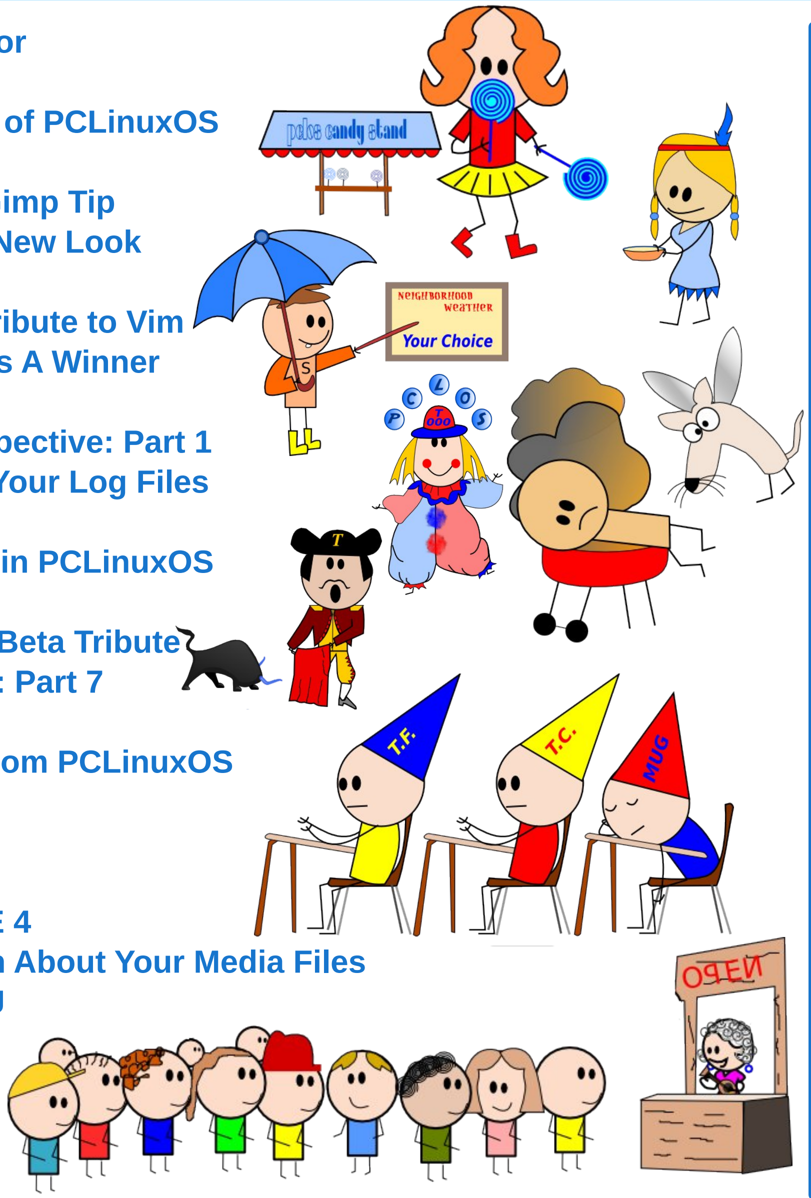
The NEW PCLinuxOS Magazine

Volume 39

April, 2010

Table Of Contents



The PCLinuxOS Magazine

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

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

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

Staff:

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

Contributors:

Agopov Sergey	Peter Kelly
Robert Stahl	Dan Malewski

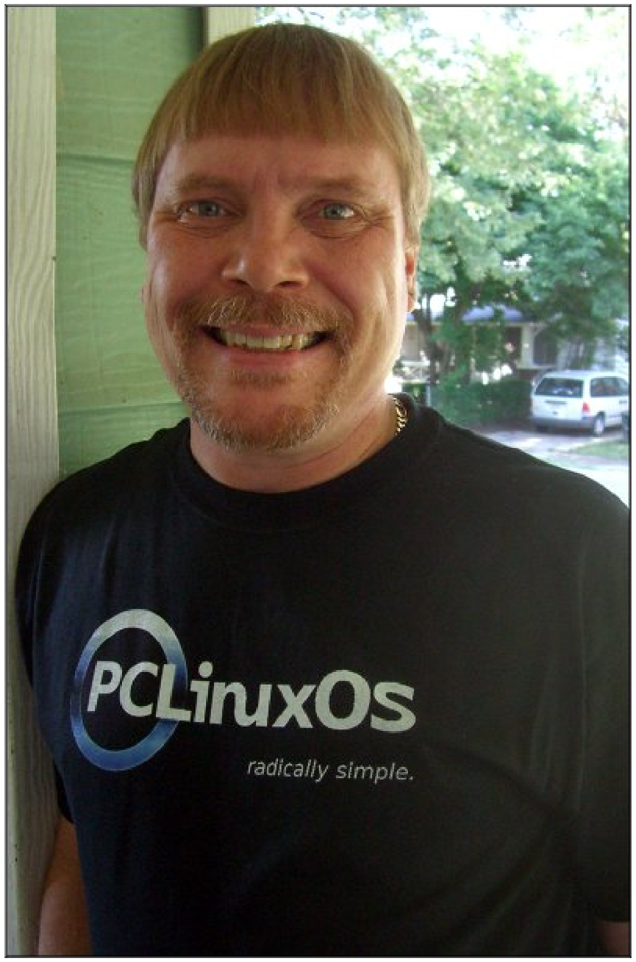
Welcome From the Chief Editor

Yes, indeed. These are very exciting times for PCLinuxOS. Texstar and the developers continue to tweak and tune the PCLinuxOS 2010 beta releases, in preparation for their (very near) final release. The forum is ablaze with excitement as the community helps to fine tune the beta releases.

As the PCLinuxOS community marches towards the eventual and emminent final release of PCLinuxOS 2010, there are no shortage of topics to cover. To start off with, I recap the state of the **current beta releases** over the last month. I then take a look at the sense of community among PCLinuxOS users, in **Community: The Heart & Soul of PCLinuxOS**. The magazine continues its look at the new features of **KDE 4, with KDE 4: KDE Control Center's New Look, KDE 4: KSystemLog Reveals Your Log Files**, and **KDE 4: KWin's Desktop Effects a Winner**.

Agapov Sergey gives us some direction on **setting up a fake RAID array in PCLinuxOS**. Dan Malewski starts us off on what is planned to be an every other month feature, with **Gimp From A Beginner's Perspective: Part 1**. Peter Kelly delivers the latest installment of his article series, **Command Line Interface Intro: Part 7**. Gary Ratliff, Sr. explores another computer programming language, in his series **Computer Languages A to Z: J**.

ms_meme has been busy this past month, as well. Her usual columns are back, with **Forum Foibles: Forum Fools**, and **ms_meme's Nook: Linux Fool**. She also delivers two more songs, with **Vim Vim Veree: ms_meme's Tribute To Vim**, and **Beta, Beta, Beta: ms_meme's Beta Tribute**. Additionally,



ms_meme writes a beginner's view of **Vim, in Vim, Vigor & Vinegar**. It is also her artwork from the PCLinuxOS Forum topic, ms_meme's Neighborhood, that graces the cover and the Table of Contents page this month.

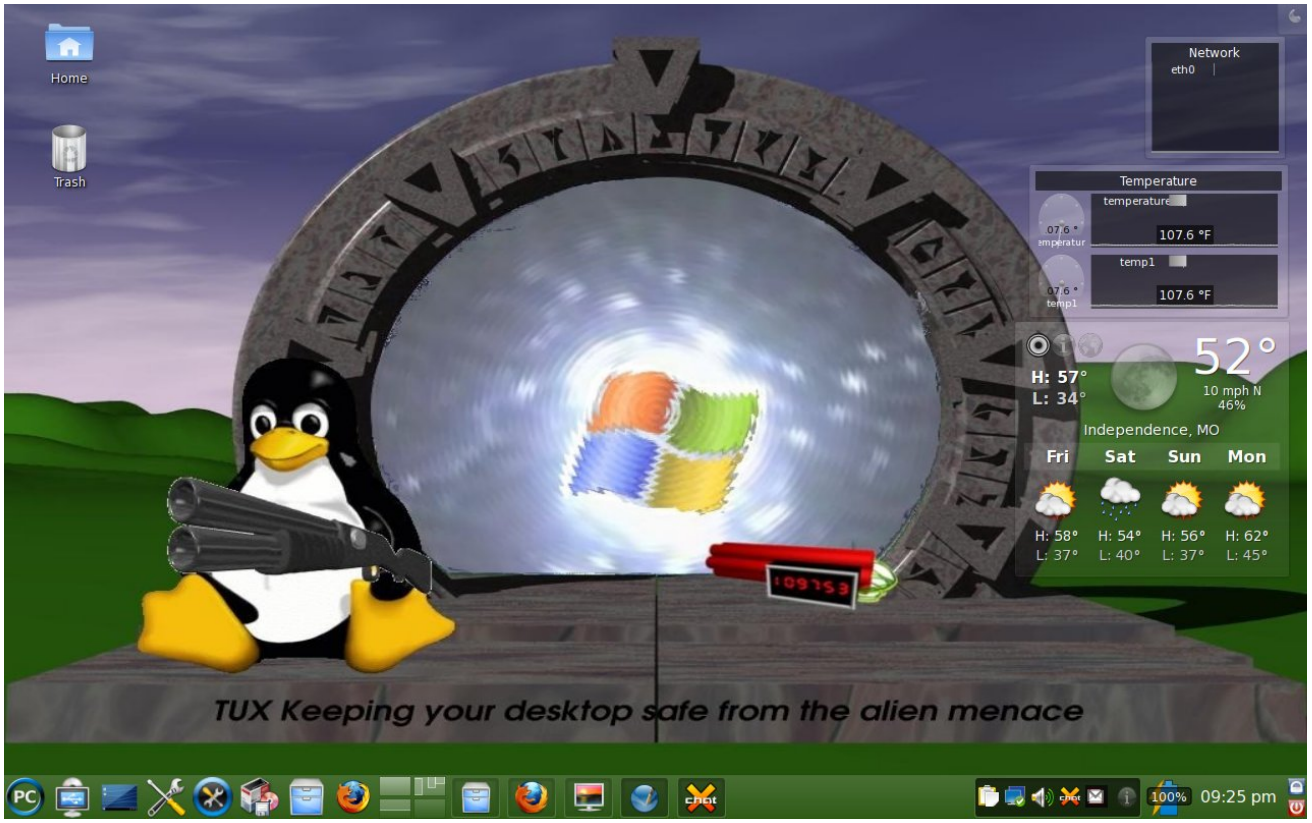
Mark Szorady delivers another dose of his monthly cartoon, **Double Take**, and accompanies it with another of **Mark's Quick Gimp Tips**. We explore another Linux game this month, in **Game Zone: Crack Attack!** Robert Stahl walks us, step-by-step, through **printing from PCLinuxOS to a printer attached to a computer running Microsoft Vista.** Leiche gives us a glimpse at **MediaInfo,** a utility in PCLinuxOS that **provides information on your media files.**

We also have two testimonials this month. One, takes a look at **The Road To KDE 4**, while the other, **Loving The Beta!**, takes a look at running the PCLinuxOS 2010 Beta.

With the beehive of activity surrounding PCLinuxOS, it would be a great idea to check in frequently to both the PCLinuxOS home page, as well as the PCLinuxOS Forum. I suspect that it won't be long before the final release version of PCLinuxOS is ready to roll out the door.

Until then, I wish each of you peace, serenity and tranquility ... and I'll see you next month!

PCLinuxOS Beta Update



PCLinuxOS 2010 Beta 2, running KDE 4.4.1

by Paul Arnote (parnote)

By now, most everyone knows about the PCLinuxOS 2010 Betas that have recently been released. So much so, that PCLinuxOS climbed to the #2 position on DistroWatch.com, based largely on the release of the public beta.

The developers have been incredibly busy working out the few snags that remain, and updating packages – all in preparation for the final release.

The last month has seen the release of the PCLinuxOS 2010 Beta, based on KDE 4.4.1. We have also seen the public release of betas for PCLXDE, Phoenix, Gnome, Zen Mini, and Enlightenment. All were built on the stable core put together by Texstar. The PCLXDE 2010 beta release was remastered by Neal Brooks. Sproggy worked on the XFCE Phoenix remaster, while Siamer worked on assembling the Zen Mini version. The Gnome version was put together by Slick50, and Linuxera created the PCLinuxOS e17 version.

With all the public betas that have been released, there is literally a version of PCLinuxOS for every person, representing all the major desktop environments, that is capable of running on the widest variety of equipment, old and new. For users with older, more modest equipment, PCLXDE, Phoenix and e17 represent very responsive and lightweight desktop environments that run very well on legacy equipment. For users with newer computers, PCLinuxOS 2010 (running KDE 4.4.1) and PCLinuxOS Gnome offer full-featured, responsive desktops.

Stay tuned to the PCLinuxOS home page and the PCLinuxOS Forum for release information. The final versions of all the versions of PCLinuxOS are very near.

The NEW PCLinuxOS Magazine

Created with Scribus 1.3.5

Community: The Heart & Soul of PCLinuxOS

by Paul Arnote (parnote)

Community. We hear that word a lot. So what, exactly, does that word mean? We can turn to the dictionary to get one definition. From dictionary.com, community is defined as the following:

com·mu·ni·ty
/ke'myu nI ti/kuh-myoo-ni-tee

–noun, plural-ties.
1. a social group of any size whose members reside in a specific locality, share government, and often have a common cultural and historical heritage.
2. a locality inhabited by such a group.
3. **a social, religious, occupational, or other group sharing common characteristics or interests and perceived or perceiving itself as distinct in some respect from the larger society within which it exists (usually prec. by the): the business community; the community of scholars.**
4. a group of associated nations sharing common interests or a common heritage: the community of Western Europe.
5. Ecclesiastical. a group of men or women leading a common life according to a rule.
6. Ecology. an assemblage of interacting populations occupying a given area.
7. joint possession, enjoyment, liability, etc.: community of property.
8. **similar character; agreement; identity: community of interests.**
9. the community, the public; society: the needs of the community.

But for those of us who frequent the PCLinuxOS Forum, community isn't just something that is defined in the dictionary. It's much more than that. In fact, and in a large way, it's that sense of community that sets the PCLinuxOS Forum apart from all other support forums. I've taken the liberty to put the dictionary definitions that best fit the PCLinuxOS Forum in bold type.

PCLinuxOS has a reputation, and deservedly so, of being and having the friendliest community around. I know that I, along with many other forum members, have been members of support forums where new users aren't treated with respect and dignity. Rather, new users in other support forums are often treated rudely, disrespectfully, and with disdain. Some veteran members of those forums feel that the new users haven't "paid their dues," or haven't "earned" the right to ask for assistance. Sadly, these same veteran members often forget that they, too, were once new users. Terms like "RTFM" (or Read The Freaking Manual) are often used.

But in the PCLinuxOS Forum, new users are welcomed by members of the PCLinuxOS community. They are treated with respect and dignity. Terms like "RTFM" are not used. Rather, new users are pointed to where to find the information, or their question is answered directly, regardless of how many times it may have been asked before. In doing so, veteran forum members may instruct or inform new users the proper etiquette of searching the forum first.

Guy & Malcolm

It makes a difference, and gives new users a better chance of learning PCLinuxOS, and may even encourage them enough to stick with PCLinuxOS.

In fact, PCLinuxOS gains some new users just because of the widespread reputation of the friendliness of the PCLinuxOS community. They come to PCLinuxOS because they know they won't get "RTFM" as an answer. They know they will be made to feel welcomed, and that they can ask questions without fear.

Does this relieve new users of any responsibility? Absolutely not. New users need to "learn the lay of the land" when they arrive. They need to read through the "stickied" posts, and get a feel for not only what's expected of them as new users, but also the rules that everyone is expected to follow. And, by reading through the "stickied" posts, they can learn not only the right way of asking questions, but also get to know who's who in the forum.

But, the sense of community in the PCLinuxOS Forum goes even deeper than just making new users feel welcome, and treated with respect. Regular and veteran members of the PCLinuxOS Forum have a deep sense of community, that many would argue has become a sense of family. Many of us shared in the joy when Old Polack welcomed his newest grandchild into the world. Many of us shared in GuypronouncedGuynotGuy's joy when he recently became a father for the fourth time. And many of us shared in the grief and profound loss of one of our own, when N1PTT passed away.

Just as with any community or family, there are times when community members or family members get on one anothers' nerves. The PCLinuxOS community is no different. But when it is all said and done, and as most communities and families do, bygones are bygones and the sense of community is restored.

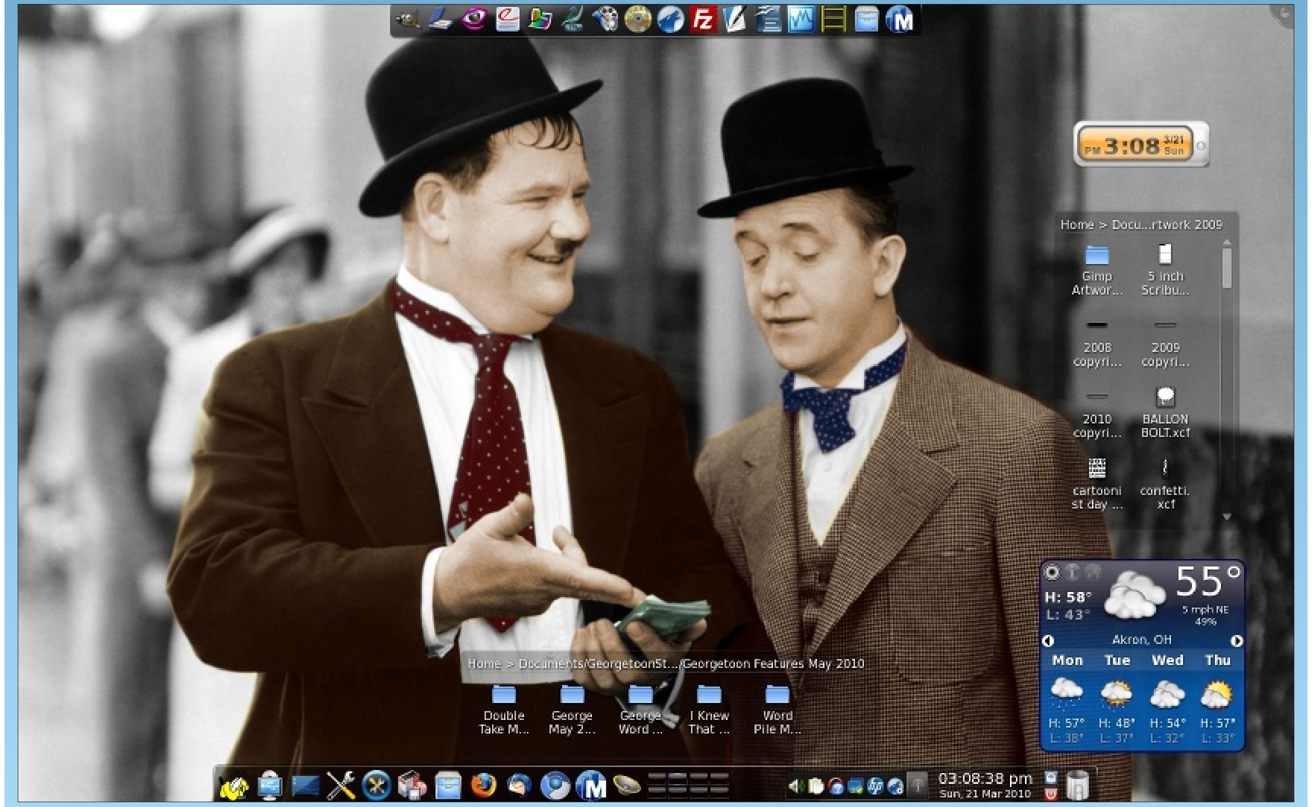
We all have common interests that bind us: our interest in PCLinuxOS, Linux, and FOSS (free open source software). We also have a reputation to uphold, and that is the reputation that PCLinuxOS has as being a warm, welcoming community.

I began thinking of this article topic as I was putting this month's magazine cover together. And, just like many of you, I am proud to be a member of the PCLinuxOS community.

DistroWatch.com
Put the fun back into computing. Use Linux, BSD.

LinFX

Screenshot Showcase



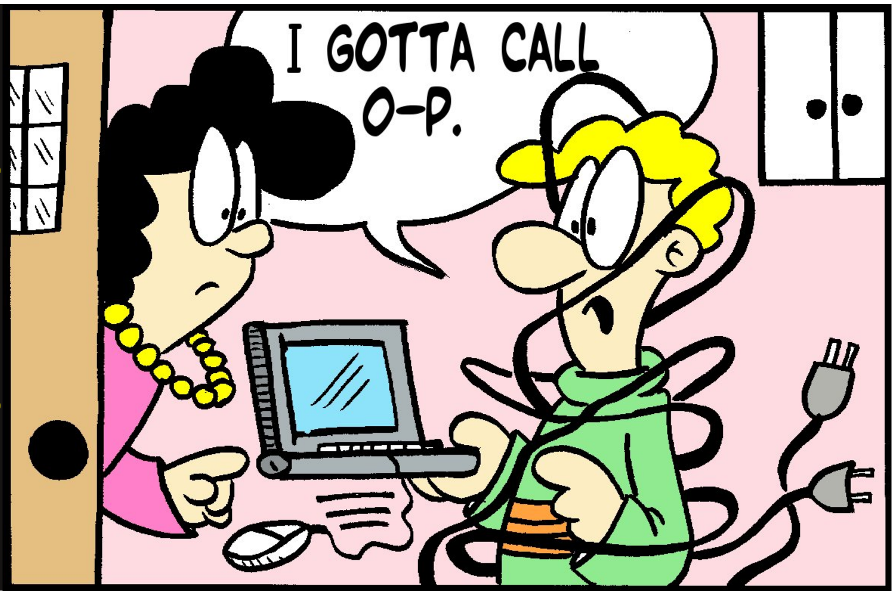
3:08
Home > Docu...rtwork 2009
55°
Akron, OH
Mon Tue Wed Thu
Home > Documents/GeorgetoonSt.../Georgetoon Features May 2010
03:08:38 pm
Sun, 21 Mar 2010

Posted by georgeoon, March 21, 2010, running KDE 4

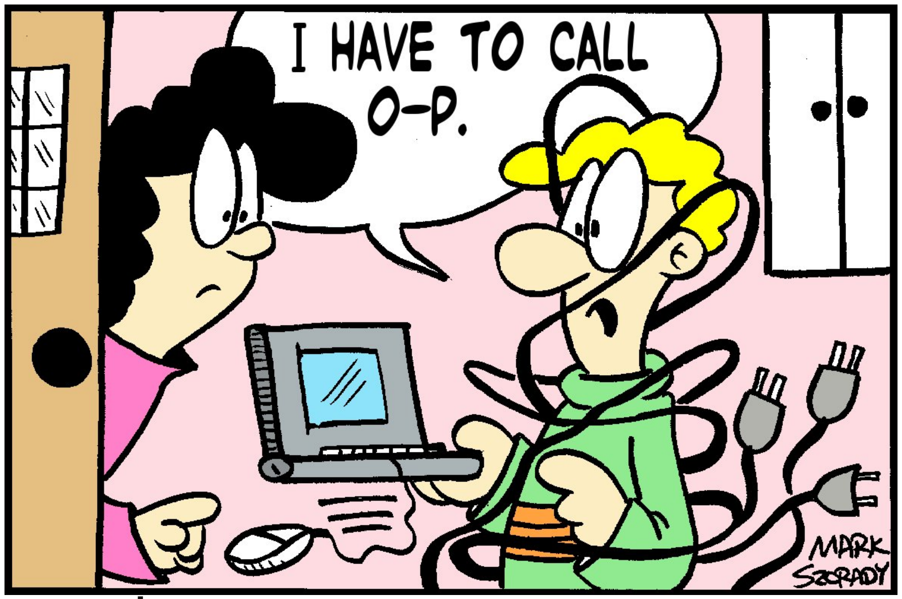
Double Take & Mark's Quick Gimp Tip

Double Take

by Mark Szorady



©2010 Mark Szorady. Distributed by georgetoon.com



Find at least seven differences between cartoons.

Double Take Answers on Page 34

Mark's Quick Gimp Tip

Need to find information about the image you're editing? In Gimp, it's just a click away! Simply select the **Image menu** then **Image properties** (or hit **Alt+Return** on your keyboard). A window will pop up giving you full details (Pixel dimensions, Print size, Resolution, color space, etc.) of the image you're working on. As you can see in the example at right, the information window gives me all the information about the George comic strip I'm working on. It's true that Gimp has a lot of powerful editing tools, but it also contains tools to give you valuable information!

