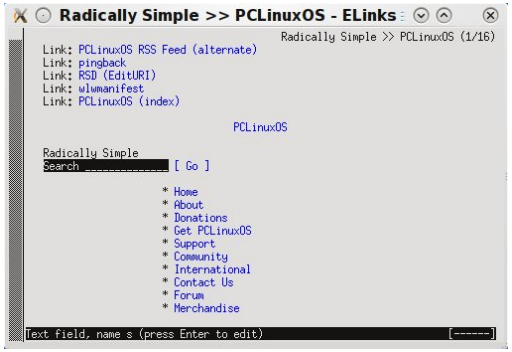
Chromium is the Google open-source project on which Google Chrome and the future Google Chrome OS is based. It uses WebKit and in our test used 25.4MB of ram spread across four processes. Chromium can sync your bookmarks with your Google account, is lightning fast with Gmail and the video also played flawlessly. Chromium has a lot of extensions at: https://chrome.google.com/extensions?hl=en-US.

Dillo



Dillo is another lightweight web broswer. It uses the Fast light toolkit (FLTK) and used 9.2MB of ram. It is a nine year old project and even though the developers say that it is still a beta, it is very stable. However, it did have difficulty rendering the PCLinuxOS webpage. You cannot select all the text in the address bar at once via the shortcut ctl-a (double clicking does select all the text), it can not access Gmail (ssl support is in alpha stage) and it won't play the video on youtube.

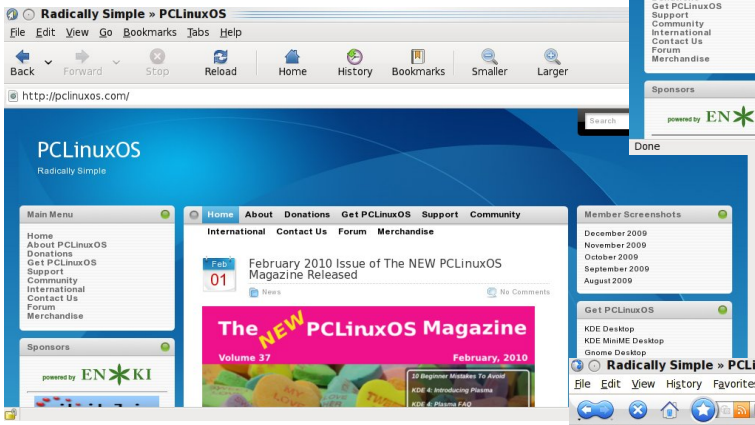
Elinks



Elinks is one of two text-based web broswers included in the PCLinuxOS repository. Navigation uses the keyboard arrow

keys. Since it is text-based, it uses the least amount of RAM of all the web browsers in this roundup, with 1.6MB used. Gmail does work, but it looks awesomely different than what we're usually used to seeing. Of course, the youtube video would not play.

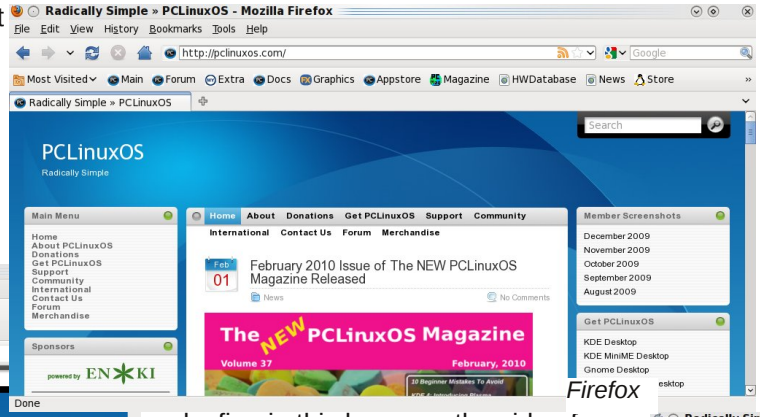
Epiphany



Epiphany is GNOME's default web browser. It has a familiar layout and uses the WebKit rendering engine. Its predecessor, Galeon, is still based on Gecko. It used 18.3MB while on our test page. Gmail does work fine in this browser and youtube worked as great as the other browsers that would play the video. I could not find any plugin support.