GEORGE® by Mark Szorady
A BARBECUE GRILL?
HA-HA! SOME FOLKS LIKE TO COOKOUT NO MATTER WHAT THE WEATHER!
DID YOU HEAR SOMETHING?
NO. LET'S GET THIS HYPER-DRIVE FIXED!
2/15
MARK SZORADY
© 2010 Mark Szorady. Distributed by georgetoon.com

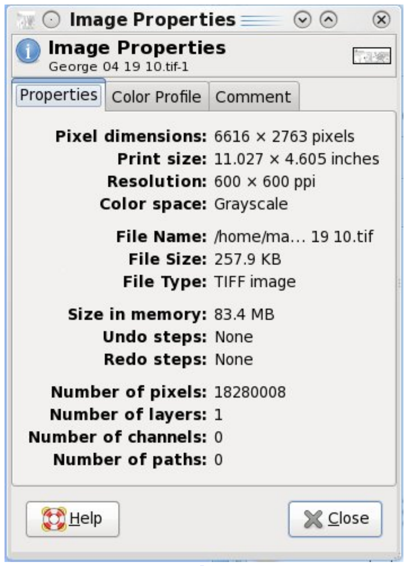


Image Properties
George 04 19 10.tif-1

```
  Pixel dimensions: 6616 × 2763 pixels
        Print size: 11.027 × 4.605 inches
        Resolution: 600 × 600 ppi
       Color space: Grayscale

         File Name: /home/ma... 19 10.tif
         File Size: 257.9 KB
         File Type: TIFF image

    Size in memory: 83.4 MB
        Undo steps: None
        Redo steps: None

  Number of pixels: 18280008
  Number of layers: 1
Number of channels: 0
   Number of paths: 0
```

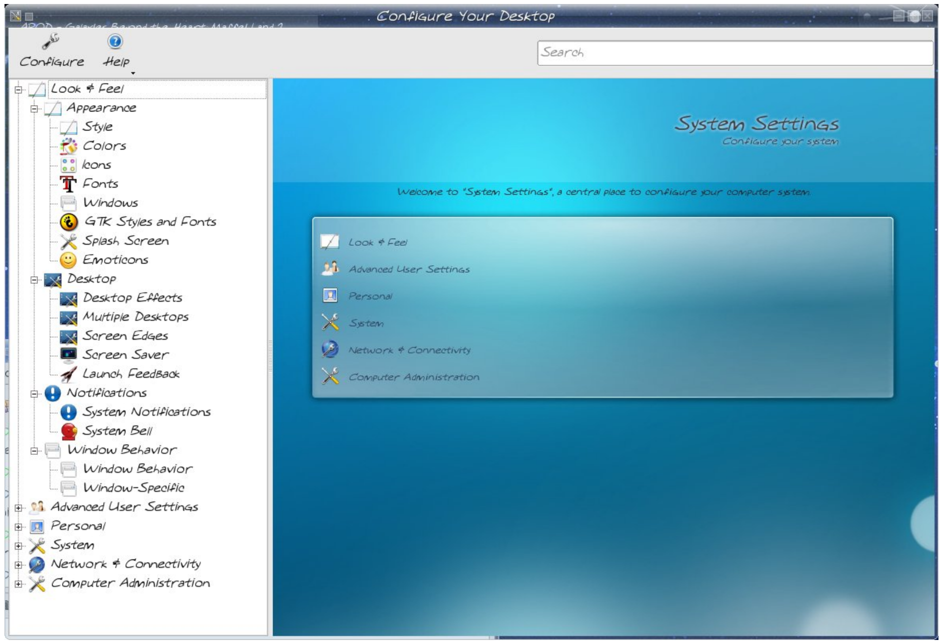
-Mark Szorady is a nationally syndicated cartoonist. His work is distributed by georgetoon.com. Email Mark at georgetoon@gmail.com.

KDE 4: KDE Control Center's New Look

by Meemaw

With the change in KDE from version 3 to version 4, we've seen many changes. Some of us are hesitant to try the new KDE4, or have tried it and don't like it. Others (like me) have jumped in and not looked back. I was nervous about it, but as I find out where more of my configuration aids went, I like it more and more.

Obviously, changes have also been made in the KDE Control Center (commonly seen in the menu as “Configure Your Desktop”). Many of the items we are accustomed to seeing are still there, only rearranged a bit. I think that the old main menu with ten items was condensed into six items by combining them into some more meaningful section. Let's look:

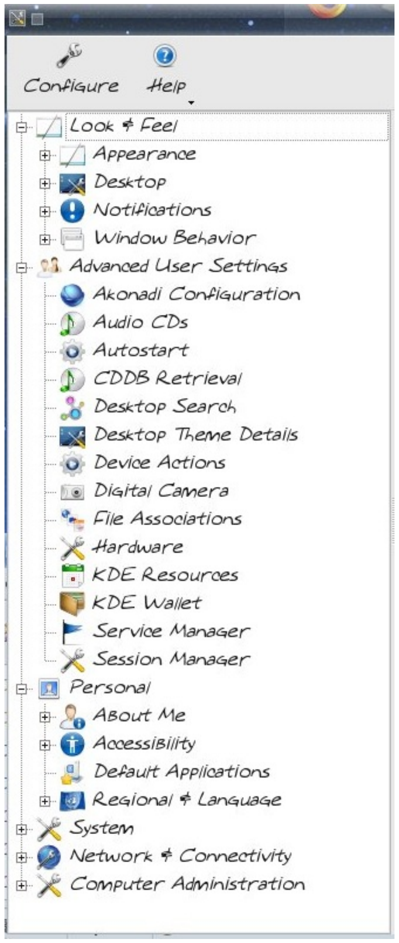


Look & Feel

The first item in the list is 'Look & Feel' rather than 'Appearance & Themes', which was in KDE3. It actually is a combination of 'Appearance & Themes' and 'Desktop', so almost everything is in one spot. One exception is 'Themes' which is now in 'Folder View Settings' or 'Desktop Settings' (in the right-click menu on your desktop.) The other exception is 'Panels,' which was in 'Desktop' in KDE3. The panels are configured in KDE4 by right-clicking the panel and choosing 'Panel Settings.'

Under 'Look & Feel' are four sub-menus; 'Appearance', 'Desktop', 'Notifications' and 'Window Behavior'. Basically, anything you might do to configure how your computer screen looks or even sounds like is in this section. The Appearance section lets you change things like colors, fonts, icon sets and types of emoticons. In Desktop, you can enable desktop effects, configure multiple desktops, designate screen edge actions or configure your screen saver. Notifications lets you assign sounds to various events (like the revelie sound that K3b makes when the burn is finished successfully.) Window Behaviors can also be configured.

Advanced User Settings



The next section is 'Advanced User Settings,' where you can configure things even more. The subsections include Akonadi, Audio CD's, Autostart, Desktop Search, Desktop Theme Details, Device Actions, File Associations, Hardware, Session Manager and KDE Wallet.

Akonadi is a newer KDE4 feature that is supposed to create a common link across your applications (like e-mail, calendar, instant messenger, etc.) which allows you to input information only once and it could be accessed in all apps (instead of having to put e-mail addresses into Thunderbird and again in Kontact.) It is not included in the 2010 beta.

Desktop Search uses Nepomuk which links data across apps so a search will bring up all relevant files no matter which program created them (and a tagged photo in one app will also tag it in another.) The goal is to link data across desktops so collaboration is possible.

Desktop Theme Details will allow you to configure your desktop even more... the subsections allow you to change individual aspects of your theme to suit yourself.

Device Actions lets you configure what you want to happen when you connect a new device to your computer.

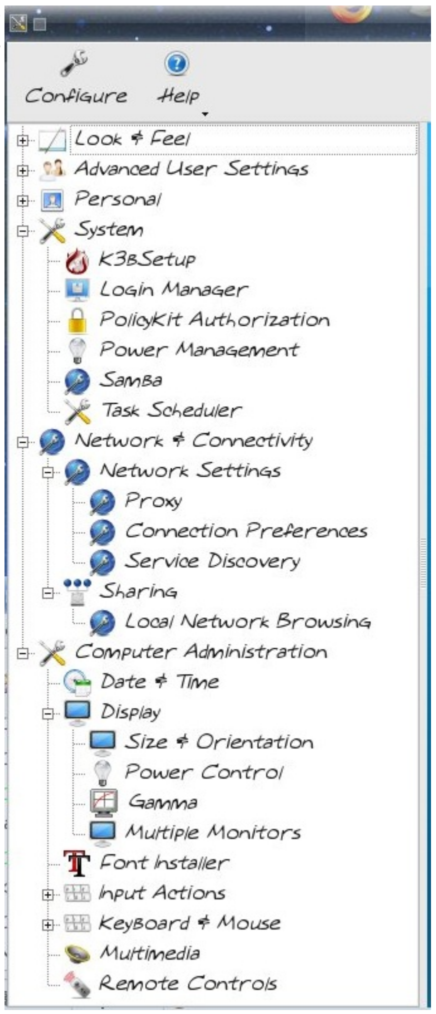
Hardware contains configuration backends for HAL, Network Manager or Bluetooth.

KDE Wallet can be configured here, if you want to use it.

Session Manager lets you configure how KDE starts. If you leave a particular program running and the next day start the previous session, that program will be started as well.

Personal

The Personal section contains items to configure your install specifically for you.... user password and icon and what default folders are used for documents, pictures, etc.; accessibility issues and default applications. Here you also find Regional and Language, which will allow you to change your default language from English to your native language (if it is one of the many languages that PCLInuxOS is capable of using.) You can also configure how your system displays date, time, money, and so on. If your keyboard layout is different from my US keyboard, you can change it here.



System

The next major section is System, which can configure some of the things that PCC does. You will be asked for your root password here as well. Login Manager lets you configure what login screen you see and which users are listed (you can choose not to have root listed if you wish.) Power Management is in this section, as is Samba configuration. Task Scheduler lets you schedule any task you want to be performed on a regular basis.

Network & Connectivity

You can configure proxy, some generic connection preferences (like timeout values) and service discovery here. Sharing with local folders is also configured here.

Computer Administration

Computer Administration includes configuring your date & time, your display (resolution, power control and designation of multiple monitors. The font installer is here, configuration of mouse and keyboard, device preferences for audio and video, and setup and configuration if you have something that runs remotely using infrared.

We can do many things in KDE Control Center. Hopefully, its changes will make things easier to find, once we get used to them.

Enjoy!!

PCLOS GNOME

powered by ENKI

LinPCLus

Vim, Vigor & Vinegar

by ms_meme

Are you already a Vim user? Then this tutorial is not for you. Are you serious about learning Vim so you can do every bit of your work with Vim? Then pass up this article and head straight for an internet search. You will find plenty of sites to help and delight you.

But maybe you thought about trying Vim and read that it is really hard and that the learning curve is steep. Maybe you looked at some of the on-line tutorials and immediately clicked the back arrow. Maybe you opened up Vim and thought, "I don't have enough vim and vigor for all this."

If so, just maybe this article is for you. I did all three of the above. Then I decided that I was going to give Vim a try. This really isn't a tutorial. It is just a chat about my experiences, the sweet and the sour, with Vim. I will attempt to give you a little taste of Vim without your having to read pages and pages of tutorials, as if you are going to use Vim day and night.

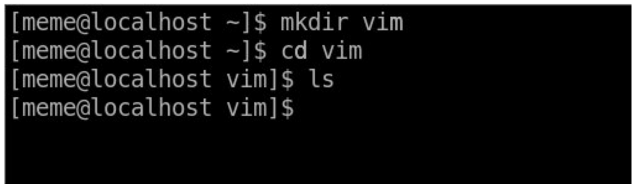
You don't know what Vim is? It is a little text editor full of verve, vigor, vitality, vivaciousness, vivacity and a dash of zip. Of course you don't have to put up with all of that in order to try it. Vim may already be loaded into your programs. More Applications/Editors/Vi editor. Vi is the original application. Vim is Vi IMproved. If it is not there, look for it in Synaptic.

One of my New Year's resolutions was to spend more time in the terminal. So if you try following along with me, everything I do with be done there. Vim has a GUI that can also be used.

The power of Vim is using the keyboard to do all the commands, instead of the mouse. This could save time or just be fun. It has lots and lots and lots of commands. I get lost with most of them, but you can still learn something about Vim with just a few.

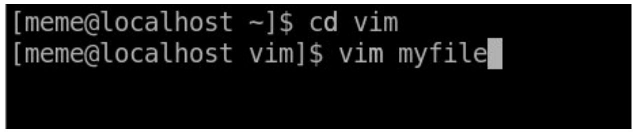
Open up a terminal. You will want to make a new directory/folder called vim. This will be where you will keep your saved files for this Vim practice. **mkdir** is the command to make that directory. To change to the vim directory, type **cd vim**.

In the illustration below, I typed **ls** after changing to the vim directory. **ls** is the command for a simple listing of what is in the directory. I like to practice those terminal commands to be true to my New Year's resolution. All is well, as nothing is listed in the vim directory.



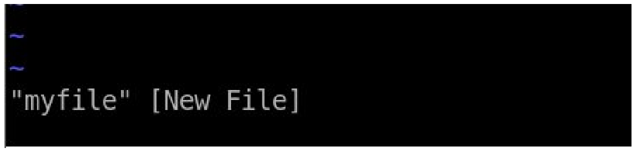
```
[meme@localhost ~]$ mkdir vim
[meme@localhost ~]$ cd vim
[meme@localhost vim]$ ls
[meme@localhost vim]$
```

Now that we are in the vim directory, we are ready to make a file. Type **vim <filename>** then press enter. You may name it anything, but it is best to use no spaces in the name. The name I have chosen is **myfile**.



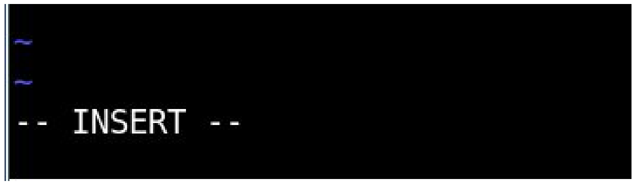
```
[meme@localhost ~]$ cd vim
[meme@localhost vim]$ vim myfile
```

The vim text editor will now open. You will see a page with a row of tildes ~ to the left. At the bottom, you will see the name of your file showing it is a new file. Vim has several modes. The default mode is NORMAL mode. It is here that the keyboard commands are given. Instead of taking your hands off the keyboard and moving them back and forth to the mouse, you will type a combination of keys to move around or edit your document. You will not be able to enter text to your document in NORMAL mode.



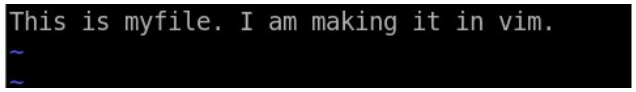
```
~
~
"myfile" [New File]
```

In order to type/edit text, you must be in INSERT mode. There are several ways to get to INSERT mode. One way is to press the **i** key. After you do this, you will see the word INSERT at the bottom left of the screen.



```
~
~
-- INSERT --
```

You can now type/insert whatever text you want.

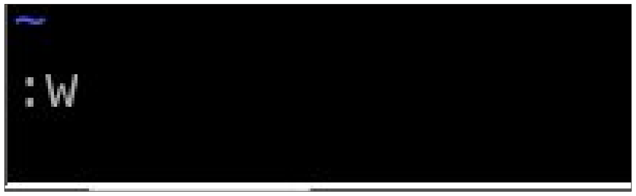


```
This is myfile. I am making it in vim.
~
~
```

You have now made a text file, but it is not yet saved. To save a file without closing, you will have to return to NORMAL mode where the commands are given. **To return to NORMAL mode press the Esc key**. You will notice that the INSERT disappears from the bottom of the screen and you will no longer be able to enter text.

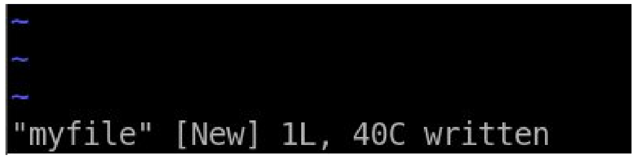
One of the things I found a bit hard was remembering if I am in INSERT or NORMAL mode. Of course, it tells you right at the bottom. I think it is more nervousness than forgetfulness. Your hands get jittery as you try to get into the Vim mood. I thought it was a bit awkward to remove my left hand from the keyboard position to tap the Esc key to get to NORMAL. You have to do this quite often. But then removing your right hand from the keyboard to get to the mouse is about eight times the distance. I can justify working in Vim right now with that ratio.

To save a file without closing, type **:** in the NORMAL mode. You will see it appear at the bottom of the screen. Next, type **w** as the command to write to disk.



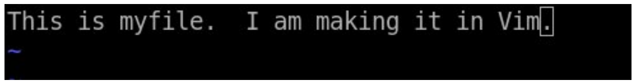
```
~

:w
```

After you press Enter, you will see your file has now been written/saved.



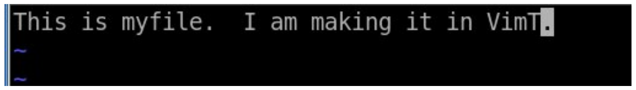
```
~
~
~
"myfile" [New] 1L, 40C written
```

Since you saved a file but did not close it, you probably want to do some more work with it. And that means returning to INSERT mode. As shown above, the letter **i** can be used to accomplish this. But before typing **i**, I want you to see what happens when I do it. I have pressed i and my cursor is on the letter/space that I last typed.



```
This is myfile.  I am making it in Vim.
~
```

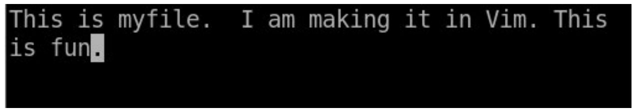
You can see below when I entered a **T** for my next text, the **T** was placed before the letter/space that the cursor is on. I don't want it there. This is because the command **i** inserts text before the cursor. If this happens, you may not know a command to get it in the right place.



```
This is myfile.  I am making it in VimT.
~
~
```

In order to get my next sentence started in the right place, I can use another INSERT command. I returned to the NORMAL mode by pressing **Esc**. This time, I typed **a** to append text. The **a** command will also bring you to INSERT mode. **a** inserts text after the cursor. I can then continue to type. There are other ways to get your text inserted where you want it, but this just shows you the difference between **i** and **a**.

I have now added more text.



```
This is myfile.  I am making it in Vim. This
is fun.
```

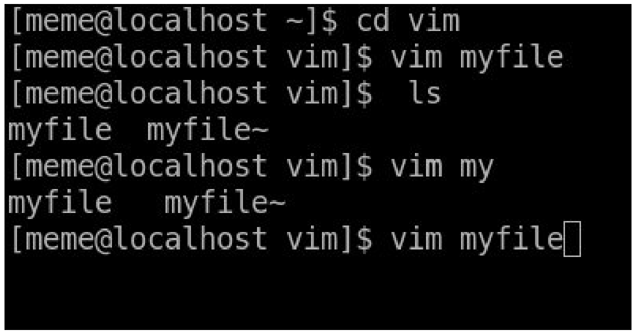
Here are some basic commands to try before saving and exiting Vim. You might want to type several lines of text before trying them. Remember you must press Esc to be in Normal mode before using the commands.

```
h   moves cursor left
j   moves cursor down
k   moves cursor above
l   moves cursor right
i   inserts text before cursor
I   inserts text at beginning of a line
a   inserts text after cursor
A   inserts text at the end of a line
x   deletes a letter when cursor is on top of it
dw  deletes an entire word when cursor is placed
    on first letter of word
d$  deletes to the end of a line from where you
    placed cursor.
u   undoes a command
0   moves cursor to beginning of a line
$   moves cursor to the end of the line
```

While working on your file, you will want to save often using the command **:w**. When you are finished, you can save and quit the Vim program by typing **:wq** in Normal mode.

Since we began in the vim directory, the file will be saved there. Of course, you can save it to another directory, but that is another lesson. I have saved and quit using **:wq**. This returns me to the terminal. I am still in the vim directory. This time when I type **ls**, I see that myfile is in the list and all is well.

I can then exit the terminal or cd to other places. If I want to work again on the file, I will cd to the vim directory and type **vim <filename>**. Of course I can access **'myfile'** from anywhere using paths. That is another lesson.



```
[meme@localhost ~]$ cd vim
[meme@localhost vim]$ vim myfile
[meme@localhost vim]$  ls
myfile  myfile~
[meme@localhost vim]$ vim my
myfile   myfile~
[meme@localhost vim]$ vim myfile
```

Perhaps, while working on a file, you have made changes but decide to quit without saving these changes. **:q** That command may not always work. Try **:q!** The **!** tells Vim that you mean it.

I hope you have learned a little about Vim. There is a Vim tutorial with the Vim install. In a terminal, type **vimtutor**. It says it is a 30 minute tutorial. I have been working on it for many months and am only half way through. I think it depends how much practice you want to do with each lesson. I like to be sure I can perform each new command with ease before adding another.

My article is entitled Vim, Vigor and Vinegar. I think you can understand that Vim is a very vigorous program. But what about the vinegary part? I found Vim to be like a pickle. It can be a bit sour, but I am always wanting another bite.

ms_meme & vi

ms_meme read Archie's comments
And wondered what was vi
It must be something wonderful
And why not give it a try

But first she had to find it
This mysterious thing called vi
And off she went to her programs
As she gave a little sigh

She started searching low
Then kept on searching high
And finally in a terminal
She found the editor vi

She pressed on the ESC key
And then tried the letter 'i'
And then with a 'h''j''k''l'
She was flying high with vi

Her fingers never left the keyboard
As she wrote css in vi
She never even touched the arrows
Although at times she wanted to cry

She googled some tutorials
And herself she did apply
And she thanks her friend Archie
For leading her to vi

ms_meme may be old
But she's still a little spry
And if it takes the rest of her years
She'll conquer dear old vi

Vim Vim Veree: A ms_meme Tribute To Vim

viminey viminey vimveree

I am as lucky as lucky can be

viminey viminey vimveroo

I use to make songs for you

Come everyone try too

viminey viminey vimveree

If you want luck do just like me

viminey viminey vimveroo

Open a konsole it's not hard to do

Start using you'll be lucky too

viminey viminey vimveroo

Hear what I tell you it is all true

viminey viminey vimveree

is lucky you will agree

And best of all is free

OGG

MP3

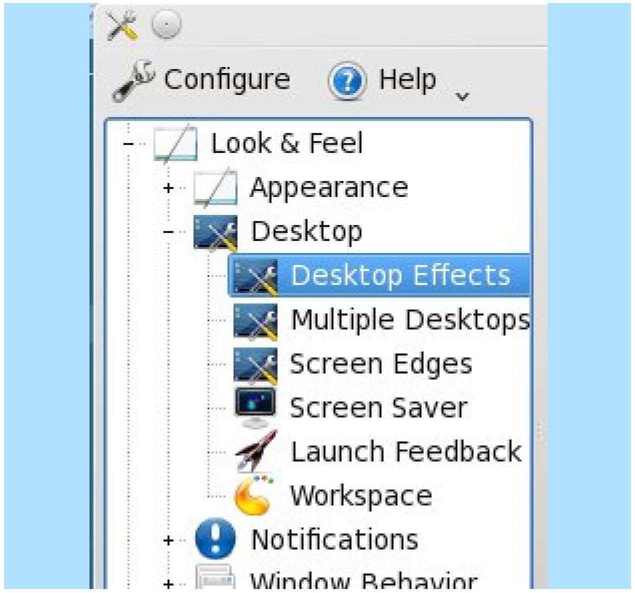
KDE 4: KWin's Desktop Effects a Winner

by Paul Arnote (parnote)

One of the newest features of KDE 4 (compared to KDE 3.5.10) is the window compositing that is built into KWin. Before, it was the relegated to the realm of those who took the time to learn and use Compiz Fusion. But now, window compositing is easily within reach of most users. In KDE 4, it's referred to as "Desktop Effects." Veteran Compiz users will most likely view the KDE 4 Desktop Effects as "child's play," but for the rest of us, they represent some very nice effects that help break the monotony of what we've become accustomed to on our desktops.

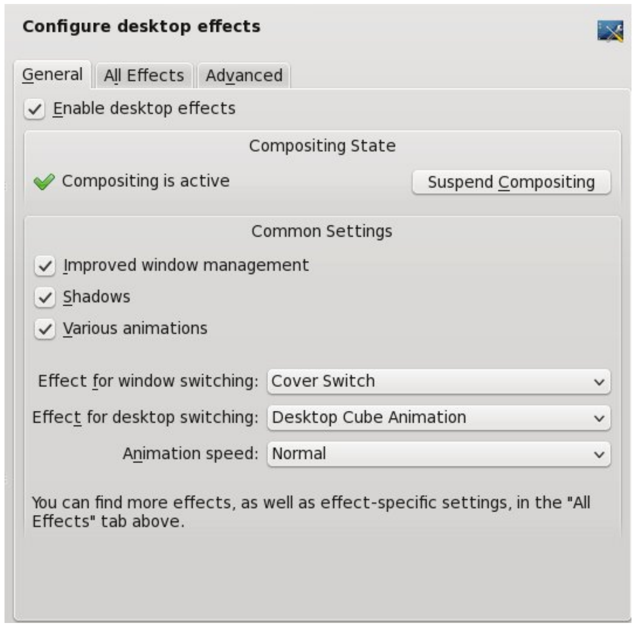
Before getting started with explaining how the new KDE 4 Desktop Effects works, you should make sure that your computer's graphic card is up to the task. While official hardware requirements are exceptionally difficult to find (if they even exist), user reports indicate that you will need a minimum of 64 MB of video RAM to successfully run the Desktop Effects feature of KDE 4. Reports abound from users attempting to run Desktop Effects with 32 MB of video RAM and less. Additionally, your video card must support the OpenGL 3D graphics standard. Without both, you risk locking up your computer, and the only way to regain control is to do a hard reboot (a sometimes dangerous proposition). I can attest to the lock ups, as I just had to try Desktop Effects on my old Pentium 3 with only 8 MB of video memory on a GPU not capable of OpenGL. (Hey! I had to try!)

To get started setting up Desktop Effects in KDE 4, go into the KDE Control Center (KCC ... also named "Configure Your Desktop"), and select Look & Feel » Desktop » Desktop Effects.



The first tab in the window that opens to the right of the selection tree in KCC is the General tab. To enable Desktop Effects, simply place a check mark in the box at the top. It is turned off by default. Hit the "Apply" button at the bottom of the window to make the selection active. Under "Compositing State," is confirmation that compositing has been turned on, along with a button to suspend compositing (you may wish to temporarily suspend compositing when you are doing CPU and memory intensive tasks). You can also suspend and resume compositing by pressing Alt + Shift + F12 from the keyboard.

Under Common Settings, you can opt to turn on "Improved window management," "Shadows" and "Various animations." It is the latter that provides the best eye candy. With animations activated, you can choose from several default window switching effects and several default effects for desktop switching. In the screen shot above, I have selected the "Cover Switch" effect to use for window switching, and the "Desktop Cube Animation" as the effect to use for desktop switching. Below is an example of some of the default effects you can choose from for window switching. (I don't have any pictures of desktop switching, but you can leave that as an exercise of discovery for yourself).



✓ Enable desktop effects

Compositing State

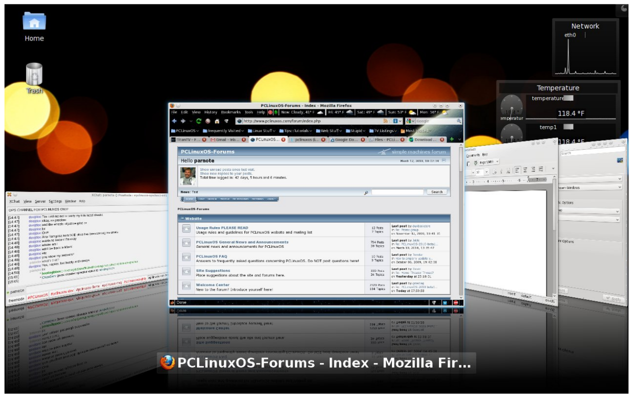
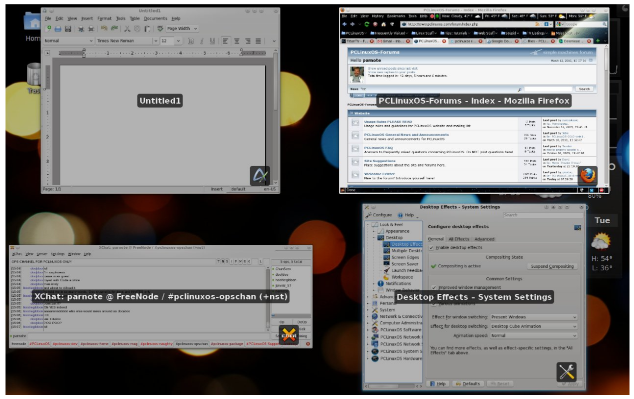
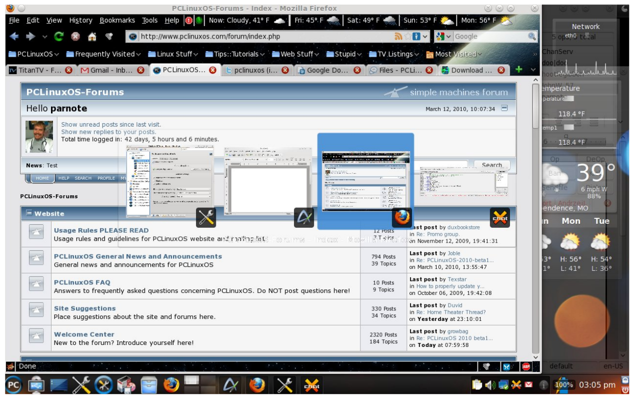
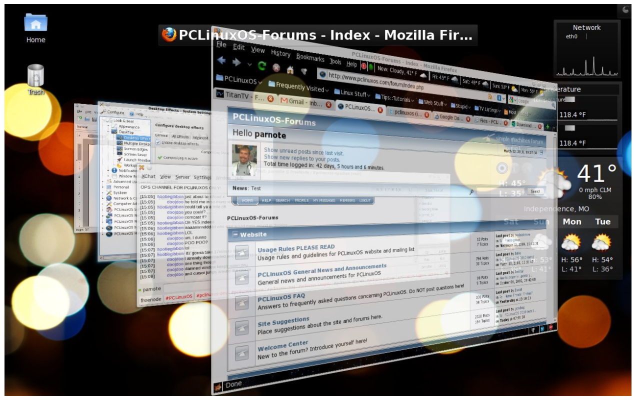
✓ Compositing is active

Common Settings

✓ Improved window management
✓ Shadows
✓ Various animations

Effect for window switching: Cover Switch
Effect for desktop switching: Desktop Cube Animation
Animation speed: Normal

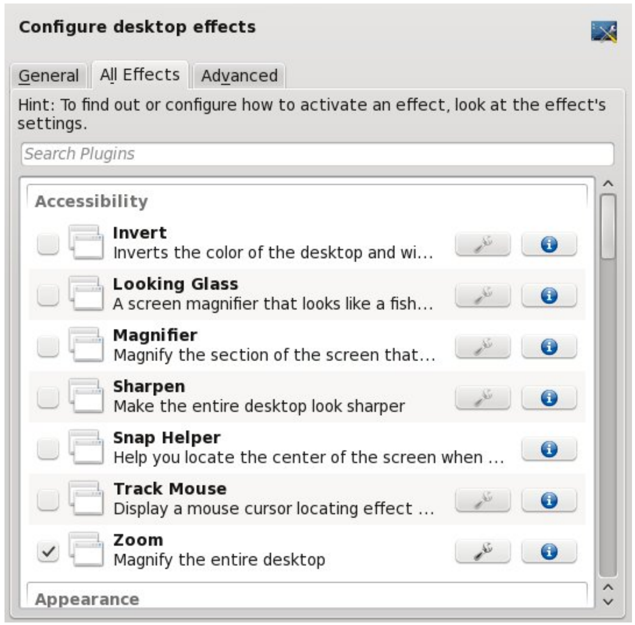
You can find more effects, as well as effect-specific settings, in the "All Effects" tab above.

Window switching effects: Cover Switch (top L), Box Switching (bottom L), Present Windows (top R), and Flip Switch (bottom R)

From the second tab, "All Desktop Effects," you can choose from a host of other special effects. Some you may wish to explore are exploding windows when you close them, or magic lamp, which makes your windows appear as if they are being sucked into Aladdin's lamp when you minimize them. Enable the desired effect by placing a check mark in the box to the left of the title of the effect you want to use. And feel free to play around with those effects and customize your Desktop Effects.

The last tab, labeled "Advanced," is where you can (of course) make some more advanced settings as to how Desktop Effects are displayed on your



Configure desktop effects

General | All Effects | Advanced

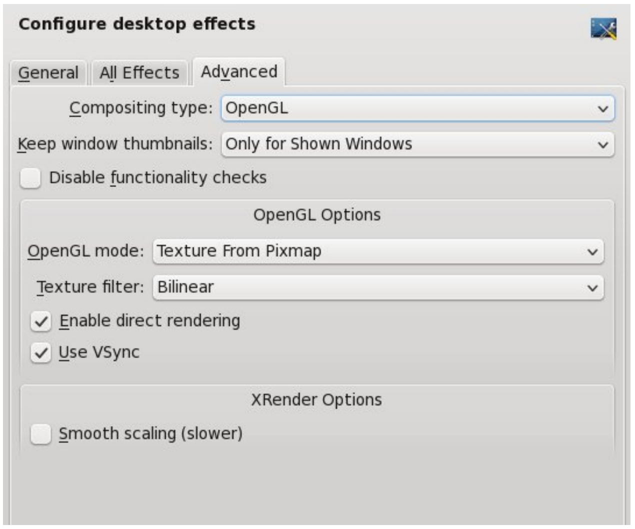
Hint: To find out or configure how to activate an effect, look at the effect's settings.

Search Plugins

Accessibility

- [ ] **Invert** Inverts the color of the desktop and wi...
- [ ] **Looking Glass** A screen magnifier that looks like a fish...
- [ ] **Magnifier** Magnify the section of the screen that...
- [ ] **Sharpen** Make the entire desktop look sharper
- [ ] **Snap Helper** Help you locate the center of the screen when ...
- [ ] **Track Mouse** Display a mouse cursor locating effect ...
- [✓] **Zoom** Magnify the entire desktop

Appearance



Configure desktop effects

General | All Effects | Advanced

Compositing type:	OpenGL
Keep window thumbnails:	Only for Shown Windows

- [ ] Disable functionality checks

OpenGL Options

OpenGL mode:	Texture From Pixmap
Texture filter:	Bilinear

- [✓] Enable direct rendering
- [✓] Use VSync

XRender Options

- [ ] Smooth scaling (slower)

computer. Most users will have little or no need to mess with these default settings.

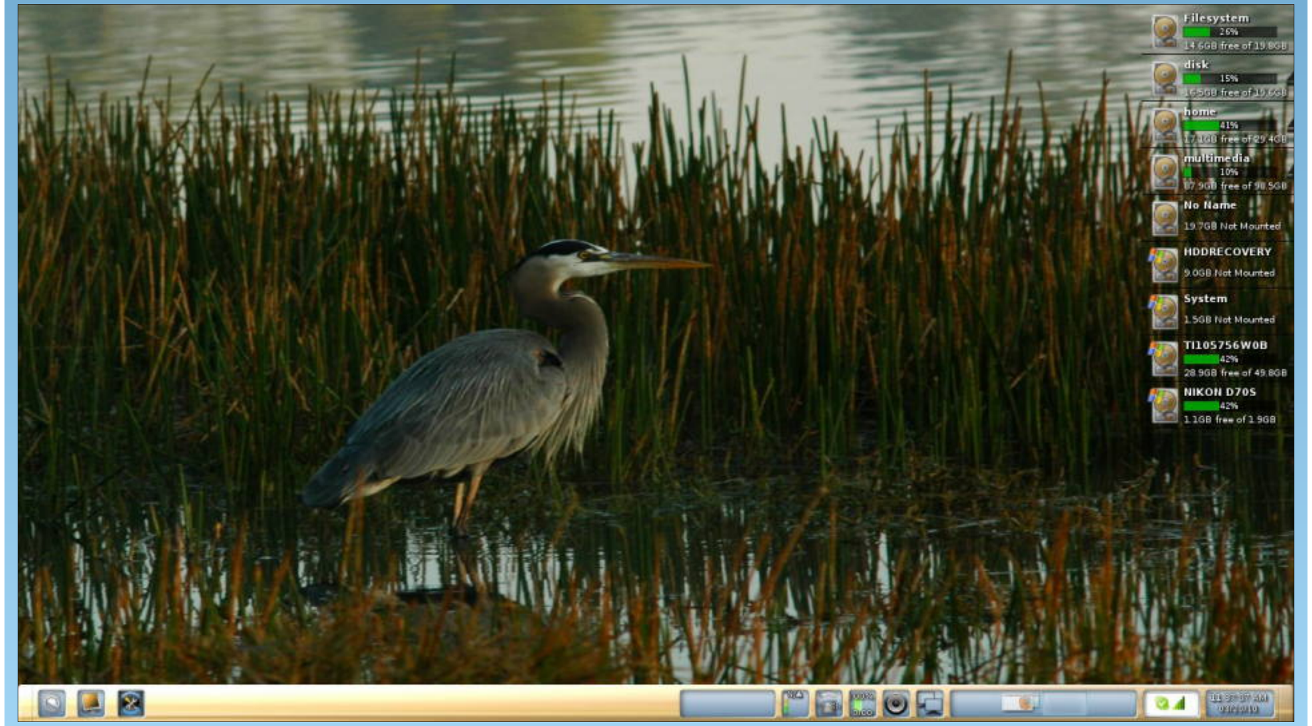
Although Desktop Effects doesn't have as many features as Compiz, it may go a long way to opening up window compositing to users who have never used or considered it before. Will it fundamentally enhance your productivity? Probably not. But then again, since it breaks the typical monotony of the desktop that most of us have become accustomed to, it just may help make you more productive. If you have the hardware to support it, you should give it a try. You can always turn it off again if you get tired of it.

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.

Screenshot Showcase



Posted by linuxera, March 20, 2010, running e17

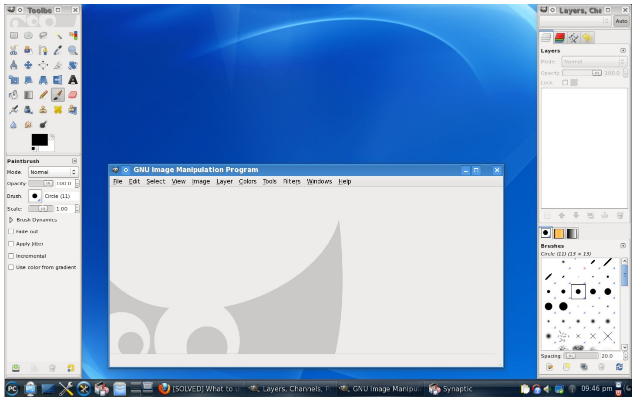
Gimp From A Beginner's Perspective: Part 1

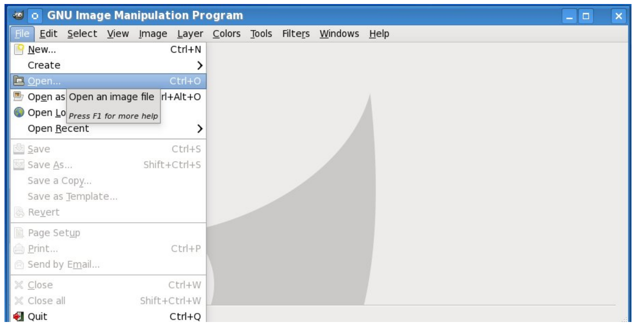
by Dan Malewski (Blndsyd)

First of all I would like to introduce myself to everyone. My name is Dan, and I go by Blndsyd on the forums. I'm a converted Microsoft Windows and Photoshop user. I am a self learner of all my computing needs. In using Photoshop for the last 6 years, I've learned enough to be dangerous. In no way am I a pro in Photoshop or GIMP, but I can find my way around both of these great photo manipulation programs. In fact, I was using GIMP for a few short days before volunteering my time to the magizine. So you and I will be learning some of this together.

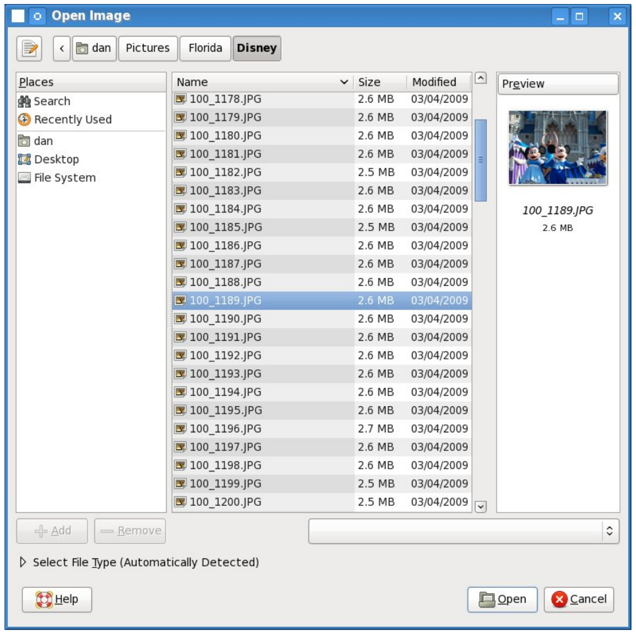
My goal here will be to show you some basics, followed by (hopefully) some more advanced usage all while keeping you awake. Remember, I am a beginner as well so let's learn a bit together.

Shall we start from the beginning? I will assume you have a working GIMP program on your install of your favorite OS. By this, of course, I mean PCLinuxOS. Let's fire it up. Your screen will look something like this.


Toolbox
Layers, Cha
GNU Image Manipulation Program
File Edit Select View Image Layer Colors Tools Filters Windows Help

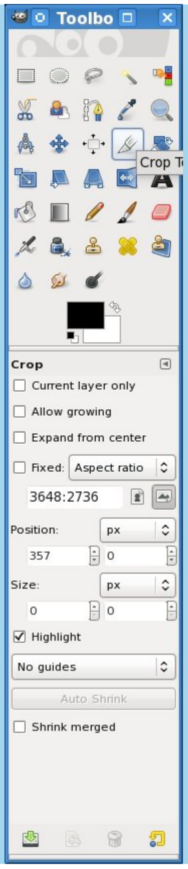


How about we open one of your favorite pictures. In your middle "window," select File » Open.



Name	Size	Modified
100_1178.JPG	2.6 MB	03/04/2009
100_1179.JPG	2.6 MB	03/04/2009
100_1180.JPG	2.6 MB	03/04/2009
100_1181.JPG	2.6 MB	03/04/2009
100_1182.JPG	2.5 MB	03/04/2009
100_1183.JPG	2.6 MB	03/04/2009
100_1184.JPG	2.6 MB	03/04/2009
100_1185.JPG	2.5 MB	03/04/2009
100_1186.JPG	2.6 MB	03/04/2009
100_1187.JPG	2.6 MB	03/04/2009
100_1188.JPG	2.6 MB	03/04/2009
100_1189.JPG	2.6 MB	03/04/2009
100_1190.JPG	2.6 MB	03/04/2009
100_1191.JPG	2.6 MB	03/04/2009
100_1192.JPG	2.6 MB	03/04/2009
100_1193.JPG	2.6 MB	03/04/2009
100_1194.JPG	2.6 MB	03/04/2009
100_1195.JPG	2.6 MB	03/04/2009
100_1196.JPG	2.7 MB	03/04/2009
100_1197.JPG	2.6 MB	03/04/2009
100_1198.JPG	2.6 MB	03/04/2009
100_1199.JPG	2.5 MB	03/04/2009
100_1200.JPG	2.5 MB	03/04/2009

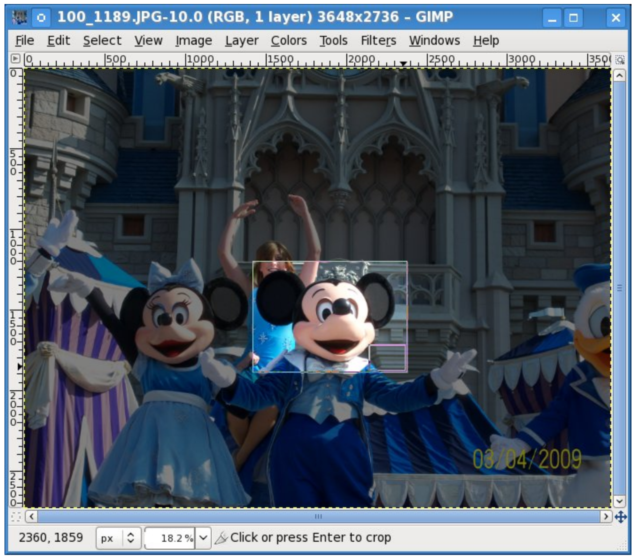
Now finally let's select your picture. Double click the file and open the picture.



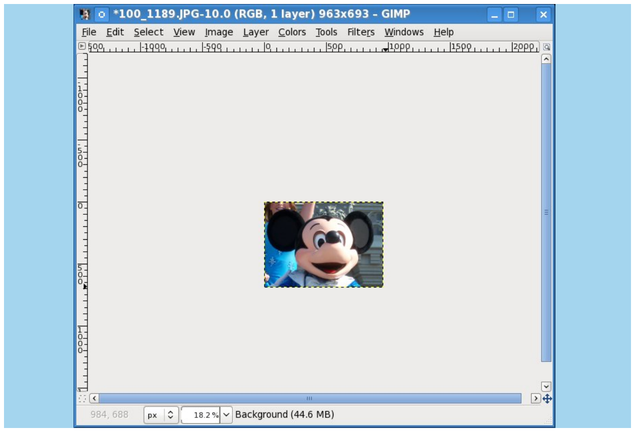
[ ] Current layer only
[ ] Allow growing
[ ] Expand from center
[ ] Fixed: Aspect ratio
3648:2736
Position: px
357 0
Size: px
0 0
[✓] Highlight
No guides
Auto Shrink
[ ] Shrink merged

I will use this same picture for a few things we will talk about. The first will be what I will call "slicing" out part of the picture. Lets select the knife in the "toolbox". Of course GIMP calls it the "Crop Tool".

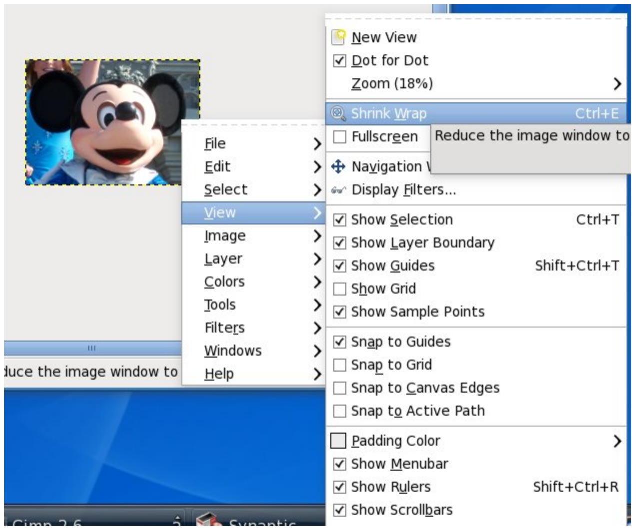
Now let's "slice" out the part of the picture you really want. Using your cursor draw a box around the part you want. For this I want Mickey's head.


100_1189.JPG-10.0 (RGB, 1 layer) 3648x2736 - GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
2360, 1859 px 18.2% Click or press Enter to crop

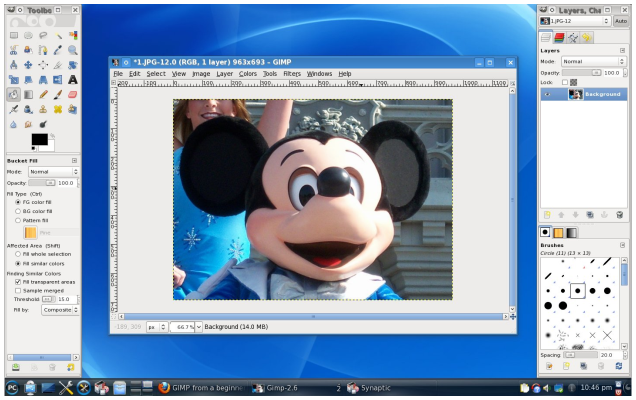
See the "box" at the lower right had corner of our slice? Click it. Now you have this.


*100_1189.JPG-10.0 (RGB, 1 layer) 963x693 - GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
984, 688 px 18.2% Background (44.6 MB)

The only thing we have left is to get rid of the extra space around the layer. Right click your image and select View »Shrink Wrap.



Now all you have to do is save your new image. Click File » Save As. Give it a unique name, and bang! You have a brand new image.



While it may seem like a rather simplistic beginning, we have to start somewhere. Hopefully as we move on, these articles will become more challenging.

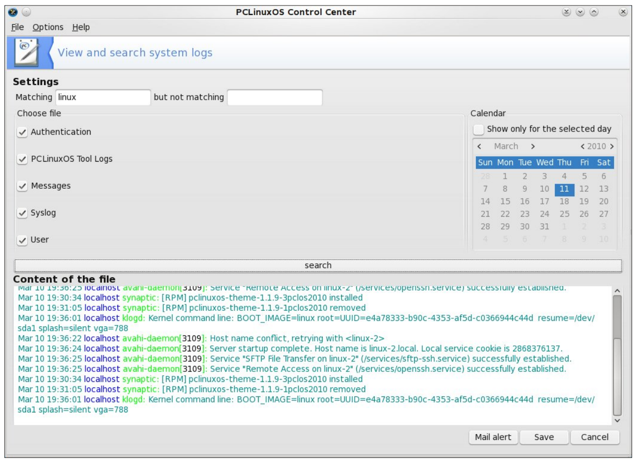
Next time I plan to show how to place some text and give the text some good effects. Feel free to email me at blndsyd@gmail.com if you would like to see some specific tutorials.