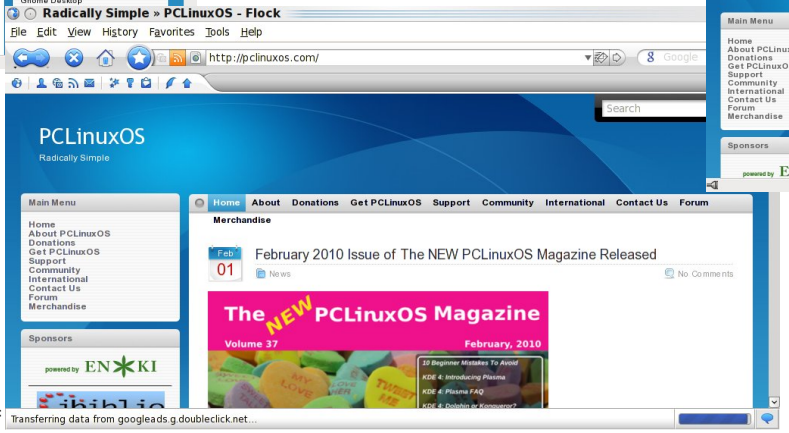
Firefox

This browser needs no introduction. It is one of the most popular default browsers for GNU/Linux distributions and one of the most known open-source projects in the world. It uses the Gecko engine and in our test used 22.3MB of ram. Gmail



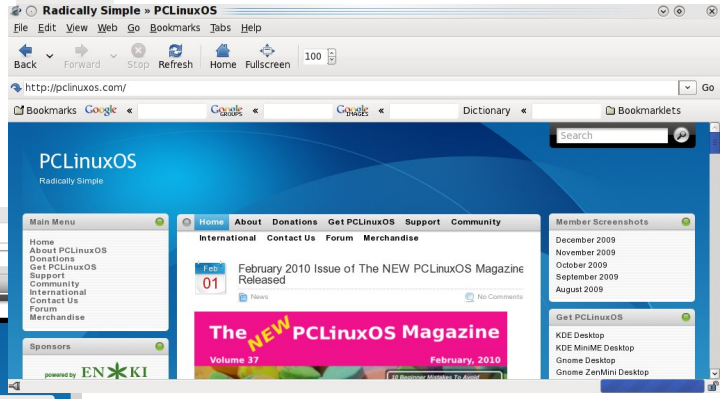
Firefox works fine in this browser, the video from youtube plays quite well and there are scads of plugins available.

Flock



Flock is a social web browser based on Firefox and as such also uses Gecko. Since it is a social browser, it comes with integrated support for numerous social networking sites such as Twitter and Facebook. The little bar underneath the large toolbar is full of social network integration with such titles as: My World, Open Blog Editor, Open Photo Up-loader, Open People Sidebar, et cetera. In our test, it used 30.0MB of RAM. Gmail works great and Flock asked to remember it so that "I'd always know when I had mail." The youtube video also played very well. Flock has 3 extensions developed especially for it and it also supports all firefox addons as well.

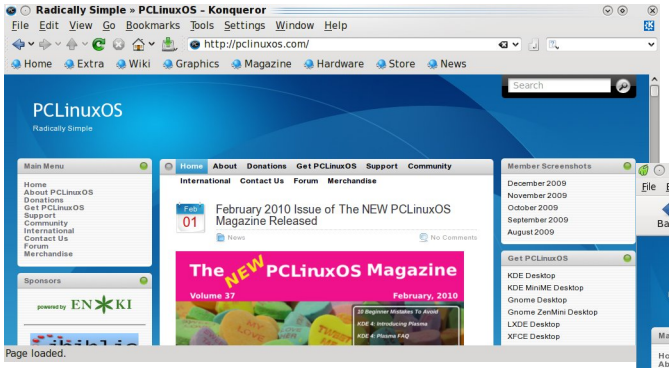
Galeon



Galeon is a GNOME browser based on Mozilla and therefore uses Gecko as well. Of the graphical web browsers, Galeon used the least amount of RAM, coming in at 13.7MB on our test site. Gmail loaded and youtube played the video.