Thanks and see you next time.

KDE 4: KSystemLog Reveals Your Log Files

by Paul Arnote (parnote)

Most of you reading this already know that PCLinuxOS is an incredibly stable operating system. But, as with any computer operating system, things may sometimes go awry. And when it does, you are likely to find clues to your problem in one of the many log files maintained on a regular basis in PCLinuxOS. Fortunately, PCLinuxOS users have two excellent ways to access those log files: either through the PCLinuxOS Control Center (PCC), or through a KDE 4 utility, called KSystemLog.



Settings

Matching: linux — but not matching: (empty)

Choose file: ✓ Authentication, ✓ PCLinuxOS Tool Logs, ✓ Messages, ✓ Syslog, ✓ User

Calendar: ☐ Show only for the selected day — March 2010, 11 selected

Content of the file

```
Mar 10 19:36:25 localhost avahi-daemon[3109]: Service "Remote Access on linux-2" (/services/openssh.service) successfully established.
Mar 10 19:30:34 localhost synaptic: [RPM] pclinuxos-theme-1.1.9-3pclos2010 installed
Mar 10 19:31:05 localhost synaptic: [RPM] pclinuxos-theme-1.1.9-1pclos2010 removed
Mar 10 19:36:01 localhost klogd: Kernel command line: BOOT_IMAGE=linux root=UUID=e4a78333-b90c-4353-af5d-c0366944c44d  resume=/dev/
sda1 splash=silent vga=788
Mar 10 19:36:22 localhost avahi-daemon[3109]: Host name conflict, retrying with <linux-2>
Mar 10 19:36:24 localhost avahi-daemon[3109]: Server startup complete. Host name is linux-2.local. Local service cookie is 2868376137.
Mar 10 19:36:25 localhost avahi-daemon[3109]: Service "SFTP File Transfer on linux-2" (/services/sftp-ssh.service) successfully established.
Mar 10 19:36:25 localhost avahi-daemon[3109]: Service "Remote Access on linux-2" (/services/openssh.service) successfully established.
Mar 10 19:30:34 localhost synaptic: [RPM] pclinuxos-theme-1.1.9-3pclos2010 installed
Mar 10 19:31:05 localhost synaptic: [RPM] pclinuxos-theme-1.1.9-1pclos2010 removed
Mar 10 19:36:01 localhost klogd: Kernel command line: BOOT_IMAGE=linux root=UUID=e4a78333-b90c-4353-af5d-c0366944c44d  resume=/dev/
sda1 splash=silent vga=788
```

The PCC utility is located under System » Administration Tools » View and Search System Logs, when you launch PCC. From it, you can select which log files you want to search through. In the example screen shot, I've marked all five categories of log files.

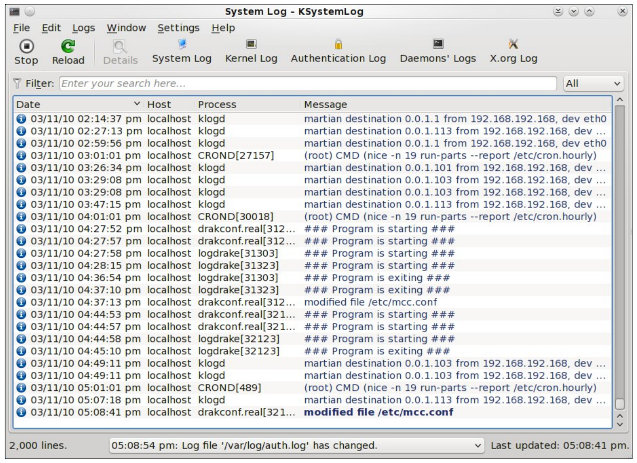
At the top of the window, you type in the term you are searching for. I've used the term "linux," but you can just as easily type in "eth0" or "video" or any other term you want to search for.

Drop down vertically to the middle of the window, and click on the "Search" button to conduct your search. The bottom portion of the window shows your search results, under "Content of the file."

At the very bottom of the window, you are given the choices to "Mail alert," "Save" the data, or "Cancel" the operation.

You can also restrict your search to only include data specific to the selected day, in the upper right corner of the window by simply checking the box above the calendar that is displayed. You can also further refine your search by specifying a term you wish to exclude from your search results, by filling in that term in the "but not matching" box.

Of course, the PCC method of viewing and searching your log files is available to every user running ***any*** version of PCLinuxOS, regardless of the desktop environment you are using. Just remember you will have to supply a search term. However, if you are running KDE 4, you have another choice that allows you to view your log files, without specifying a search term. You can use **KSystemLog**, which is installed from Synaptic as part of the KDEAdmin4 package. KSystemLog is designed to make it easy for beginning KDE users to find the various system logs on their system, but also designed to make the log files accessible and useful for more experienced users.

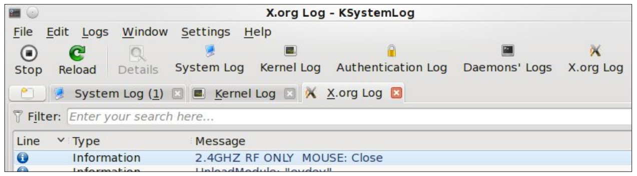


```
Date                  Host       Process              Message
03/11/10 02:14:37 pm  localhost  klogd                martian destination 0.0.1.1 from 192.168.192.168, dev eth0
03/11/10 02:27:13 pm  localhost  klogd                martian destination 0.0.1.113 from 192.168.192.168, dev ...
03/11/10 02:59:56 pm  localhost  klogd                martian destination 0.0.1.1 from 192.168.192.168, dev eth0
03/11/10 03:01:01 pm  localhost  CROND[27157]         (root) CMD (nice -n 19 run-parts --report /etc/cron.hourly)
03/11/10 03:26:34 pm  localhost  klogd                martian destination 0.0.1.101 from 192.168.192.168, dev ...
03/11/10 03:29:08 pm  localhost  klogd                martian destination 0.0.1.103 from 192.168.192.168, dev ...
03/11/10 03:29:08 pm  localhost  klogd                martian destination 0.0.1.103 from 192.168.192.168, dev ...
03/11/10 03:47:15 pm  localhost  klogd                martian destination 0.0.1.113 from 192.168.192.168, dev ...
03/11/10 04:01:01 pm  localhost  CROND[30018]         (root) CMD (nice -n 19 run-parts --report /etc/cron.hourly)
03/11/10 04:27:52 pm  localhost  drakconf.real[312... ### Program is starting ###
03/11/10 04:27:57 pm  localhost  drakconf.real[312... ### Program is starting ###
03/11/10 04:27:58 pm  localhost  logdrake[31303]      ### Program is starting ###
03/11/10 04:28:15 pm  localhost  logdrake[31323]      ### Program is starting ###
03/11/10 04:36:54 pm  localhost  logdrake[31303]      ### Program is exiting ###
03/11/10 04:37:10 pm  localhost  logdrake[31323]      ### Program is exiting ###
03/11/10 04:37:13 pm  localhost  drakconf.real[312... modified file /etc/mcc.conf
03/11/10 04:44:53 pm  localhost  drakconf.real[321... ### Program is starting ###
03/11/10 04:44:57 pm  localhost  drakconf.real[321... ### Program is starting ###
03/11/10 04:44:58 pm  localhost  logdrake[32123]      ### Program is starting ###
03/11/10 04:45:10 pm  localhost  logdrake[32123]      ### Program is exiting ###
03/11/10 04:49:11 pm  localhost  klogd                martian destination 0.0.1.103 from 192.168.192.168, dev ...
03/11/10 04:49:11 pm  localhost  klogd                martian destination 0.0.1.103 from 192.168.192.168, dev ...
03/11/10 05:01:01 pm  localhost  CROND[489]           (root) CMD (nice -n 19 run-parts --report /etc/cron.hourly)
03/11/10 05:07:18 pm  localhost  klogd                martian destination 0.0.1.113 from 192.168.192.168, dev ...
03/11/10 05:08:41 pm  localhost  drakconf.real[321... modified file /etc/mcc.conf

2,000 lines.   05:08:54 pm: Log file '/var/log/auth.log' has changed.   Last updated: 05:08:41 pm.
```

Just as with PCC, you will need to supply your root password when launching KSystemLog. When it opens, you are presented with the System Log preloaded, as in the second screen shot.

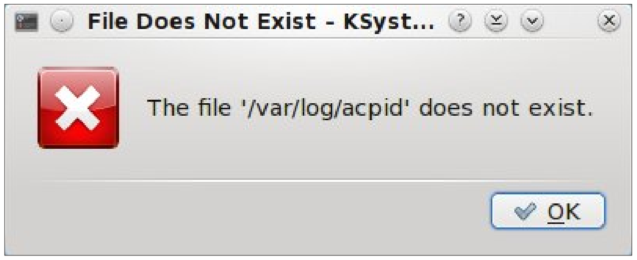
If you want to view a different log file, simply click on one of the log files listed across the top of the KSystemLog window, or select one from the "Logs" menu (there are quite a few more log files listed under the "Logs" menu than just those listed across the top of the window). By doing so, the log you select will replace the System Log in the main window.

KSystemLog also sports a multiple tab interface, so you can view multiple log files at one time. By pressing Ctrl + T, or by selecting Window » New Tab from the menus, you can open up a new tab, and select which log file you would like to view in that tab.



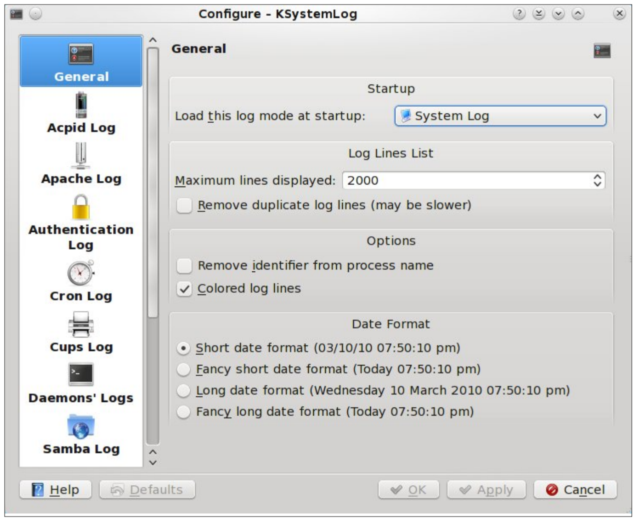
```
Information    2.4GHZ RF ONLY  MOUSE: Close
```

If you select a log file that does not exist, or cannot be found in the default location, you will receive the following error message box:



The file '/var/log/acpid' does not exist.

Fortunately, there are quite a few configuration options for KSystemLog.



General

Startup

Load this log mode at startup: System Log

Log Lines List

Maximum lines displayed: 2000

☐ Remove duplicate log lines (may be slower)

Options

☐ Remove identifier from process name

☑ Colored log lines

Date Format

● Short date format (03/10/10 07:50:10 pm)

○ Fancy short date format (Today 07:50:10 pm)

○ Long date format (Wednesday 10 March 2010 07:50:10 pm)

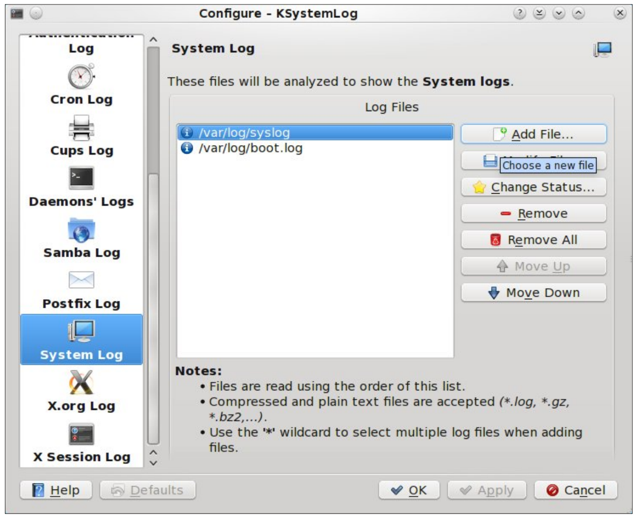
○ Fancy long date format (Today 07:50:10 pm)

Under the General section, you can specify which log file you want to be loaded by default when KSystemLog starts. The default is System Log, but you can choose to have no log file automatically loaded, or you can choose from 16 other log files. They include:

* ACPI Log
* Apache Access Log
* Apache Log
* Authentication Log
* Cron Log
* Cups Web Log1000
* Cups Log
* Cups Page Log
* Cups PDF Log
* Daemons' Log
* Kernel Log
* Postfix Log
* Samba Access Log
* Samba Log
* X.org Log
* X Session Log

Right below the selection of the log file to load when KSystemLog starts, you can specify how many lines of the log file to load. Some of the log files, due to how complete and complex they are, can become quite long. The default number of lines to load is 1,000. In the example above, I've increased the maximum number of lines to 2000. You also have an option to remove duplicate lines, although doing so may cause slower loading of the log files.

When you select the individual log files listed on the left side of the Configuration, you are given the chance to specify a different location where each log file resides, just in case log files are stored in different locations from Linux distro to distro.



System Log

These files will be analyzed to show the **System logs**.

Log Files

/var/log/syslog
/var/log/boot.log

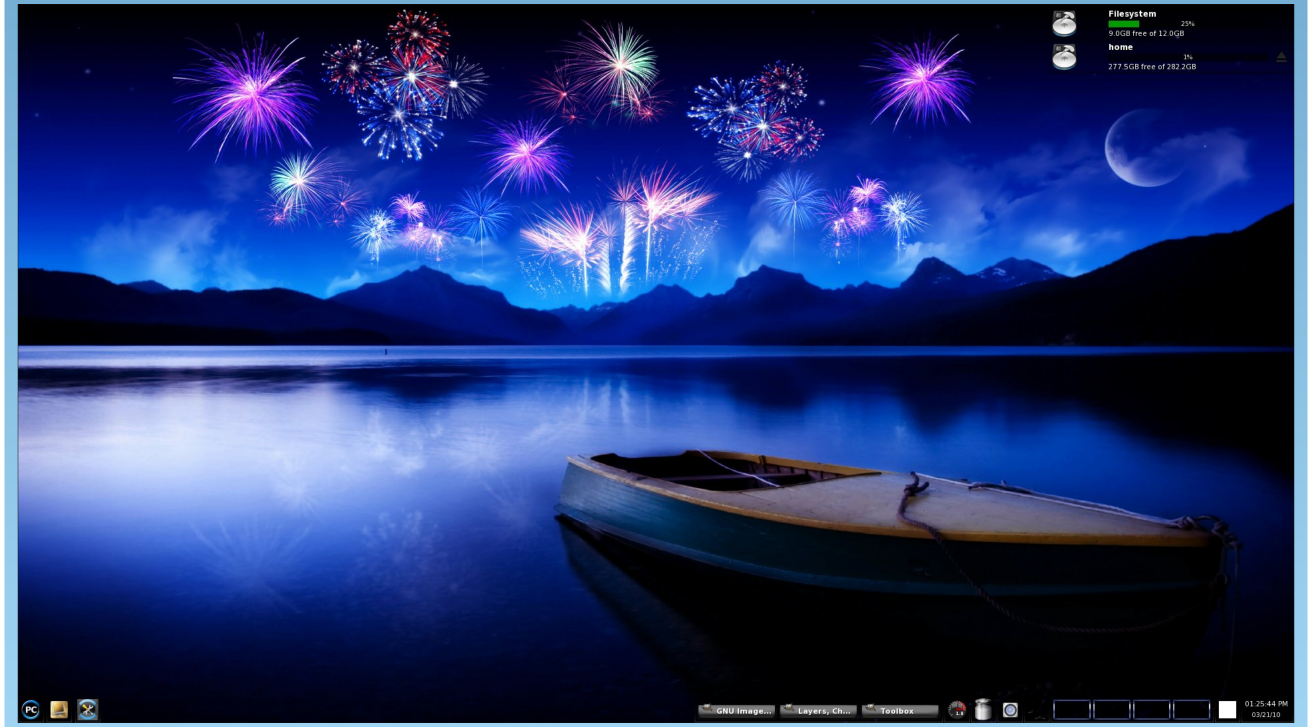
Notes:
• Files are read using the order of this list.
• Compressed and plain text files are accepted (*.log, *.gz, *.bz2,...).
• Use the '*' wildcard to select multiple log files when adding files.

Somewhat surprisingly, the Boot Log was left out of the choices you can choose to view. Never fear, as you have a couple of options. First, you can simply point one of the log file categories to load the Boot Log, instead of the one it's suppose to display. The only shortcoming to this is that the Boot Log will always be displayed with the name of the category you chose to change. For example, if you specify the Boot Log in place of the Apache Access Log, the label will still read Apache Access Log, while displaying the Boot Log. The other way to display the Boot Log is to add it to the end of the System Log. Select "Add File ..." from the configuration dialog box, and select the Boot Log from the list of logs that occupy /var/log.

With KSystemLog, you can search the log file being currently displayed (Edit » Find...). You can also save the log file (File » Save), and you can even add an entry to the currently displayed.

All in all, Linux is about choice. Every PCLinuxOS user can view and search the log files on their system through PCC. And KDE 4 users have another choice to use the KDE 4 tool, KSystemLog.

Screenshot Showcase



Posted by exploder, March 21, 2010, running e17

Setting Up a Fake RAID Array in PCLinuxOS

by Agapov Sergey (darkgfan)

Editor's Note: Many users may not be familiar with exactly what a RAID array is. According to the Webopedia Computer Dictionary it is:

*(rād) Short for **R**edundant **A**rray of **I**ndependent (or **I**nexpensive) **D**isks, a category of disk drives that employ two or more drives in combination for fault tolerance and performance. RAID disk drives are used frequently on servers but aren't generally necessary for personal computers. RAID allows you to store the same data redundantly (in multiple paces) in a balanced ay to improve overall performance.*

There are number of different RAID levels:

* ***Level 0*** *-- Striped Disk Array without Fault Tolerance: Provides data striping (spreading out blocks of each file across multiple disk drives) but no redundancy. This improves performance but does not deliver fault tolerance. If one drive fails then all data in the array is lost.*
* ***Level 1*** *-- Mirroring and Duplexing: Provides disk mirroring. Level 1 provides twice the read transaction rate of single disks and the same write transaction rate as single disks.*
* ***Level 2*** *-- Error-Correcting Coding: Not a typical implementation and rarely used, Level 2 stripes data at the bit level rather than the block level.*
* ***Level 3*** *-- Bit-Interleaved Parity: Provides byte-level striping with a dedicated parity disk. Level 3, which cannot service simultaneous multiple requests, also is rarely used.*
* ***Level 4*** *-- Dedicated Parity Drive: A commonly used implementation of RAID, Level 4 provides block-level striping (like Level 0) with a parity disk. If a data disk fails, the parity data is used to create a replacement disk. A disadvantage to Level 4 is that the parity disk can create write bottlenecks.*
* ***Level 5*** *-- Block Interleaved Distributed Parity: Provides data striping at the byte level and also stripe error correction information. This results in excellent performance and good fault tolerance. Level 5 is one of the most popular implementations of RAID.*
* ***Level 6*** *-- Independent Data Disks with Double Parity: Provides block-level striping with parity data distributed across all disks.*
* ***Level 0+1*** *-- A Mirror of Stripes: Not one of the original RAID levels, two RAID 0 stripes are created, and a RAID 1 mirror is created over them. Used for both replicating and sharing data among disks.*
* ***Level 10*** *-- A Stripe of Mirrors: Not one of the original RAID levels, multiple RAID 1 mirrors are created, and a RAID 0 stripe is created over these.*
* ***Level 7****: A trademark of Storage Computer Corporation that adds caching to Levels 3 or 4.*
* ***RAID S****: (also called Parity RAID) EMC Corporation's proprietary striped parity RAID system used in its Symmetrix storage systems.*

This article is devoted to all owners of so-called fake-raid disk arrays. But let's hold down our mighty troika and discuss the details first.

As you must now, there are two types of raid arrays, classified by logical organization - it's our infamous jbod (linear), 0, 1, 0+1, 5, 10 and so on; and their identity to hardware devices organization - consisting of software, hardware and fake-raid arrays.

Hardware cards are still expensive (but it's the best choice, indeed). Software organization is a very good choice, too. All you need are hard disk drives and free plugs. Besides, it has a simple rescue mechanism.

But what about fake-raid? It's on almost all motherboard built-in devices or cards with pricing below $30. Any advantages? In performance comparison with software-based raid arrays, these two stride side-by-side. If it's possible, I strongly recommend to use software raid (to avoid problems in the future). But if there are no free plugs and there is no true hardware raid card nearby, you may have no other choice, and our only way is to "fake" it a little.

We'll focus on building a level zero ("0") array, based on the Silicon 3112 chipset, so let's roll.

1. I suppose, you have already installed your raid controller and connected hard drives to it. And, of course, you will have needed to select the necessary mode in the controller bios, during the

boot up of your system. For me, it was "ZERO mode." If not, now is a good time to do it.

2. Don't worry about device drivers; your system will automatically load them, installing dmraid and dmsetup utilities.

In a terminal session, type:

```
# apt-get install dmraid
```

dmsetup is a required dependence, and it will preload with dmraid. Just select "Yes" if you are prompted.

3. After those tasks, two files must appear in folder **/dev/mapper**. They have such frightful names as:

```
sil_aiabcddccecj
sil_aiabcddccecj1
```

You will also need two symbolic links for them (one for each), in the **/dev** folder, with names **dm-0** and **dm-1**.

If it didn't happen, we must install our array manually:

```
# dmraid –l
```

When listing all supported controllers, remember we only need `<sil>`.

```
asr : Adaptec HostRAID ASR (0,1,10)
ddf1 : SNIA DDF1 (0,1,4,5,linear)
hpt37x : Highpoint HPT37X
(S,0,1,10,01)
hpt45x : Highpoint HPT45X (S,0,1,10)
isw : Intel Software RAID (0,1,01)
jmicron : JMicron ATARAID (S,0,1)
lsi : LSI Logic MegaRAID (0,1,10)
nvidia : NVidia RAID (S,0,1,10,5)
pdc : Promise FastTrack (S,0,1,10)
sil : Silicon Image(tm) Medley(tm)
(0,1,10)
via : VIA Software RAID (S,0,1,10)
dos : DOS partitions on SW RAIDs
```

Entering the following command in a terminal session:

```
# dmraid –r
```

shows all disks in our array, only to insure that all goes right;

```
/dev/sdc: sil, «sil_ajaddfbgejde»,
stripe, ok, 976771072 sectors, data@ 0

/dev/sdb: sil, «sil_ajaddfbgejde»,
stripe, ok, 976771072 sectors, data@ 0
```

We can activate our array with the following command, typed into a terminal session:

```
# dmraid –s
```

array activation;

```
active set

# dmraid –ay –f sil
```

where `<sil>` is our controller name, which we fetched above.

Take notice that using "`-f sil`" construction sometimes is unnecessary. It's just a precaution to prevent accidents (they happen sometimes). Often, it's enough to set "`dmraid –ay`" and all works good. Undoubtedly, both constructions will be useful.

If something goes wrong, we can easily reset our configuration and start with the beginning:

```
# dmsetup remove_all –f
```

will force all of our block devices to an off state.

4. Now, let us take it a step further, and make a filesystem on our array:

```
#mkfs.ext3 –m 0 –L raid_multimedia
/dev/dm-0
```

"**`-L raid_multimedia`**" - is only a label. Use it if you need. "**`-m 0`**" sets our volume space reserve. It defaults to 5% of the drive volume. It's wise to change it (just imagine what it will it be, if you have 1TB raid - approximately 50GB won't be available).

Make sure that **dm-0** is our phantom raid block device and **dm-1** is our main partition on it.

5. After that, only the last thing we need to do is to mount our fresh partition and enter the information in fstab:

```
# mount –t ext3 /dev/dm-1 /mnt/raid
```

Your fstab line should resemble something like this:

```
/dev/dm-1 /mnt/raid ext3
auto,user,rw,async 1 3
```

It's possible to change the last digit to 0, if you don't want the partition to be checked. And, it won't be a mistake to change disk dump from 1 to 0. Just set it up as you like.

Furthermore, you can change filesystem checking, using this:

```
# tune2fs -c 0 -i 1m /dev/dm-1
```

I recommend you do this once a month.

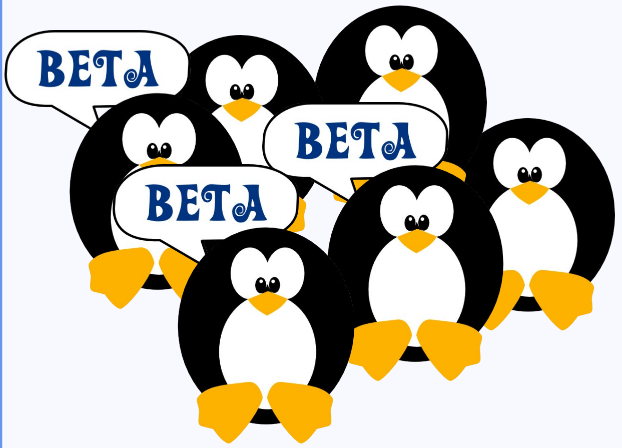
LinPC.us

Screenshot Showcase

Posted by Crow, March 27, 2010, running KDE 3.5.10

Beta, Beta, Beta: ms_meme's Beta Tribute

Way down in Linux Land lived a happy Linuxer
He loved to tinker with his OS
Lordy how he loved it
Each day he would come to the forum
And beat upon his chest
And the members said give us something new
And better than all the rest



MP3

OGG

Beta Beta Beta Beta Beta Beta Beta
From the Forum came the cry
Waita Waita Waita Waita Waita Wait
Texstar said in reply
All night long Tex worked in the shell
2010 will be the best I can tell
He was swingin' and singin'
As he told '09 farewell

Beta Beta Beta Beta Beta Beta Beta
He's done with it at last
Greata Greata Great in Linux Talk
Means now we'll have a blast
From the forum came one loud cheer
2010 is finally here
A toast to THE MAN we all revere

Command Line Interface Intro: Part 7

by Peter Kelly (critter)

Shell Scripting

A script is simply a list of instructions that we want the system to execute, and in its simplest form it will do just that, line after line, obediently and blindly, with no concern of the consequences. Writing scripts is not difficult, but care must be taken to ensure that the instructions passed in the script perform what is **intended**, which unfortunately is not always what is actually written.

There are two common types of computer programs: compiled programs and interpreted programs. Compiled programs have their code converted to a machine language that the processor can understand, but is unintelligible to humans. This enables them to execute extremely quickly, but makes them more difficult to alter. Interpreted programs are mostly plain text files that are read line by line by the interpreter, which then instructs the processor. Shell scripts are interpreted programs, and in a bash script, the bash shell is the interpreter.

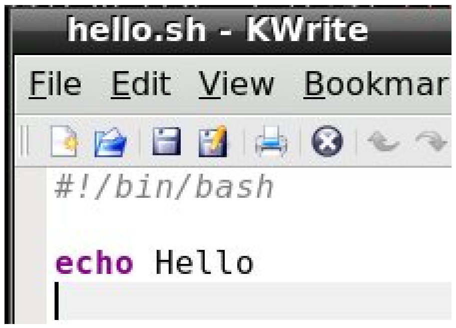
When we are at the command line, we can type in commands and have the shell perform some function for us. Sometimes, we type in the same commands regularly over a period of time, and at times, the commands get quite long and complex. Other times, we have to type in a whole series of commands to get our end result. If this sounds like you, then it is time to find out about scripting, and let the machine do the tedious work. The bash shell scripting language is a vast topic, and you will see many large volumes devoted to the subject in book stores. Fortunately, you need only a small part of all that wizardry to become a proficient script writer, and to be able to understand some of the scripts which are used to manage and control your system.

To write a script, you have to be able to type text into a file, and to then make that file executable. That can be as simple as entering text on the command line as follows:

cat > myscript
echo Hello type Ctrl+D here to end the text entry and close the file.
chmod +x myscript

Typing **./myscript** would then execute the script and print the word Hello to the screen. (The ./ is needed to tell the shell where the script is, as it is not in any of the usual places where executable files are to be found).

The method above works, but if we really want to write scripts, we should use a text editor, not a word processor, as these include strange formatting sequences that would confuse the shell. Any text editor will do. My personal favorite is kwrite, but there are many others. Ideally, you want an editor that supports **syntax highlighting**.

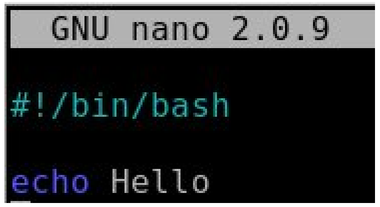


```
#!/bin/bash

echo Hello
```

Any programming language comprises many things such as comments, keywords, variables, text etc. Syntax highlighting displays these things in different colors to make it easier to find a certain item.

If you want to be able to do this using a terminal editor, then nano supports syntax highlighting, but by default it is turned off. To enable it, you need to copy a file to your home directory. In a terminal, type **cp /usr/share/nano/sh.nanorc ~/.nanorc**.

Now every file that you edit in nano that ends in **.sh** will have syntax highlighting suitable for the bash scripting language. Other files will not be affected. The **.sh** extension is not required for scripts, but is a way of telling nano that “this is a bash script, so highlight it accordingly,” and it does help to distinguish your scripts from other files. It is also a good idea to create a scripts directory in your home folder and store all your scripts in there. If you write a script that you find really useful, you can always transfer it to the /bin directory so that that is always available, as that directory is almost certainly in your PATH environment variable. Before you do that, please make sure that it is fully tested and won't one day spring a nasty surprise on you or any other user of the system.



```
#!/bin/bash

echo Hello
```

When an executable file is passed to the shell, it passes it to the kernel, which then starts a new process and attempts to execute it. If this is not a compiled, machine language file (usually referred to as a binary), then this will fail and return an error to the shell, which then searches the file for commands that it knows how to process. It may get lucky, as in the simple **myscript** example above, but as there are many scripting languages, this is not guaranteed and so we should tell the shell which interpreter to use. If the very first characters on the first line of a script are #! (known as shebang), then bash will take the rest of the line to be the fully qualified path of the interpreter. For a bash script we should always start with **#!/bin/bash**, or **#!/bin/sh** /bin/sh is usually a symbolic link to /bin/bash. For a perl script for example, we might use **#!/usr/bin/perl**.

What needs to be in a script? Well just the one line #!/bin/bash is technically a script, although it wouldn't do very much. In fact, it would do absolutely nothing more than start and end a process. To get results, we need to give it some commands, which it will execute one after another, **unless we tell it to do things differently**. This is the clever bit. **We** are in control and now have so much more power than when we simply typed in commands at the terminal. With a script, we can execute commands in the order that we want to, when we want to, dependent upon certain conditions that we define or that arise from system activity, and we can do this repeatedly or until a certain condition is met. We can pass options and arguments to the script at start up, or read information from a file or from the user at the terminal.

We could write a simple script to search a directory for all bash scripts like this:

```
#!/bin/bash

grep -rs "#!/bin/bash" /usr/bin/*
```

(Don't forget to make it executable with **chmod +x**).

The quotes are needed so that the script doesn't treat everything after the '#' as a comment. The second argument /usr/bin/* has no quotes, as we do want the shell to expand the * into a list of files.

We could do this at the command line without a script, or even define an alias that we might call find-scripts: **alias find-scripts='grep -rs "#!/bin/bash" /usr/bin/*'**

Both of these would work, but would also find the pattern anywhere in a file or text embedded in a binary file, not only at the beginning, which denotes a bash script, but they suffice as examples.

About variables

To make the effort worthwhile, we can enhance our script by passing it the name and path of the scripting language on the command line, making its use similar to a regular Linux command.

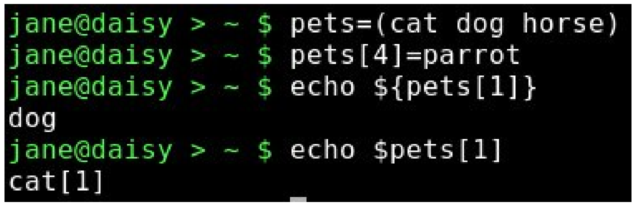
find-scripts {search pattern}

To do this we need to use **variables**. We've met variables before. They are the names of bits of information that we or the shell need to keep track of, such as **PWD**, which holds the absolute path name of our current working directory, and **PATH**, which is a colon separated list of directories to search for executable files. These are **Environment Variables** used by the shell, but generally available to the user. You can also create your own variables. They are known as variables because if, for example you did a **cd** to another directory then the contents of **PWD** would change: Their contents are variable.

Many programming languages require that variables are declared before they are used, and that the type of content that they will be assigned is defined in that declaration. The type of content may be a string (of characters), an integer, floating point number or any one of many other types. Variable declaration is available in bash using the keyword **declare**, but for most purposes, it is not necessary, and the variables you create can be used to store strings or integers, as you require. Bash doesn't handle floating point arithmetic and needs to use utility commands, such as bc, when that functionality is required.

Bash also supports one dimensional **arrays** – one dimensional means that you can't (easily) have arrays of arrays!. An array is really just a group of variable elements with the same name and an index starting at zero for the first element. Arrays in bash are extremely flexible. For example, if we create an array named **pets** with the command **pets=(cat dog horse)**, then **pet[0]** refers to the value **cat** and **pet[2]** the value **horse**. If we now add **pets[4]=parrot** then that element gets added to the array even though **pets[3]** hasn't been assigned.

To access the contents of a particular element of an array we need to use brackets to enclose the index and braces to prevent expansion by the shell:

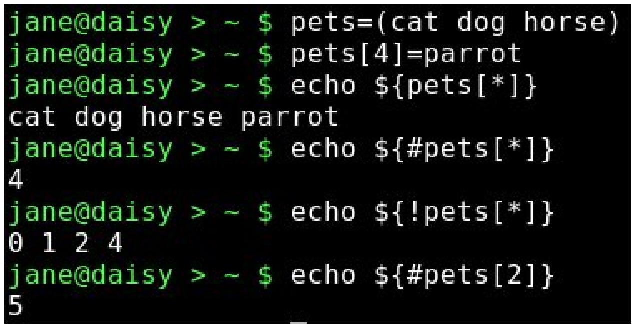


```
jane@daisy > ~ $ pets=(cat dog horse)
jane@daisy > ~ $ pets[4]=parrot
jane@daisy > ~ $ echo ${pets[1]}
dog
jane@daisy > ~ $ echo $pets[1]
cat[1]
```

echo ${pets[1]} correctly prints out **dog**, but **echo $pets[1]** prints **cat[1]** as the shell expands the variable **pets**, with no index number, to the first (zero) element of the array, and then echo treats the string **[1]** literally and adds it to the text on screen.

Quite often arrays in scripts are iterated through in a loop and their values passed directly to the rest of the code for processing which is a great way of getting things like lists of names or music into a script..

There are, of course, some special ways of accessing certain information about an array.



```
jane@daisy > ~ $ pets=(cat dog horse)
jane@daisy > ~ $ pets[4]=parrot
jane@daisy > ~ $ echo ${pets[*]}
cat dog horse parrot
jane@daisy > ~ $ echo ${#pets[*]}
4
jane@daisy > ~ $ echo ${!pets[*]}
0 1 2 4
jane@daisy > ~ $ echo ${#pets[2]}
5
```

echo ${pets[*]}
Outputs all the elements of the array

echo ${#pets[*]}
Outputs the number of elements in the array

echo ${#pets[2]}
Outputs the length of element **[2]** in the array

echo ${!pets[*]}
Outputs all the indexes of elements present in the array. Notice that the unassigned index 3 is not present.

Special bash variables

The shell has many variables at its disposal and uses some special ones to access arguments passed to a script. The first argument is known as **$1**, the second as **$2**, and so on. In the unlikely event that you need to pass more than 9 arguments to a script, then the number must be enclosed in braces, as **${14}** for the 14th argument. **$0** contains the name of the command as used on the command line.

Modifying the script like this

```
#!/bin/bash
grep -rs "#!$1" /usr/bin/*
```

allows us to call the script and pass it the absolute address of the interpreter.

./find-scripts /bin/bash to locate our bash scripts or **./find-scripts /usr/bin/perl** to find any perl scripts. We use the fact that the shell stores the first argument passed to it in the variable **$1**.

Notice that here I have changed the single quotes to double quotes, which allow variable expansion ($1 is replaced by the first argument) to take place, but still treats the **#!** literally. This is where syntax highlighting is invaluable. In the first example, the **"#!/bin/bash"** in the command expression is displayed all in red text, which is the color used for strings. In the second example, **#!** is in red text, while **$1** is in green text, the color used to highlight variables. If I had used single quotes here, then the **$1** would not have been expanded, leaving grep searching for files containing the pattern of characters **#!$1**.

We can further refine the script by passing it the search directory as a second argument, which will be stored in **$2**. We now call the script like this **./find-scripts /bin/bash /usr/bin**, passing two arguments to the script and making it much more flexible.

```
#!/bin/bash
grep -rs "#!$1" $2
```

During execution, **$1** will be expanded to **/bin/bash**, and **$2** expands to **/usr/bin.**

These enhancements unfortunately add a complication to the script, as we are now **required** to pass these arguments to the script. If we fail to pass the correct number of arguments to the script, then the variables **$2** and/or **$1** will be **undefined**. That means their value is not controlled by us and will contain a null value. As we are only reading files from a directory here, then we shouldn't cause any damage. But if the script was writing or deleting stuff, then the consequences can be imagined. You can get into a whole lot of trouble with an empty gun! This simple example should be enough to convince you!

DON'T TRY THIS!

rm -rf /$RETURNED_PATH

rm remove

-r recursing through the sub-directories

-f without asking or prompting

/$RETURNED_PATH: if this variable is undefined, then it expands to / and the command completes as “remove everything below the root directory recursively, without prompting” and deletes everything in and below the root directory – your entire system is gone, permanently and without so much as a “thank you”.

Conditional Programming

Another special variable is **$#**, which contains the number of arguments passed to the script. We can use this to check that all is OK before proceeding.

```
1 #!/bin/bash
2
3 if [ $# != 2 ]
4 then
5         echo Usage: Find-scripts pattern directory >2
6         exit1
7 fi
8
9 grep -rs "#!$1" $2
```

I've put line numbers in to help clarify things. They have nothing to do with the script are not typed in.

Lines 3 to 7 contain a construct known as an **if-then statement**. After the initial **if** keyword on line 3, we test a **condition** for truth. Here the test **[$# != 2]** checks if the total number of arguments passed is not equal to 2. The spaces inside the brackets are very important.

If it is true (that there are not 2 arguments) we execute lines 4,5 and 6. Line 4 is the entry point to commands that are only executed when the test condition is true. Line 5 echoes a usage message to the terminal. Line 6 exits the script, as we don't have sufficient information to continue, and returns a value of 1. Line 7 ends the conditional statement and allows the script to continue.

In this instance we don't use the return value of 1, which by convention signifies a failure – 0 means success. Other numbers up to and including 125 are available for the programmers use. If this script was called from another, then that parent script would know the outcome from this value and could then act accordingly.

If you use the script in a couple of months time, or even a few years down the line, you might not remember what pattern and directory the script is complaining about. It is even less likely that another user would know. One thing we can and should do is to add comments to the script, detailing what is going on. A comment is any text on its own line, or at the end of a line, that starts with a # (with exception of the initial **#!** which has special meaning to the shell). This text is ignored by the script.

```
#!/bin/bash
# search a directory to find scripts
# Needs a pattern to find e.g. /bin/bash
# and a directory path to search
#
# Jane Doe February 2010

if [ $# != 2 ] # We need 2 arguments
then           # Print out a message
        echo Usage: Find-scripts pattern directory >2
        exit1  # Quit the script
fi
# The correct number of arguments
# have been supplied so continue
grep -rs "#!$1" $2 # find script files
```

There are more comments in this file than you may usually find, but an informative header can save a lot of head scratching. Indentation can also help to make a script more readable.

The test used in the example above, **$# != 2**, is derived from the negation symbol **!**. And with the equality symbol **=** together, they give a 'not equal test.' But what if we want to test if a file was a directory or if the file even exists? Well, the shell has its very own test command with the following basic syntax: **test {expression1} {condition} {expression2}**.

Using this command the test in the if statement would have been written like this: **if test $# -ne 2**. As a matter of the fact, the two forms are completely interchangeable, and the conditions available for the test command can be used equally well by the original format **[$# -ne 2]**. The shell has many functions like test built in to it. They are known, unsurprisingly, as **shell builtins**. The keyword **test** is a builtin, as is **[**, which has the same meaning.

The use of tests is so central to shell scripting to determine the flow of the program that you should be aware of the tests available. I give here a complete list of the tests available as described in the official man page documentation.

Where EXP is an expression

`( EXP )`	`EXP is true`
`! EXP`	`EXP is false`
`EXP1 -a EXP2`	`both EXP1 and EXP2 are true (logical and)`
`EXP1 -o EXP2`	`either EXP1 or EXP2 is true (logical or)`

where STR is a string

`-n STR`	the length of STR is nonzero
`STR`	equivalent to -n STR
`-z STR`	the length of STR is zero
`STR1 = STR2`	the strings are equal
`STR1 != STR2`	the strings are not equal

Where INT is an integer

`INT1 -eq INT2`	INT1 is equal to INT2
`INT1 -ge INT2`	INT1 is greater than or equal to INT2
`INT1 -gt INT2`	INT1 is greater than INT2
`INT1 -le INT2`	INT1 is less than or equal to INT2
`INT1 -lt INT2`	INT1 is less than INT2
`INT1 -ne INT2`	INT1 is not equal to INT2

Where F is a file

`F1 -ef F2`	F1 and F2 have the same device and inode numbers
`F1 -nt F2`	F1 is newer (modification date) than F2
`F1 -ot F2`	F1 is older than F2
`-b F`	F exists and is block special
`-c F`	F exists and is character special
`-d F`	F exists and is a directory
`-e F`	F exists
`-f F`	F exists and is a regular file
`-g F`	F exists and is set-group-ID
`-G F`	F exists and is owned by the effective group ID
`-h F`	F exists and is a symbolic link (same as -L)
`-k F`	F exists and has its sticky bit set
`-L F`	F exists and is a symbolic link (same as -h)
`-O F`	F exists and is owned by the effective user ID
`-p F`	F exists and is a named pipe
`-r F`	F exists and read permission is granted
`-s F`	F exists and has a size greater than zero
`-S F`	F exists and is a socket
`-t FD`	file descriptor FD is opened on a terminal
`-u F`	F exists and its set-user-ID bit is set
`-w F`	F exists and write permission is granted
`-x F`	F exists and execute (or search) permission is granted

That list should give you some idea of the flexibility you have when performing a test.

Note! The **-e** test for the existence of a file can also be written **-a**, but I choose to ignore this, as it is too easy to confuse with the **-a** (logical and) test. You may, however, see it used in other scripts.

The **if-then** statement may also contain the **else** keyword, which works like this:

if {condition}

then
    commands to execute if the condition is met
else
    commands to execute if the condition is not met
fi

In the next example, I use the command **read**, which is an easy way to get user input into a variable as a script is running.

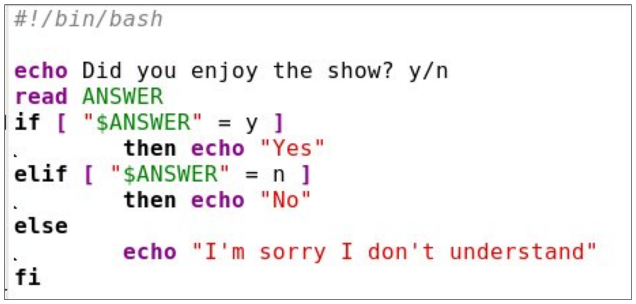
```
#!/bin/bash

echo Did you enjoy the show? y/n
read ANSWER
if [ $ANSWER = y ]
        then echo "Yes"
else
        echo "No"
fi
```

After the first echo command, the script pauses until the user enters something at the keyboard and presses the return key. The users input is stored in the variable **ANSWER**. This time the script does something different, depending on the users input.

But what if the user types in something other than Y or N? To cope with this, we introduce another keyword – **elif**.

```
#!/bin/bash

echo Did you enjoy the show? y/n
read ANSWER
if [ $ANSWER = y ]
        then echo "Yes"
elif [ $ANSWER = n ]
        then echo "No"
else
        echo "I'm sorry I don't understand"
fi
```

In this script, the acceptable responses are caught and acted upon. Any other response is dealt with by the code after **else**. This would appear to solve the problem, but if the return key is pressed without the user entering a response, then nothing is assigned to the variable **ANSWER**, which defaults to a null value, and the script would see the tests as **[= y]** and **[= n]**, which produces the error message **unary operator expected**. The way around this is to use double quotes around the variable, which causes the test to be seen as **["" = y]** or **["" = n]**, which are valid expressions that the shell can work with. The "" in the test is an empty string (a string with no characters), which is not the same as a null.

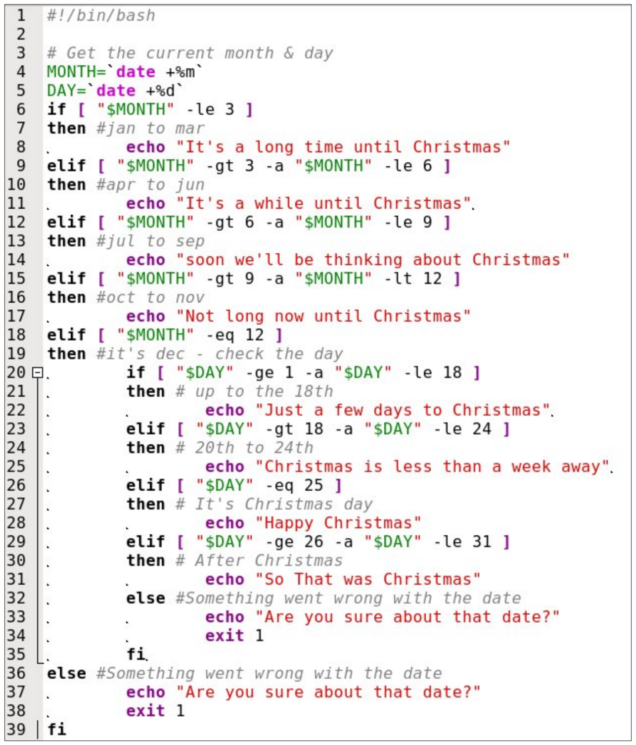


```
#!/bin/bash

echo Did you enjoy the show? y/n
read ANSWER
if [ "$ANSWER" = y ]
        then echo "Yes"
elif [ "$ANSWER" = n ]
        then echo "No"
else
        echo "I'm sorry I don't understand"
fi
```

You can have as many **elif** tests as you wish, and the if statement can be nested as many times as you can keep track of.

```
If [condition]
then
        if [condition]
        then
                if [condition]
                …
                …
                …
                fi
        fi
fi
```

And of course each **if** statement can have its own **elifs** and **elses**. Here's a longer one with those line numbers again.



```
1  #!/bin/bash
2
3  # Get the current month & day
4  MONTH=`date +%m`
5  DAY=`date +%d`
6  if [ "$MONTH" -le 3 ]
7  then #jan to mar
8          echo "It's a long time until Christmas"
9  elif [ "$MONTH" -gt 3 -a "$MONTH" -le 6 ]
10 then #apr to jun
11         echo "It's a while until Christmas"
12 elif [ "$MONTH" -gt 6 -a "$MONTH" -le 9 ]
13 then #jul to sep
14         echo "soon we'll be thinking about Christmas"
15 elif [ "$MONTH" -gt 9 -a "$MONTH" -lt 12 ]
16 then #oct to nov
17         echo "Not long now until Christmas"
18 elif [ "$MONTH" -eq 12 ]
19 then #it's dec - check the day
20         if [ "$DAY" -ge 1 -a "$DAY" -le 18 ]
21         then # up to the 18th
22                 echo "Just a few days to Christmas"
23         elif [ "$DAY" -gt 18 -a "$DAY" -le 24 ]
24         then # 20th to 24th
25                 echo "Christmas is less than a week away"
26         elif [ "$DAY" -eq 25 ]
27         then # It's Christmas day
28                 echo "Happy Christmas"
29         elif [ "$DAY" -ge 26 -a "$DAY" -le 31 ]
30         then # After Christmas
31                 echo "So That was Christmas"
32         else #Something went wrong with the date
33                 echo "Are you sure about that date?"
34                 exit 1
35         fi
36 else #Something went wrong with the date
37         echo "Are you sure about that date?"
38         exit 1
39 fi
```

Line 1 is our standard bash script header. Line 3 is a comment and ignored.

Lines 4 & 5 use the date function with format modifiers (**%m** and **%d**) to get the current month and date into our variables

Line 6 Starts the first of 2 if-then constructs checking if the month is **less than or equal to 3**.

Line 9 tests if the month is **greater than 3 <u>and</u> less than or equal to 6**. That is, it is either 4, 5 or 6.

Line 19 We've discovered that it is December so we start the second if-then construct to check the day.

Lines 23, 26 and 29 do more day testing.

Line 32 the default else statement. If we got here, the the day was not in the range 1 – 31, so something is wrong and we leave the script.

Line 36 We find ourselves back in the first if-then construct at the else statement. If we got here, the the month was not in the range 1 -12, so something is wrong and we leave the script.

Line 39 Terminates the first if-then construct.

While the above script is useful to demonstrate the use of nesting if – then statements and the use of multiple **elifs**, it is not the only way or the most efficient way to program this.

We could have used the **case** statement, which is another conditional construct. This is the syntax for the **case** statement.