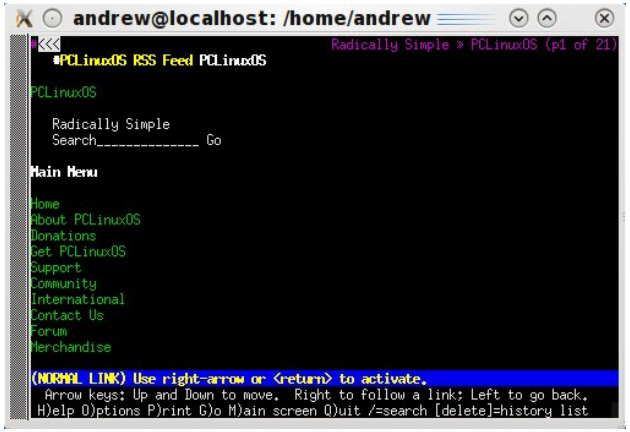
Konqueror

Konqueror is the default web browser for KDE and uses its own rendering engine, KHTML. In our test, it used 24.0MB of RAM. Konqueror does have some extensions that are useful, such as translations, text-



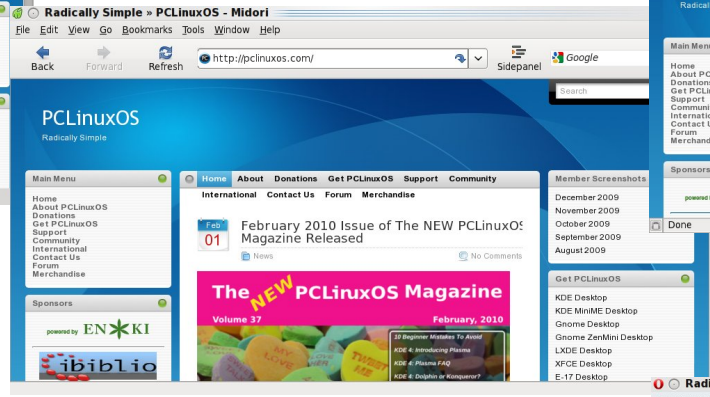
to-speech and a plugin that archives webpages. Unfortunately, Gmail doesn't support Konqueror and only renders the basic HTML view. The youtube video rendered quite well in Konqueror, however.

Lynx



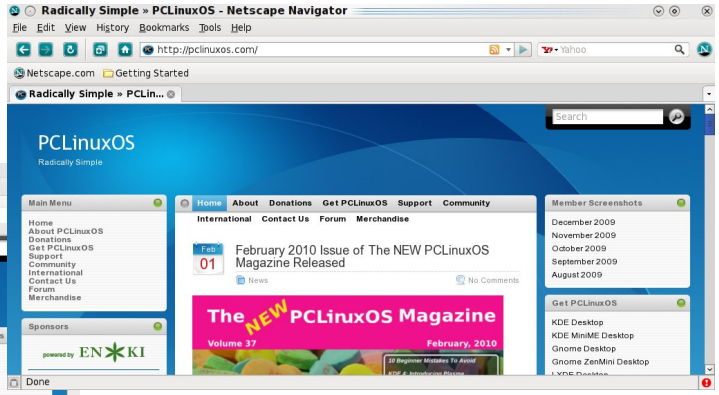
Lynx is the oldest web browser in the repository. It was first built in 1992. It is a text based browser and used 1.7MB of RAM in our test. What's interesting to note in Lynx is that one must use keyboard shortcuts and the arrow keys to navigate. Gmail is usable in Lynx, though the sidebar is above the inbox area. The youtube video would not display for obvious reasons.

Midori



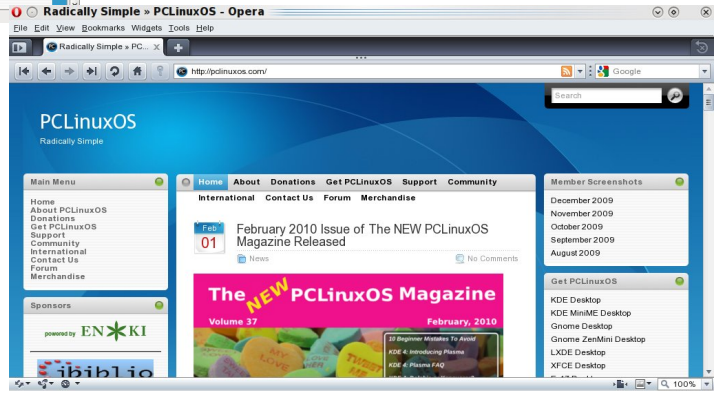
Midori is the default web browser in the community spun PCLXDE distribution. It uses WebKit as its rendering engine and in our test used 17.7MB of RAM. Gmail rendered in its basic HTML view for this browser as well, citing that I should use a supported browser. The youtube video played flawlessly.