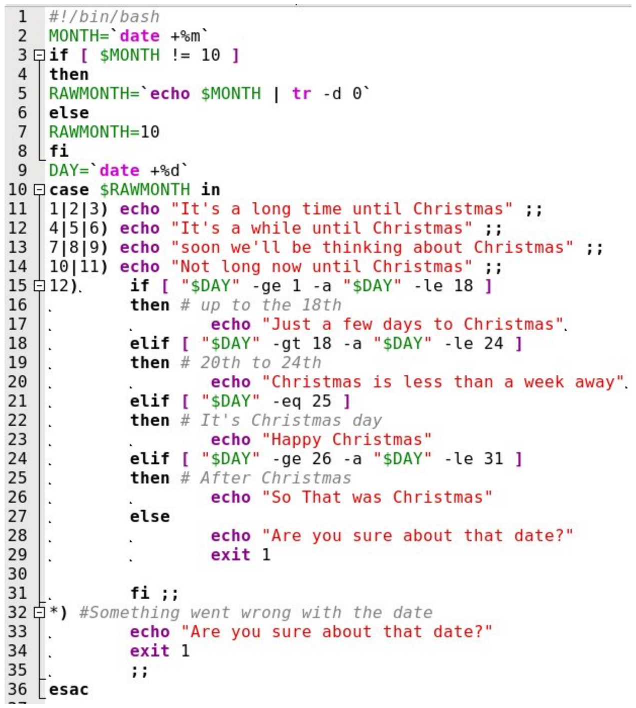
```
Case {pattern} in
        value1)
                commands
                ;;
        value2)
                commands
                ;;
        …
```

```
    ...
    ...
    *)
            commands
            ;;
esac
```

In this structure, the pattern is something, like the contents of a variable, that you want to use to control the actions of the script. If it has value1, then those commands up to the **;;** are executed. Value2 causes a different set of commands to be executed and so forth, until all values that you wish to test for have been checked. The default at the end of the statement catches any other value, and is used as a fail-safe for unwanted or unexpected values. It can also provide a way to exit the script (or code segment). To test for multiple values, separate them with a pipe symbol **|**.

In the next example, I have mixed a case statement and the nested if-then statement from the previous example, and added line numbers to the figure.

Because the values to be tested in line numbers 11 t o 15 are integer numbers and the date function returns a two character string such as “02,” the tests in lines 8 to 12 would fail because of the leading “0”. To overcome this, we echo the value through a pipe to the **tr** (translate) command and use the option **-d** (delete) with the argument **“0,”** which deletes any zeroes in the string. unless the string is “10,” which is an integer. This expression is evaluated in the back ticks and assigned to the new variable **RAWMONTH**.



```
1  #!/bin/bash
2  MONTH=`date +%m`
3  if [ $MONTH != 10 ]
4  then
5  RAWMONTH=`echo $MONTH | tr -d 0`
6  else
7  RAWMONTH=10
8  fi
9  DAY=`date +%d`
10 case $RAWMONTH in
11 1|2|3) echo "It's a long time until Christmas" ;;
12 4|5|6) echo "It's a while until Christmas" ;;
13 7|8|9) echo "soon we'll be thinking about Christmas" ;;
14 10|11) echo "Not long now until Christmas" ;;
15 12)        if [ "$DAY" -ge 1 -a "$DAY" -le 18 ]
16            then # up to the 18th
17                    echo "Just a few days to Christmas"
18            elif [ "$DAY" -gt 18 -a "$DAY" -le 24 ]
19            then # 20th to 24th
20                    echo "Christmas is less than a week away"
21            elif [ "$DAY" -eq 25 ]
22            then # It's Christmas day
23                    echo "Happy Christmas"
24            elif [ "$DAY" -ge 26 -a "$DAY" -le 31 ]
25            then # After Christmas
26                    echo "So That was Christmas"
27            else
28                    echo "Are you sure about that date?"
29                    exit 1
30
31            fi ;;
32 *) #Something went wrong with the date
33            echo "Are you sure about that date?"
34            exit 1
35            ;;
36 esac
```

We could have used the two character string as returned from the data function in the case statements, but using integers demonstrates the need to be aware of the **type** of data we use in tests.

Each test in the case statement is on one line here to make it more compact. If there are multiple commands for a test, then they should be separated by a semicolon or by a newline character (which means on separate lines). I think that you'll agree that the **case** statement is easier to read than the many **elifs** in the **if** statement.

The if-then and case structures are examples of conditional programming where the progress and direction of the script is determined by the results of certain tests. The shell has two conditional operators, **&&** and **||**, known as **“logical and”** and **“logical or”**. They work both in unary (one argument) and binary (two arguments) mode.

In unary mode:

["$A" -gt 4] && echo "yes"

If the expression **["$A" -gt 4]** evaluates to true the the echo command is executed, if false the script ignores the interruption and continues.

The **||** operator has the opposite effect in that the expression has to evaluate to false for the command to be executed.

Binary mode is used to test two arguments:

if ["$A" -lt 4] && ["$B" -gt 9] echo “yes”

The echo command is executed if and only if both expressions are true.

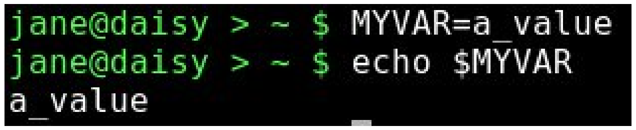
if ["$A" -lt 4] || ["$B" -gt 9] echo “yes” The echo command is executed if either or both expressions are true.

This is similar, but not the same, as the test operators **-a** and **-o**. When using the test operators, both expressions are evaluated, and then the test is performed. The **&&** shell operator evaluates the first expression, and if it is false, then there is no point in looking at the second expression, as the **'and'** condition cannot be met.

In a similar manner. if the first expression in an 'or' test using the || operator evaluates to true, then the condition has already been met and the second expression doesn't need to be evaluated. For this reason, they are known as **short circuit operators**.

The scope of variables

As we have now started to use our own variables, it is important that you understand the **scope** of variables before we move on. The scope of a variable is where its assigned value is valid. Variables may be **local** or **global**. For example, while on the command line, you are in a running shell and you may create variables

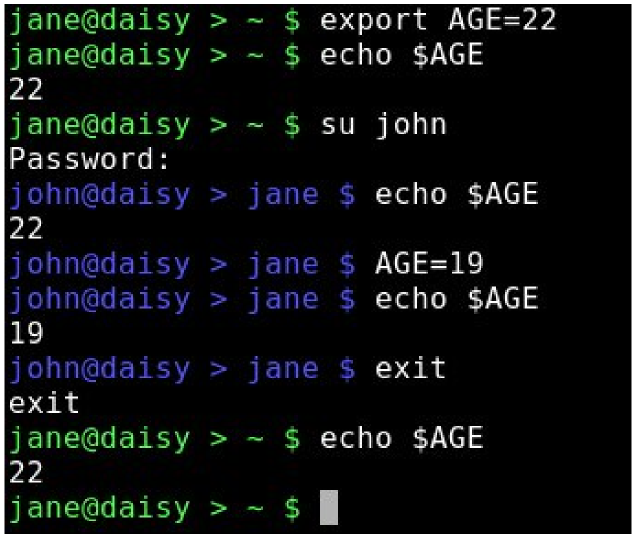


```
jane@daisy > ~ $ MYVAR=a_value
jane@daisy > ~ $ echo $MYVAR
a_value
```

The scope of that variable is the currently running shell process. When you exit the shell, the variable ceases to exist, and if you start a new shell the variable isn't available, as it is **local** to the shell process where it was created. When you run a script a new shell process is started and any variables that you create are **local** to that script and not available elsewhere.

Environment variables are **global** variables and are available to all processes. In order to make your variables available to other processes, they need to be **exported** to the **environment**. All new processes inherit the environment of their parent process.

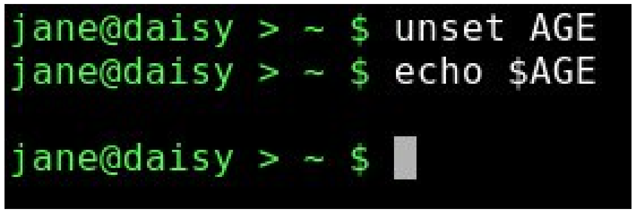
When an exported variable is passed to a child process, it retains the value assigned it in the parent process. The child process may change the value of the variable, but the value seen by the parent remains unchanged.



```
jane@daisy > ~ $ export AGE=22
jane@daisy > ~ $ echo $AGE
22
jane@daisy > ~ $ su john
Password:
john@daisy > jane $ echo $AGE
22
john@daisy > jane $ AGE=19
john@daisy > jane $ echo $AGE
19
john@daisy > jane $ exit
exit
jane@daisy > ~ $ echo $AGE
22
jane@daisy > ~ $
```

Jane set the variable **AGE** to 22, her age, and exported it. When the **su** command was executed to switch to user john, a new shell process was started which could access the variable and its value, as set by jane, which john subsequently changed to 19, his age. Jane still sees the variable set as 22.

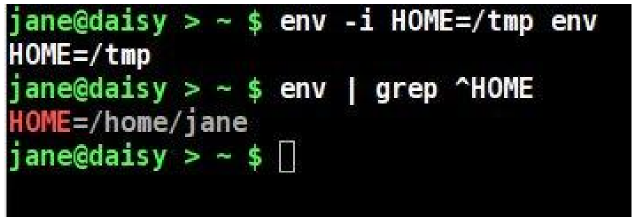
To remove a variable, use the command **unset**.



```
jane@daisy > ~ $ unset AGE
jane@daisy > ~ $ echo $AGE

jane@daisy > ~ $
```

Another command used with variables is **readonly**, which has the effect of turning a variable into a **constant** – a variable whose value, once set, cannot vary. For example, **readonly KB=1024**. The assigned value cannot be changed during the life of the process and readonly variables cannot be **unset**.

The **env** command is used to display the environment variables that are currently set and to control the environment that gets passed to commands. The **env** command on its own will display a list of all current environment variables. If the **env** command is followed by one or more variable assignments and used before a command, the environment passed to the command will be modified, but the current working environment will be unaffected. With the **-i** option the current environment will be ignored and only variables passed to the command will be used.



```
jane@daisy > ~ $ env -i HOME=/tmp env
HOME=/tmp
jane@daisy > ~ $ env | grep ^HOME
HOME=/home/jane
jane@daisy > ~ $
```

The environment variable **HOME**, which normally contains the full path name of the users home directory, is temporarily changed to /tmp, and all other environment variables discarded. This new environment is then passed to the command **env**, which starts in a new process and lists out all its known environment variables. There is only one, as the others were discarded.

The **env** command is then immediately executed in the current shell, and the output searched for lines that begin with the pattern “HOME”. The changed environment existed only for the process to which it was passed.

Please Donate To PCLinuxOS.

PCLXDE

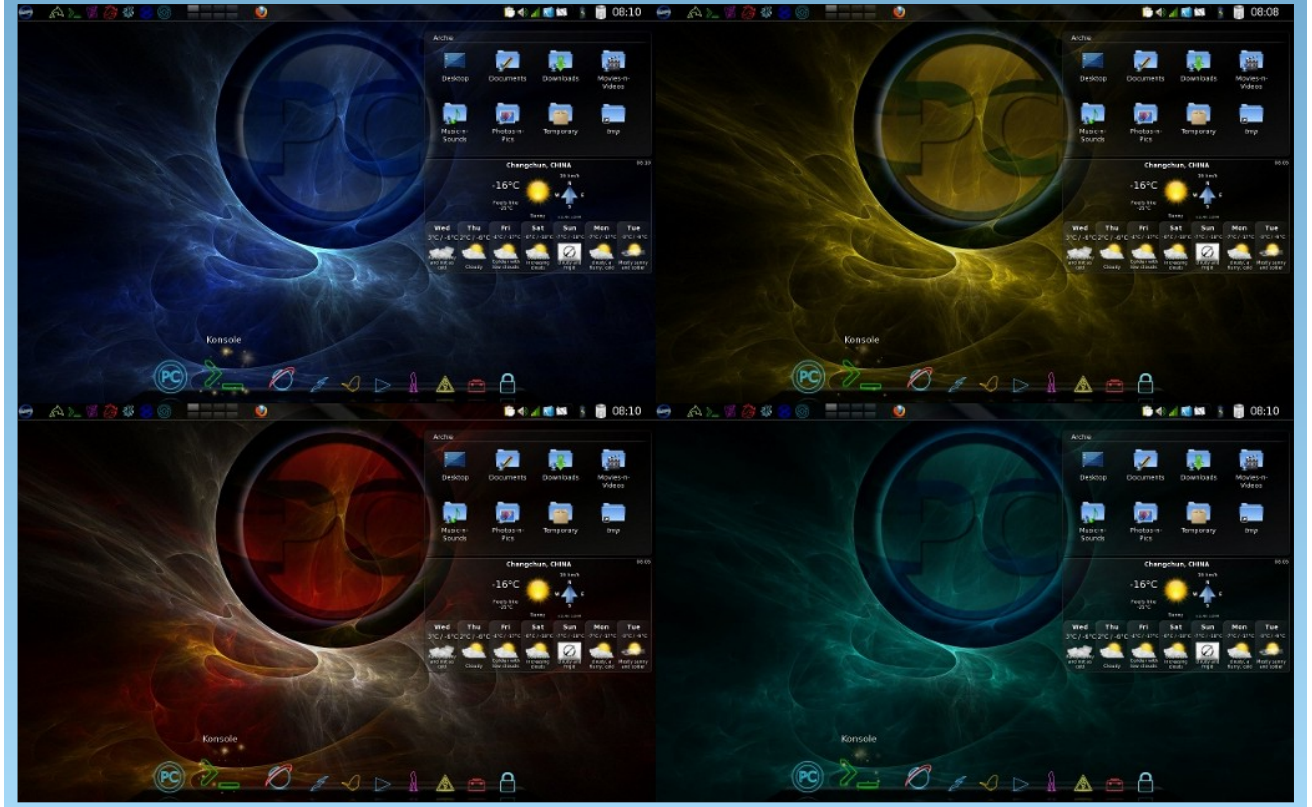
MyPCLinuxOS

Your Community Projects Forum

Answers to Mark Szorady's Double Take: *(1) “Gotta” changed to “have to”; (2) Necklace missing; (3) Doorknob higher; (4) Cabinet larger; (5) Arm lower; (6) Plug added; (7) Laptop screen smaller.*

This month's Double Take features Old Polack, a PCLinuxOS Forum Global Moderator. To learn more about Old Polack, check out the Behind The Scenes column in the ***August, 2009*** *issue of the PCLinuxOS Magazine.*

Screenshot Showcase



Posted by Archie, March 2, 2010, running KDE 4

Printing to a Vista Machine from PCLinuxOS

by Robert Stahl (Hairyplotter)

I ran in to a very interesting problem when I was trying to get my laptop running PCLinuxOS 2009.2 to print to my wife's machine running Windows Vista. After configuring both computers, I sent a test page to her printer. Her printer queue displayed the job and claimed it was printing, yet the job stopped showing only 64kb had been sent to the printer. After double checking both machines for settings and the proper drivers I went to Google to see if anyone else had this same issue. I found several pages dating back 5 years where people asked for help on this very same problem, yet the replies were of little to no help. I did manage to stumble across a couple pages that did more than suggest updating drivers, and from these pages I discovered the solution to my problem.

The following is a walk through on setting up PCLinuxOS to print to a printer connected to a Windows Vista machine.

The very first step is to make sure your printer is connected to the Vista computer, and the proper drivers are installed. The manuals and CD that came with your printer will walk you through this process.

The next step is to tell the Vista computer that it needs to share it's printer with others on the network, for this step, you will need to open the start menu and select "Control Panel".

Once the control panel is open you need to select the "Network and Sharing Center" highlighted in illustration 1.1

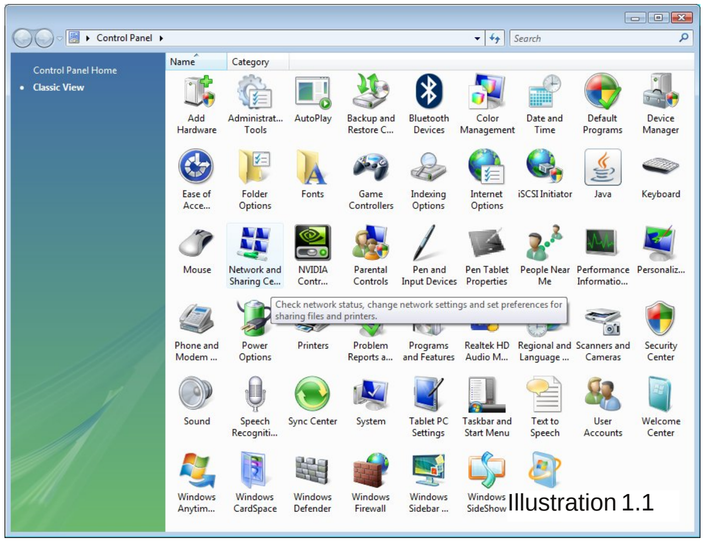
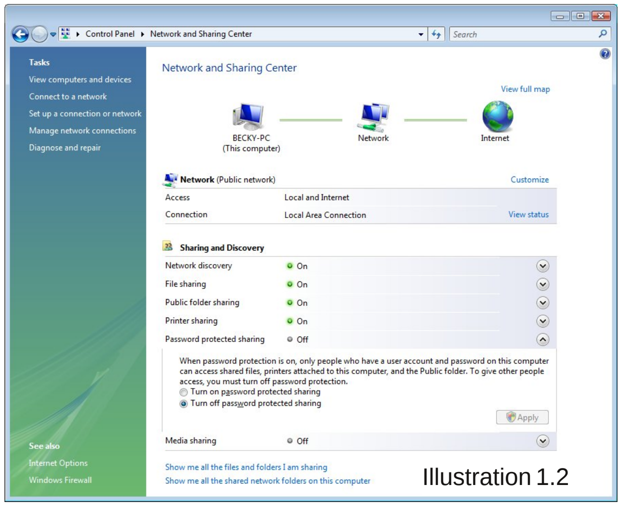


Illustration 1.1

The following dialog box will open allowing you to make all of the necessary changes to the network settings.



Network and Sharing Center

Network (Public network)

Access: Local and Internet
Connection: Local Area Connection

Sharing and Discovery

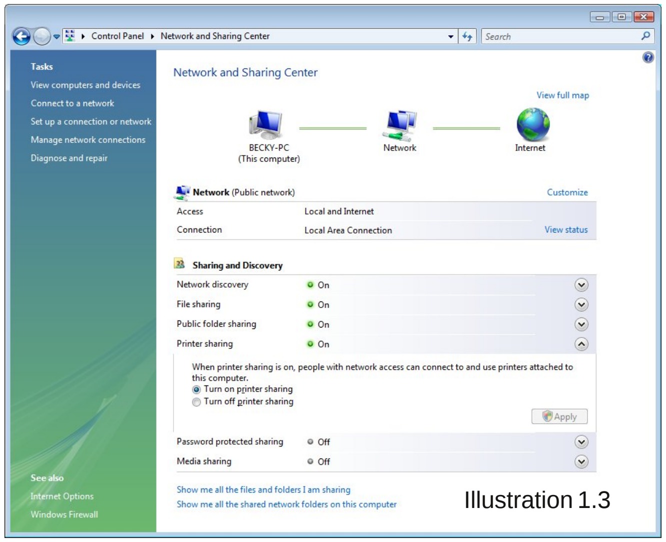
Network discovery: On
File sharing: On
Public folder sharing: On
Printer sharing: On
Password protected sharing: Off

When password protection is on, only people who have a user account and password on this computer can access shared files, printers attached to this computer, and the Public folder. To give other people access, you must turn off password protection.
Turn on password protected sharing [✓ Turn off password protected sharing]

Media sharing: Off

Show me all the files and folders I am sharing
Show me all the shared network folders on this computer

Illustration 1.2



Network and Sharing Center

Network (Public network)

Access: Local and Internet
Connection: Local Area Connection

Sharing and Discovery

Network discovery: On
File sharing: On
Public folder sharing: On
Printer sharing: On

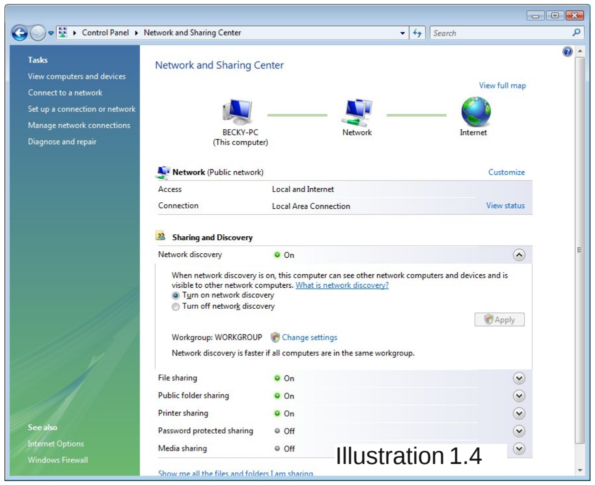
When printer sharing is on, people with network access can connect to and use printers attached to this computer.
[✓ Turn on printer sharing] Turn off printer sharing

Password protected sharing: Off
Media sharing: Off

Show me all the files and folders I am sharing
Show me all the shared network folders on this computer

Illustration 1.3

This option is more a matter of personal preference. If you leave password protected sharing on, then you will need to supply a valid login and password for the Vista computer before you will be allowed to print.



Network and Sharing Center

Network (Public network)

Access: Local and Internet
Connection: Local Area Connection

Sharing and Discovery

Network discovery: On

When network discovery is on, this computer can see other network computers and devices and is visible to other network computers. What is network discovery?
[✓ Turn on network discovery] Turn off network discovery

Workgroup: WORKGROUP

Network discovery is faster if all computers are in the same workgroup.

File sharing: On
Public folder sharing: On
Printer sharing: On
Password protected sharing: Off
Media sharing: Off

Show me all the files and folders I am sharing

Illustration 1.4

This option needs to be turned on to allow users on the network access to the printer.

Turning on network discovery

will allow printdrake to see the printer on the network making it a little easier during the Linux setup. If you feel comfortable manually specifying the workgroup, computer name and print share name then you can leave this turned off.

This is the last option we need to look at in this window. Once these 3 options are configured to your liking, you can close this window and go back to the Control Panel.

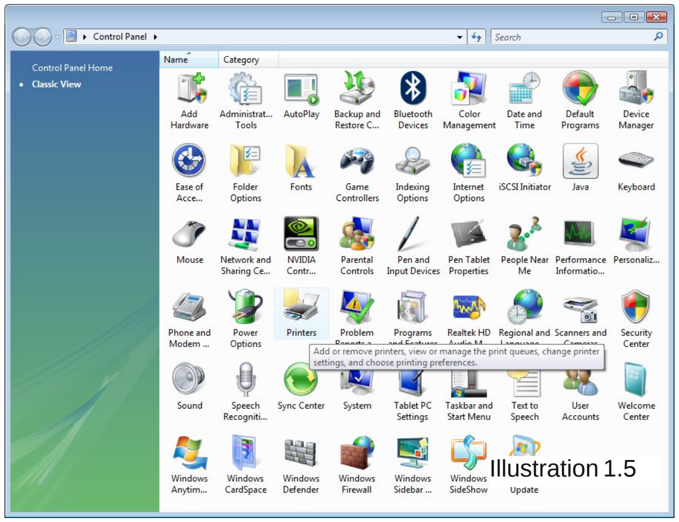


Illustration 1.5

Opening the "Printers" icon will give you a window listing all of the printers installed on the Vista computer (see Illustration 1.6). When you get to this point, the printer should already be installed on the Vista computer. All we need to do now is share it with the network and change a couple of options.

Right click on the printer you want to enable sharing for, then choose "Run as Administrator" -> "Properties". See illustration 1.7.

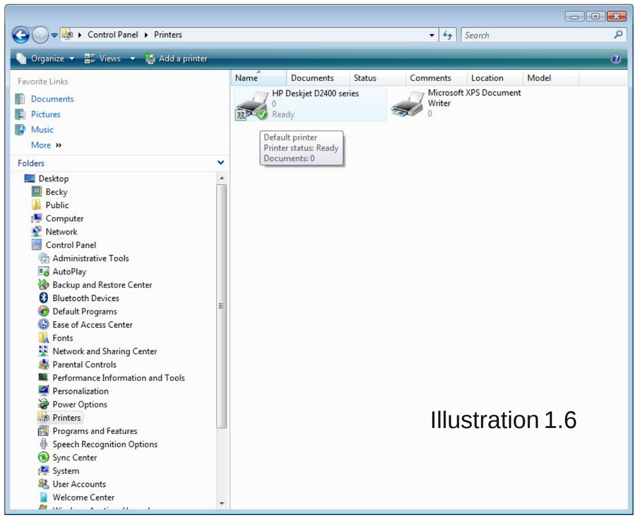


Illustration 1.6

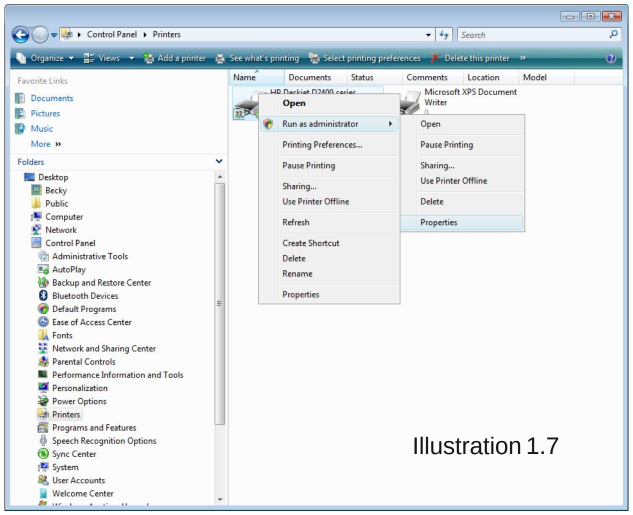
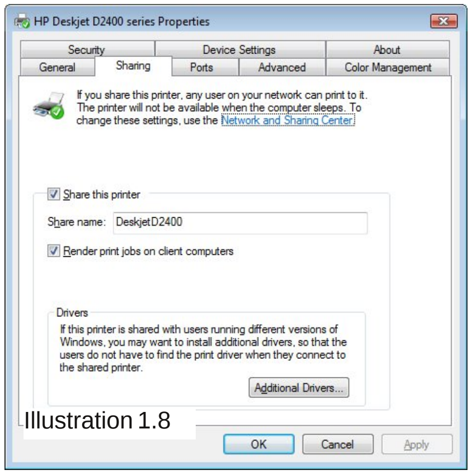


Illustration 1.7

After confirming that you want to continue as administrator, the following dialog box will open.


HP Deskjet D2400 series Properties

Security | Device Settings | About
General | Sharing | Ports | Advanced | Color Management

If you share this printer, any user on your network can print to it. The printer will not be available when the computer sleeps. To change these settings, use the Network and Sharing Center.

[✓] Share this printer

Share name: DeskjetD2400

[✓] Render print jobs on client computers

Drivers

If this printer is shared with users running different versions of Windows, you may want to install additional drivers, so that the users do not have to find the print driver when they connect to the shared printer.

Additional Drivers...

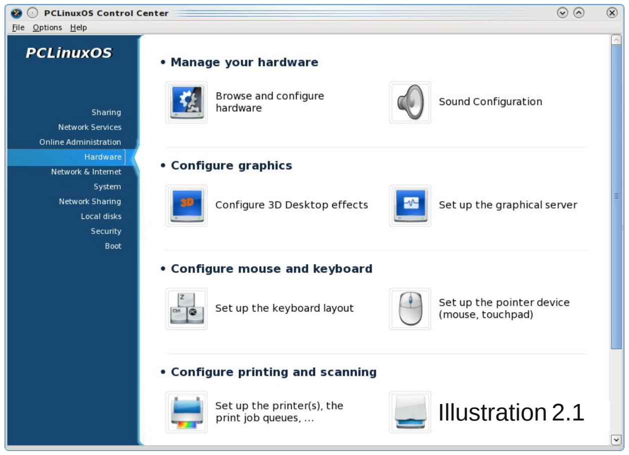
OK Cancel Apply

Illustration 1.8

Choose the "Sharing" tab and check the box for "Share this printer", then type in a name for the printer that will be used when you configure the PCLinuxOS computer.

The next step is to configure the PCLinuxOS computer to take advantage of the printer.

The first step is to open PCC or "PClinuxOS Control Center", on the left side of the window that opens, choose "Hardware" then click the option "Set up the printer(s), the print job queues, ..."


PCLinuxOS Control Center
PCLinuxOS
Sharing
Network Services
Online Administration
Hardware
Network & Internet
System
Network Sharing
Local disks
Security
Boot
Manage your hardware
Browse and configure hardware
Sound Configuration
Configure graphics
Configure 3D Desktop effects
Set up the graphical server
Configure mouse and keyboard
Set up the keyboard layout
Set up the pointer device (mouse, touchpad)
Configure printing and scanning
Set up the printer(s), the print job queues, ...
Illustration 2.1

From this window, click the "Add Printer" button on the top left.

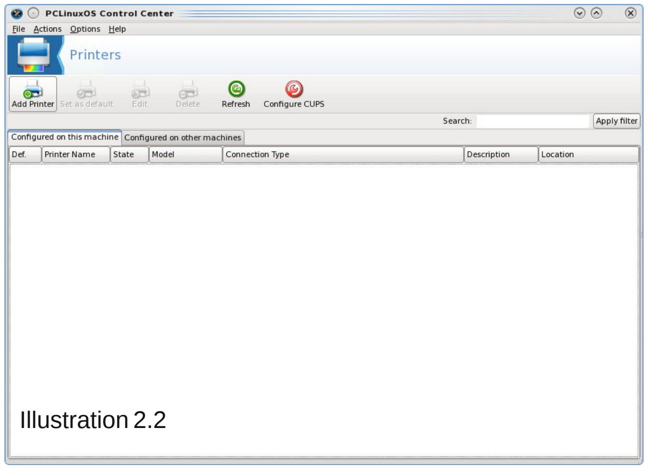
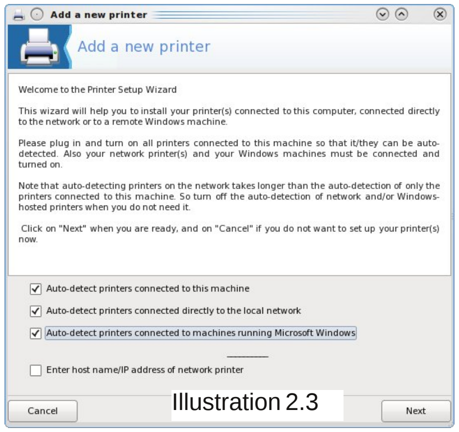


Illustration 2.2

Make sure that the 3 options are checked as in illustration 2.3 then click "Next".

Your printer *should* show up in this window. If it doesn't, try rebooting the Vista computer. Verify all of the previous settings once the Vista machine is back up and running. If you still can't see the printer, refer to NOTES below.



Welcome to the Printer Setup Wizard

This wizard will help you to install your printer(s) connected to this computer, connected directly to the network or to a remote Windows machine.

Please plug in and turn on all printers connected to this machine so that it/they can be auto-detected. Also your network printer(s) and your Windows machines must be connected and turned on.

Note that auto-detecting printers on the network takes longer than the auto-detection of only the printers connected to this machine. So turn off the auto-detection of network and/or Windows-hosted printers when you do not need it.

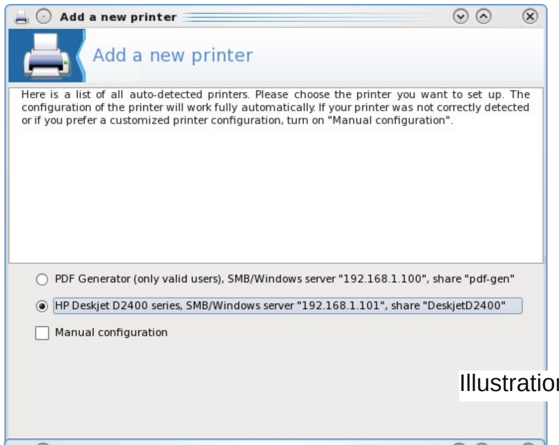
Click on "Next" when you are ready, and on "Cancel" if you do not want to set up your printer(s) now.

- [✓] Auto-detect printers connected to this machine
- [✓] Auto-detect printers connected directly to the local network
- [✓] Auto-detect printers connected to machines running Microsoft Windows
- [ ] Enter host name/IP address of network printer

Illustration 2.3

If your printer is shown, make sure its selected then click "Next"

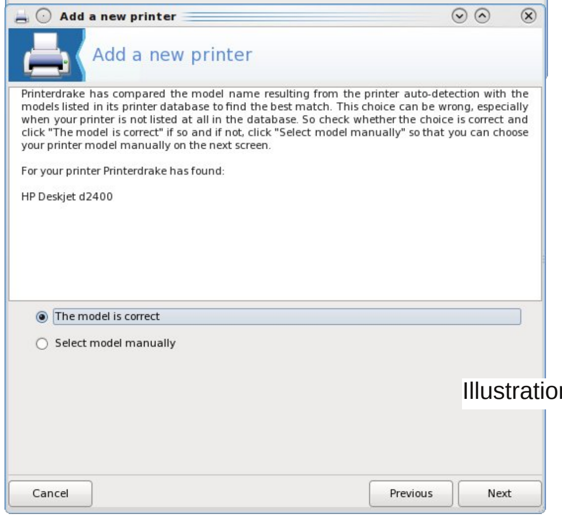
Verify that the printer is the correct model or one that is compatible, then choose "Next". If your printer isn't listed, will have to "Select model manually" and choose your printer from a large list of available printers. If you don't find your printer, then you may have to do some searching to find if one of the available drivers will work with your printer.



Here is a list of all auto-detected printers. Please choose the printer you want to set up. The configuration of the printer will work fully automatically. If your printer was not correctly detected or if you prefer a customized printer configuration, turn on "Manual configuration".

- ○ PDF Generator (only valid users), SMB/Windows server "192.168.1.100", share "pdf-gen"
- ● HP Deskjet D2400 series, SMB/Windows server "192.168.1.101", share "DeskjetD2400"
- [ ] Manual configuration

Illustration 2.4



Printerdrake has compared the model name resulting from the printer auto-detection with the models listed in its printer database to find the best match. This choice can be wrong, especially when your printer is not listed at all in the database. So check whether the choice is correct and click "The model is correct" if so and if not, click "Select model manually" so that you can choose your printer model manually on the next screen.

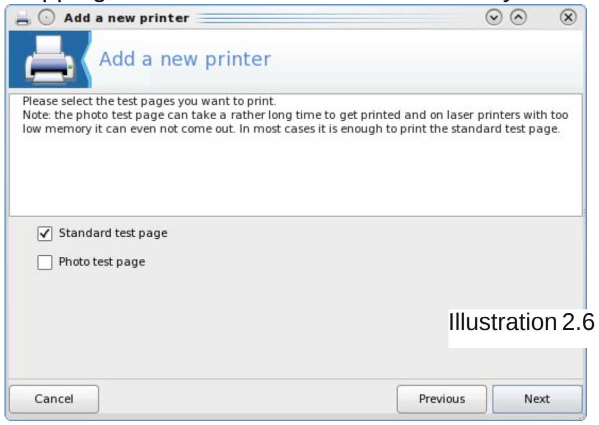
For your printer Printerdrake has found:

HP Deskjet d2400

- ● The model is correct
- ○ Select model manually

Illustration 2.5

Congratulations!
Your printer is installed and configured. Now all you need to seal the deal is print a simple test page to verify that everything is in good working order. Seems like a very simple matter almost worth skipping, Don't! This is where I ran in to my troubles.

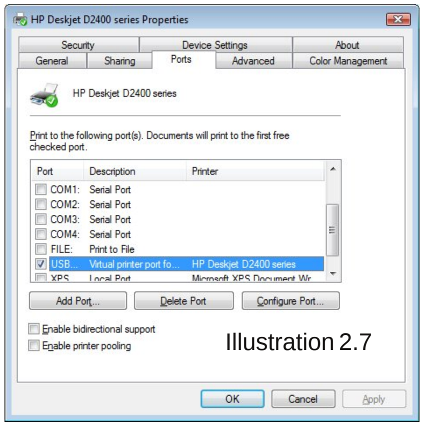


Please select the test pages you want to print.
Note: the photo test page can take a rather long time to get printed and on laser printers with too low memory it can even not come out. In most cases it is enough to print the standard test page.

- [✓] Standard test page
- [ ] Photo test page

Illustration 2.6

I sent a test page to my printer and the Vista computer showed that only 64k of the job was sent to the printer and it hung there. The following is what I had to do in order to get it working.



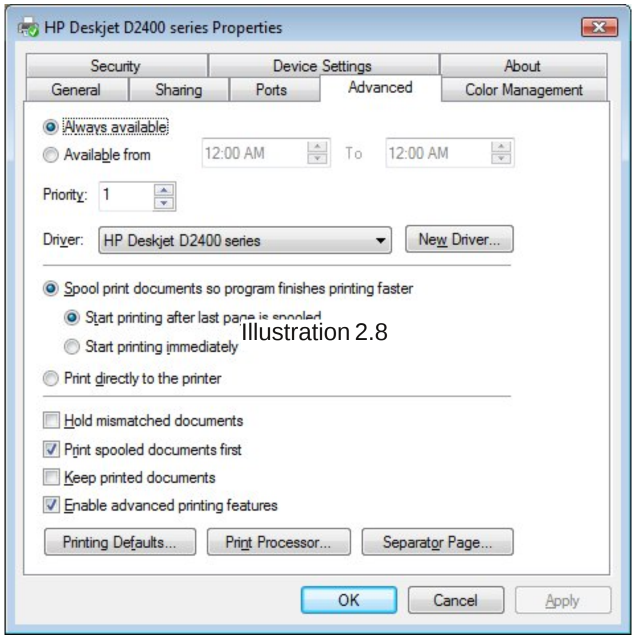
Print to the following port(s). Documents will print to the first free checked port.

Port	Description	Printer
[ ] COM1:	Serial Port	
[ ] COM2:	Serial Port	
[ ] COM3:	Serial Port	
[ ] COM4:	Serial Port	
[ ] FILE:	Print to File	
[✓] USB...	Virtual printer port fo...	HP Deskjet D2400 series
[ ] XPS	Local Port	Microsoft XPS Document Wr

- [ ] Enable bidirectional support
- [ ] Enable printer pooling

Illustration 2.7

Go back through the steps illustrated in 1.5 - 1.7. When the printer properties dialog box is open, click on the "Ports" tab, then **un**check "Enable bidirectional support" (illustration 2.7)



HP Deskjet D2400 series Properties

Tabs: Security | Device Settings | About | General | Sharing | Ports | Advanced | Color Management

- (●) Always available
- ( ) Available from 12:00 AM To 12:00 AM

Priority: 1

Driver: HP Deskjet D2400 series — New Driver...

- (●) Spool print documents so program finishes printing faster
  - (●) Start printing after last page is spooled
  - ( ) Start printing immediately
- ( ) Print directly to the printer

- [ ] Hold mismatched documents
- [✓] Print spooled documents first
- [ ] Keep printed documents
- [✓] Enable advanced printing features

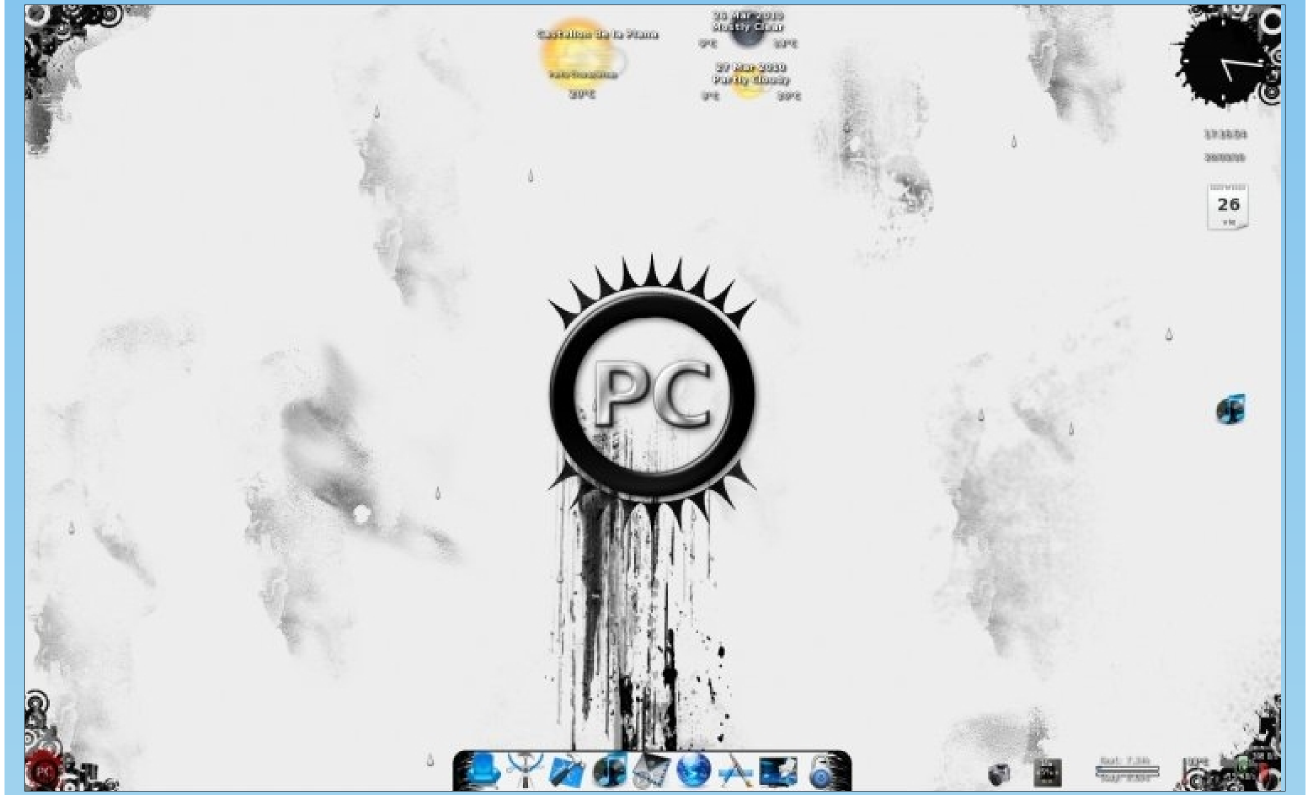
Illustration 2.8

Next, choose the "Advanced" tab then change the radio button to "Start printing after last page is spooled"

Click the "Apply" button.

Once I had that done I was able to print flawlessly.

Screenshot Showcase



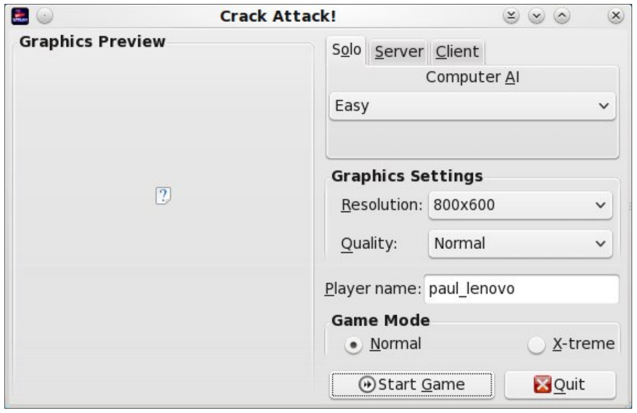
Posted by Agust on March 26, 2010, running E17 beta 1

Game Zone: Crack Attack!

by Paul Arnote (parnote)

To the uninitiated, Crack Attack!, which is available in the PCLinuxOS repository, seems more like a twist between Bejeweled and Tetris. Game author Daniel Nelson asserts that it is based off of the Super Nintendo classic game, Tetris Attack. Which ever it's most alike, game play is quick and very addictive.

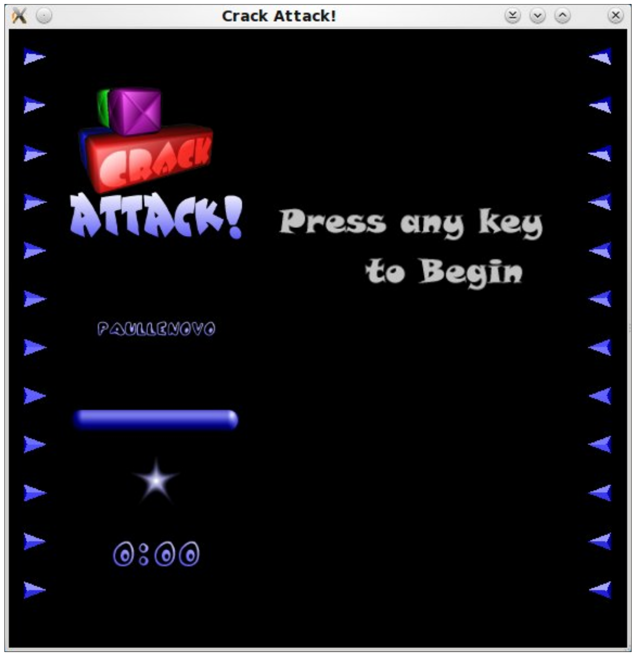
The game requires that your video card support OpenGL 3D graphics rendering. Without it, the game is unplayable, according to the game's web site, at http://aluminumangel.org/attack/. Fortunately, most computers sold in the last four to five years support OpenGL graphics. Unfortunately, this means that I cannot play this game on my old Thinkpad T23 laptop, with a Pentium III processor and 8 MB of video ram.



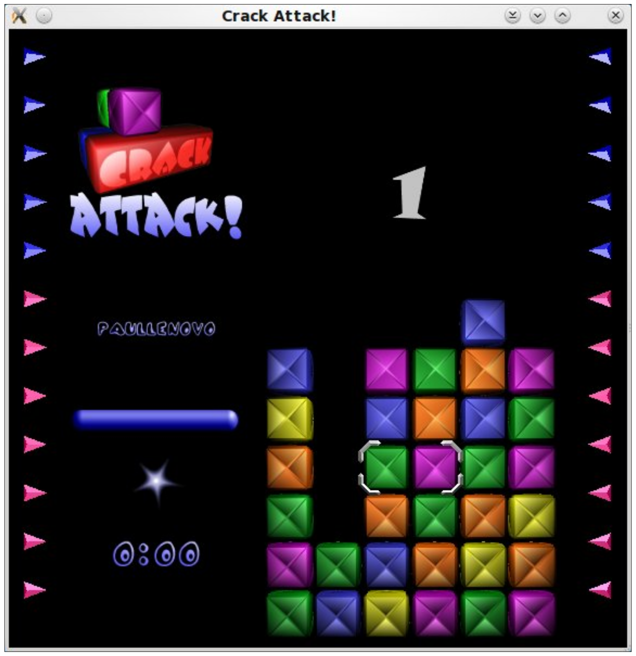
Computer AI: Easy

Graphics Settings
Resolution: 800x600
Quality: Normal
Player name: paul_lenovo
Game Mode: [✓ Normal] X-treme

When the game is first launched, you will be given the chance to make some selections on game play. Under the "Solo" tab, you can set the difficulty level, under "Computer AI." Since I am not much of a gamer, I chose the "Easy" level of game play. In the middle of the window, on the right, you can set the size of the display for the game, as well as the quality of the rendering. You can also set the name of the player here. I allowed mine to default to the same name as my computer login name. You can also set the "Game Mode" to either "Normal" or "X-treme." Perhaps it's due to my game play skill (or serious lack thereof), but I cannot see any difference between the two game modes.

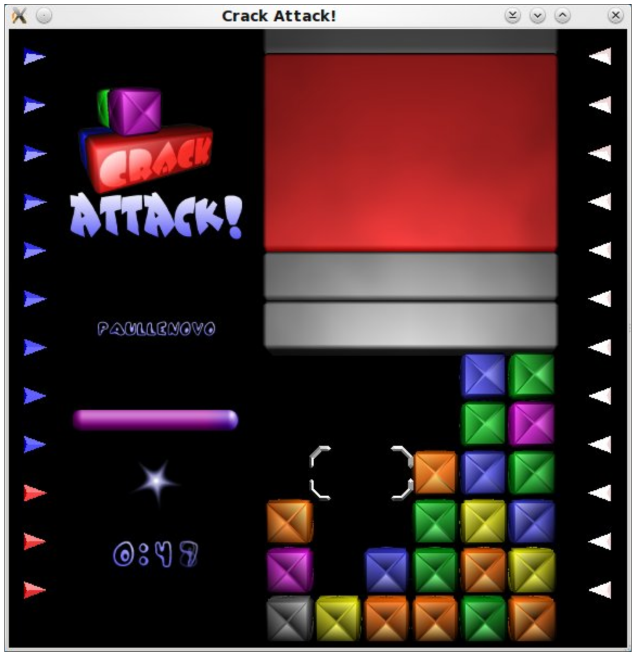


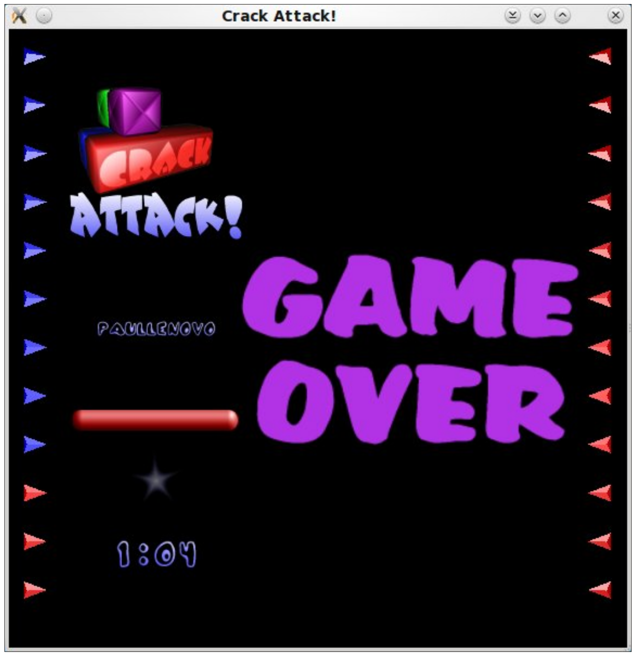
Once you hit the "Start Game" button, you are presented with the above screen. Press any key to start game play.