Netscape Navigator



Although official support for this browser ended on March 1st 2008, Netscape Navigator is in the repository as well. Netscape was among the first graphical web browsers and the company started the Mozilla project in 1998. Although the company no longer exists, it is nice to have this in our repository as a tribute to all the good that came out of the company. Netscape Navigator 9 is based on Mozilla code and acts a lot like Firefox 2. In our test, Netscape used 17.9MB of RAM. Gmail loaded as normal (no HTML view), and the youtube video played normally.

Opera



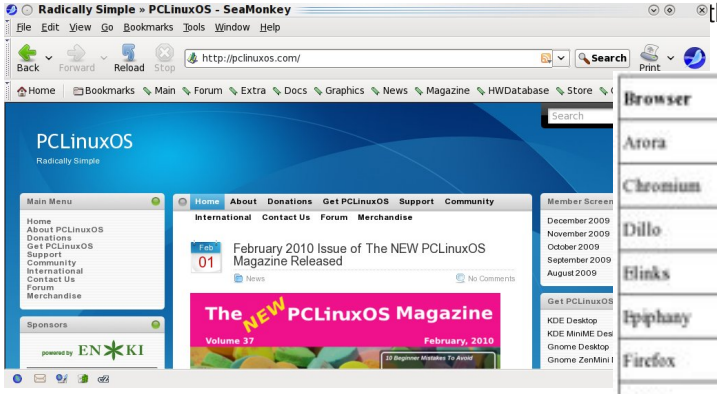
Opera has been one of the most innovative browsers in recent years although it is non-free software. Upon loading in KDE, Opera gives the following information:

"Opera has detected KDE is running.
Some Opera keyboard shortcuts (like Ctrl+F4) may

not work because KDE has reserved them. You can modify KDE shortcuts in the KDE Control Center."

Opera uses the Presto rendering engine. It has widgets, which opera describes as 'self-contained web applications' that use open web standards. One of my personal favorites is the aquarium widget. In our test, the browser used 27.3MB of RAM. Gmail loaded up great. The youtube video started off jerky, but after a few seconds was fine again.

SeaMonkey



SeaMonkey is an all-in-one software suite by the Mozilla project that aims to emulate the older Netscape Communicator. It has a built in e-mail client, IRC client, website creator and newsgroup reader. In our test, it used 24.0MB of RAM. Gmail loaded fine and the youtube video played flawlessly.

Concluding Remarks

I've just given a brief overview of each web browser in the PCLinuxOS repository. In my opinion, all of them have their strengths and weaknesses. However, not all of them could be used as your main web browser if you particularly need javascript applications or video playback. My personal favorites out of all of them, in no particular order, are: Firefox, Chromium, Opera and SeaMonkey. I'd also recommend Flock to anyone who is into the social networking scene. I enjoyed testing out all of them for this article, and I plan on keeping all of them for future use. Some of them are still in beta, and I excitedly await new releases. At the end of this article is a table of the web browsers, their rendering engines, and the amount of RAM they used on my test machine.

Browser	Rendering Engine	Ram Used
Arora	Webkit	47.9mb
Chromium	Webkit	16.7, 1.4, 6.0, 1.3mb = 25.4mb
Dillo	FLTK	9.2mb
Elinks	Text	1.6mb
Epiphany	Webkit	18.3mb
Firefox	Gecko	22.3mb
Flock	Gecko	30.0mb
Galeon	Gecko	13.5mb
Konqueror	KHTML	24.0mb
Lynx	Text	1.7mb
Midori	Webkit	17.7mb
Netscape	Gecko	17.9mb
Opera	Presto	27.3mb
Seamonkey	Gecko	24.0mb

More Screenshot Showcase

Clockwise, from top left:

georgetoon, running KDE 4, posted February 7, 2010.

Archie, running KDE 4, posted February 6, 2010.

Crow, running e17, posted February 4, 2010.

coffeetime, running e17, posted February 17, 2010.