Using the arrow keys and the space bar, you manipulate the colored blocks to get three blocks of the same color in a row, either vertically or horizontally. The arrow keys will move your selector around the screen, and pressing the space bar will reverse the position of the two blocks.

When you get three colored blocks lined up, they will disappear from your screen. As you clear the screen of colored blocks, "trash" will fall down from the top, while your colored blocks rise up from the bottom of the screen. As game play progresses, the colored blocks rise from the bottom of the screen at an increasing rate of speed. You can "blast" the trash at the top of your blocks by eliminating a row of same-





colored blocks that come into contact with the trash. The "trash blocks" then become normal colored blocks, which you continue to remove in the normal fashion.

Game play for the current match is over once there is no more room at the top of the screen to fit your blocks into. You will either receive the "Game Over" screen, or the "You Win!" screen, depending on whether or not you won the match. I only received the "You Win!" screen once.

There is also an option to play others over a network. I only tested the "Solo" play aspect of the game.

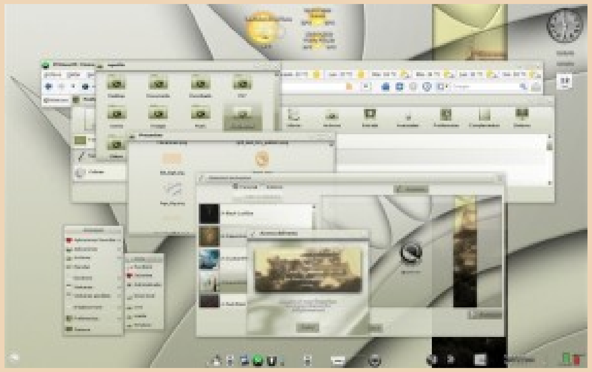
Although rather simple, game play is challenging and very addictive. If you are anything like me, you will find yourself playing match after match. The best I've ever been able to do is to last 1:27 before getting the "Game Over" end-of-game screen. I'm sure you'll find Crack Attack! as addictive as I do.

Want to keep up on the latest that's going on with PCLinuxOS?

Follow PCLinuxOS on Twitter!

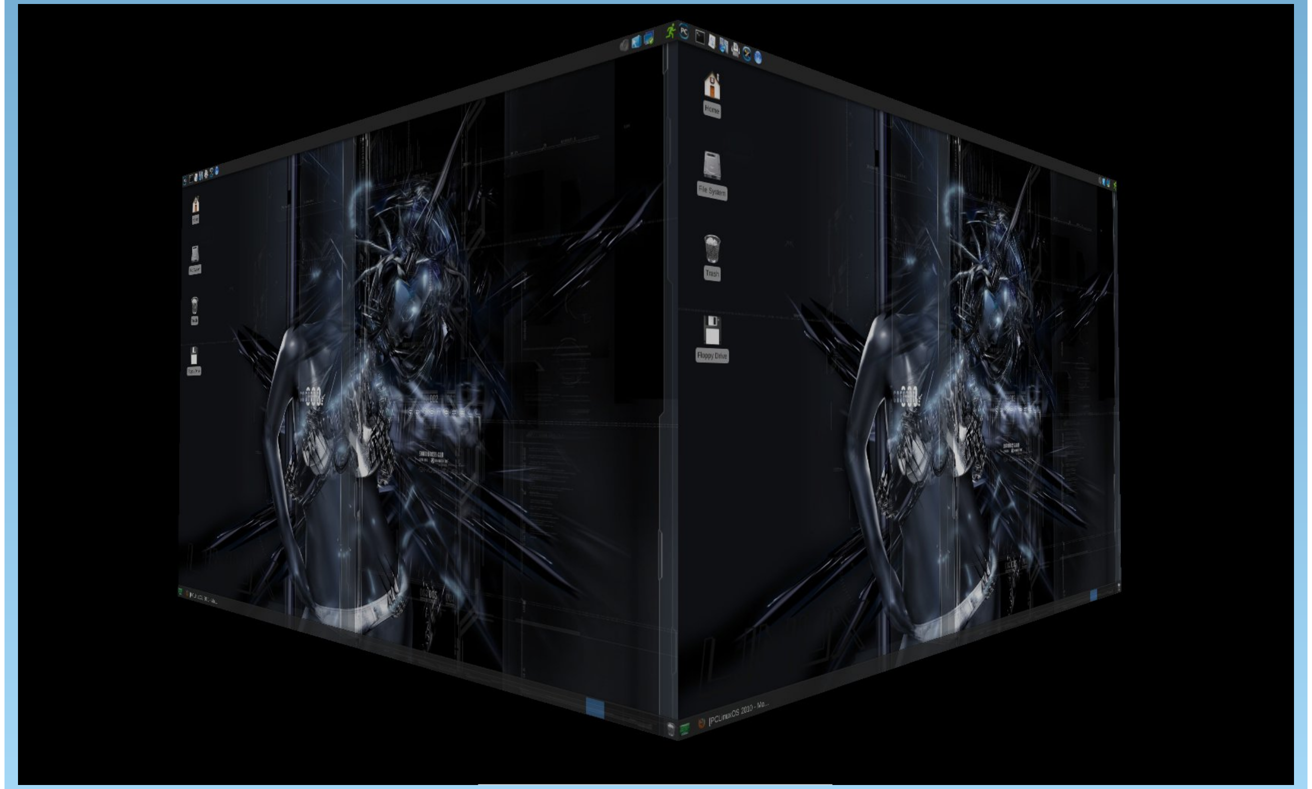
http://twitter.com/iluvpclinuxos

PCLinuxOS Enlightenment e17 ISO



Coming soon!

Screenshot Showcase



Posted by coffeetime, March 3, 2010, running Phoenix (XFCE)

Testimonial: The Road To KDE 4

by Harley Babcock-Doyle (Chomp)

I've been using PCLinuxOS for ... ummm ... I'm not exactly sure actually, but I believe I made the switch to the distro hopper stopper in early '09. I've never bothered to praise the efforts of Tex and the devs of PCLinuxOS for the excellent work they did with KDE 3.5 for some reason. When Tex stopped supporting KDE 3.5, I switched to KDE 4 shortly after. I had a mixed reaction to this. I was used to the excellent implementation and super stable KDE 3.5 in PCLinuxOS, and found KDE 4 to be rather slow and a bit of a pain to use because of this. Additionally, serious problems (for me) occured in Firefox under KDE 4. Firefox was so unresponsive and slow that I was forced, for the first time since Firefox 1.0, to switch to a different browser. I made a forum topic about it, and shortly afterwards, Tex released an update that improved Firefox enough for me to be able to switch back to using it as my full time browser. It was still rather unresponsive and laggy, but usable enough it didn't frustrate me to the point of not using it. I wasn't upset about all of this, however. Patience, I told myself. You can't expect the devs to make an implementation as solid as they did with KDE 3.5 overnight. It will take them time to work out the kinks, and besides that, you have to work with what you are provided with and try to make the best of it.

Copyright © 2009, Mark Szorady. All Rights Reserved.

A little while later, a somewhat sizable update for KDE 4 occurred. Performance was improved slightly and users came to the forum to praise the increased performance. I held back from commenting, as I felt the DE was still not as snappy and responsive as it should be. Patience young padawan, I repeated to myself. You know perfectly well you have a good idea of how these things work, and the difficulty that can be experienced when working with software on a development/packaging end.

It had been a couple of days since I had checked for an update, so about 45 minutes ago I opened Synaptic (I prefer manually checking for updates for some reason), and found another sizable update to KDE 4 was available. After I finished updating, I logged out to restart X, and then back in, not really expecting anything significant. I re-opened all the programs I had opened before in my previous session (Firefox, Chromium, Kmess, Xchat, Ktorrent, Synaptic, and Amarok), then went back to doing what I was doing before. But something was different. I got a weird feeling in my stomach. Everything was just so darn responsive and snappy! "What the heck is this?" I cried out loud (I have a bad habit of talking to myself, even when others are present). I quickly started playing with apps I knew to be what I considered uncharacteristically slow. The quick response by the apps to my actions both surprised and delighted me. Then it came time for the ultimate test. Just how much more responsive was Firefox now? The first thing I did was my scroll test. I navigated to the PCLinuxOS forums, as I knew it to be slow when scrolling. My scrolling test consists of rolling the middle click as rapidly as possible back and forth. The response I was used to was about 5 to 10 seconds of the page continuing to scroll up and down after I physically stopped scrolling. Not this time. As soon as I stopped scrolling so did the page. "Waaaaaaaaaaaaaahhhhhhhhhh!!!" I screamed out loud (good thing my upstairs neighbour is used to hearing me talk to myself). I furiously started clicking on the folders in my bookmarks toolbar, as I found before the update that it would take about 3 to 4 clicks before the folder would actually display the content it held. This wasn't consistent but it did happen on average, I'd estimate, 50% of the time. Every single click on the folders in my bookmarks toolbar immediately opened displaying the juicy contents they contained. WOW!!!!

So here is my long overdue kudos to Tex and the rest of the PCLinuxOS team. Thank you for all of your hard work. You guys are responsible for what I consider to be the best distro out there and I appreciate your efforts.

PCLinuxOS Beautification 2009 Onwards

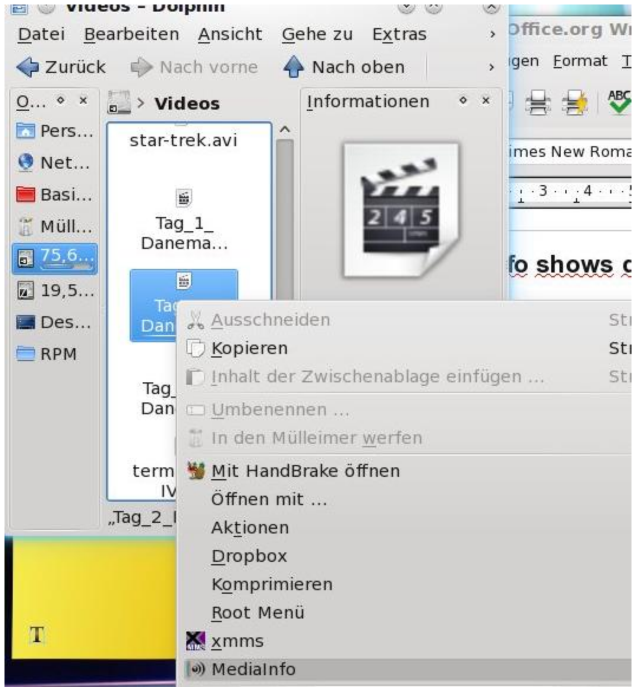
MediaInfo Reveals Information About Your Media Files

by Daniel Meiß-Wilhelm (Leiche)

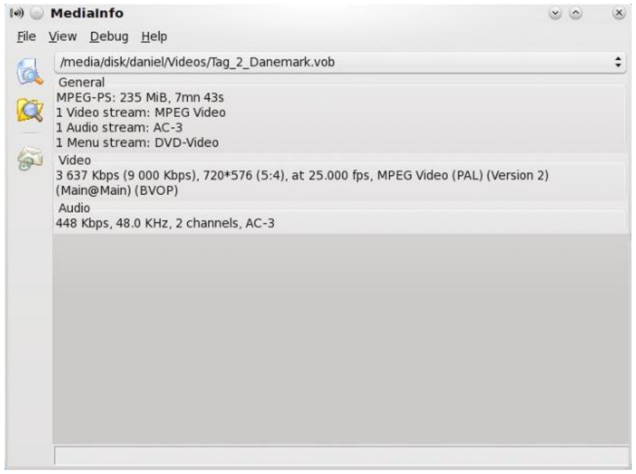
You have a video error, and you have no idea why? A codec is needed, but which one? Or, you would like to create a dvd video, but you don't know if the media file is compatible.

Try MediaInfo. It can give the meta-information you need.

MediaInfo is available in our repositories, and you can install it with Synaptic. At version 0.7.28, it adds a right click menu/service-menu for kde4. You can analyze any media file with the right click menu.



If you click on MediaInfo in the menu, you get a new window with a lot of details about the media file.



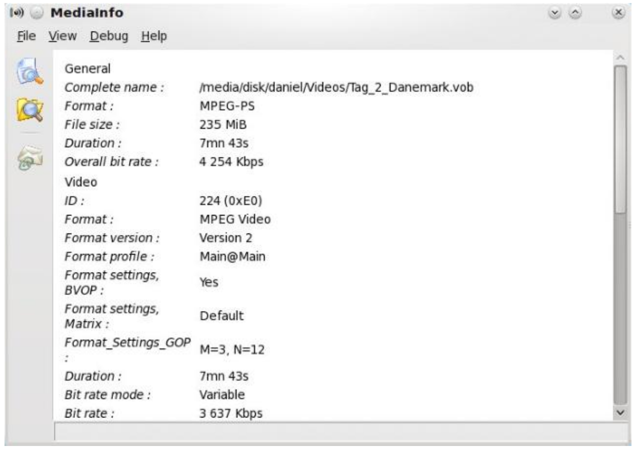
```
/media/disk/daniel/Videos/Tag_2_Danemark.vob
General
MPEG-PS: 235 MiB, 7mn 43s
1 Video stream: MPEG Video
1 Audio stream: AC-3
1 Menu stream: DVD-Video
Video
3 637 Kbps (9 000 Kbps), 720*576 (5:4), at 25.000 fps, MPEG Video (PAL) (Version 2)
(Main@Main) (BVOP)
Audio
448 Kbps, 48.0 KHz, 2 channels, AC-3
```

By the way, it's a simple window, and it can only open media files or directories. When you use a directory, please note that it will take a little time to analyze the files inside the directory.

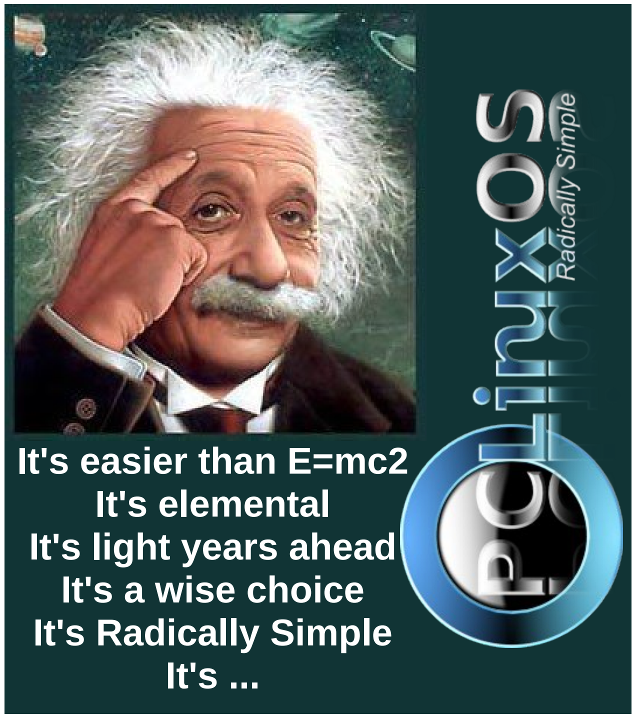
To see what I mean, change the window output to html. To do this, go to View > html.

And you will get the last graphic in the next column.

You can start MediaInfo from the KMenu > Multimedia > Video or Audio.



```
General
Complete name :          /media/disk/daniel/Videos/Tag_2_Danemark.vob
Format :                 MPEG-PS
File size :              235 MiB
Duration :               7mn 43s
Overall bit rate :       4 254 Kbps
Video
ID :                     224 (0xE0)
Format :                 MPEG Video
Format version :         Version 2
Format profile :         Main@Main
Format settings,
BVOP :                   Yes
Format settings,
Matrix :                 Default
Format_Settings_GOP
:                        M=3, N=12
Duration :               7mn 43s
Bit rate mode :          Variable
Bit rate :               3 637 Kbps
```



Computer Languages A to Z: J

by Gary L. Ratliff Sr. (eronstuc)

In the first article in this series on APL, we introduced Dr. Kenneth Iverson, who developed the APL notation in 1957. He was hired by IBM in 1960 to develop this notation for use on the IBM360. Much later, he went to work for Sharpe in the development of its APL systems. He received the Turing Award, and later in 1989 began to develop the J language, along with Roger Hui and Arthur Whitney. It appears to be a variant of APL which does not use any symbols not found on an ASCII keyboard.

Like APL, J is proprietary. It may be obtained for free from Jsoftware, (http://www.jsoftware.com). It is available for many platforms. I have installed it on both the 2009-1 Kde and 2009-2 Gnome versions of PCLINUXOS. It relies upon Java being installed. There is a 64bit version. However, as this could not find any useful Java64 system, it would not function. Unlike most installs, you will not want to install this as root. Doing so would place the code in /root, and thus make it necessary to run this as root all the time. This is not a good practice. To obtain this download (the 32 bit version), download j602a_linux32.sh from the jsoftware web site.

Installing and setting up J

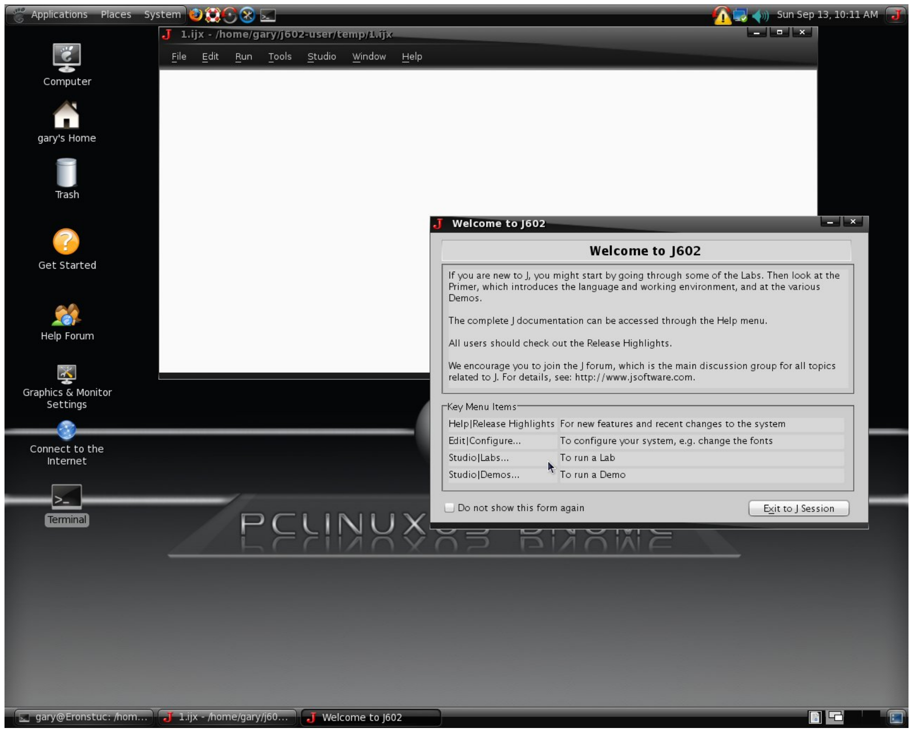
Installation is very simple. When the installation began, you should have selected the save option. Simply open a terminal and move to the location of the script. Then execute this command to run the script:

```
sh j602a_linux32.sh -install
```

You will soon be presented with a menu of commands to use to run J. However, we are going to create simple scripts to change the path for us. The jconsole command would be run to run the J interpreter, once you have learned enough about the J language to actually run it. The jwd command contains a terminal with several menu items which will aid you in exploring the J system. Open either kwrite or gedit, depending on which version of PCLINUXOS you are in. Next, create the scripts for jwd, jbreak and jconsole:

```
#!/bin/sh
cd /home/gary/j602/bin
./jwd

#!/bin/sh
cd /home/gary/j602/bin
./jconsole

#!/bin/sh
cd /home/gary/j602/bin
./jbreak
```

Of course, you will substitute your own name in the path to the file. These scripts simply allow the execution of the programs without typing in the path. Now the directory /home/gary/bin should be found on the PATH, and may be verified by executing **echo $PATH**. If this is not the case, then it may be set there with the command **PATH=/home/gary/bin:$PATH**, which would establish this as the first item in the search PATH. Once you have created the files, you will need to make them executable. This is done by executing **chmod +x jwd, chmod +x jconsole**, and **chmod +x jbreak**. Finally, you will move them into the /home/gary/bin directory with **mv jwd /home/gary/bin/**. Once all the files have been created and moved into the /home/gary/bin folder, we are ready to launch J for the first time. Open a terminal and enter the command **jwd**.



Welcome to J602

If you are new to J, you might start by going through some of the Labs. Then look at the Primer, which introduces the language and working environment, and at the various Demos.

The complete J documentation can be accessed through the Help menu.

All users should check out the Release Highlights.

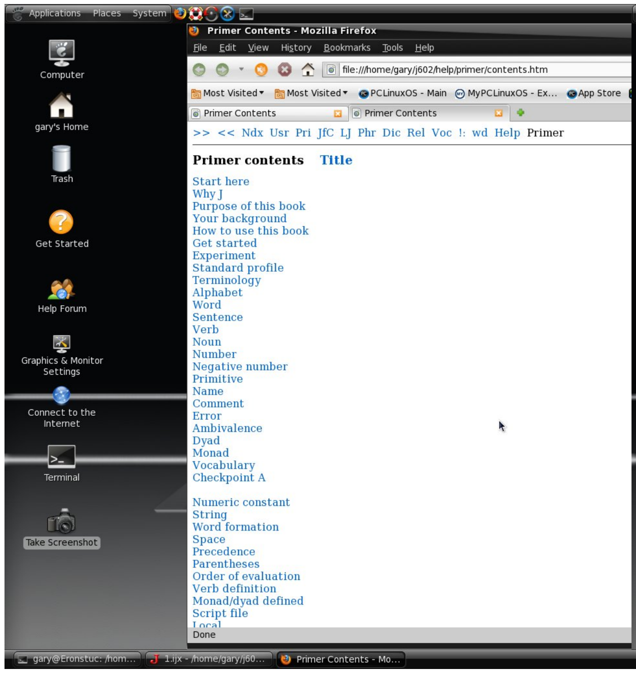
We encourage you to join the J forum, which is the main discussion group for all topics related to J. For details, see: http://www.jsoftware.com.

Key Menu Items

Help|Release Highlights	For new features and recent changes to the system
Edit|Configure...	To configure your system, e.g. change the fonts
Studio|Labs...	To run a Lab
Studio|Demos...	To run a Demo

Do not show this form again

The first thing you will see is a welcome page and an ijx page. The welcome page contains information for new users. Your most likely choice, once you have clicked the Exit to J Session button, would be to select the Help menu.



File Edit View History Bookmarks Tools Help

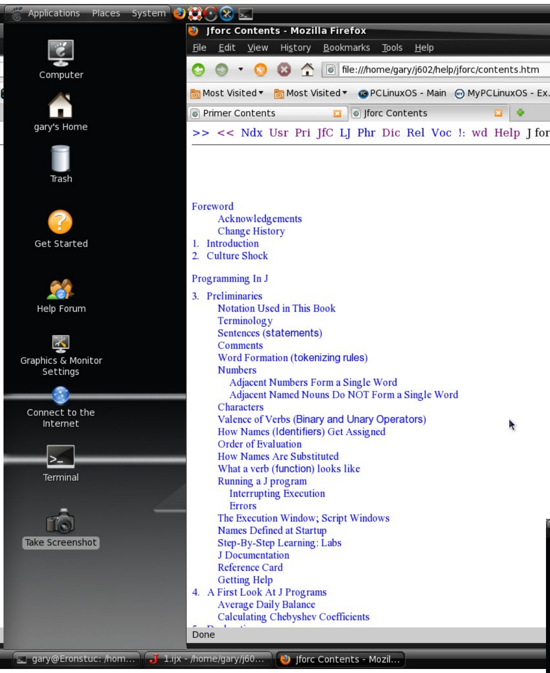
file:///home/gary/j602/help/primer/contents.htm

>> << Ndx Usr Pri JfC LJ Phr Dic Rel Voc !: wd Help Primer

Primer contents Title

Start here
Why J
Purpose of this book
Your background
How to use this book
Get started
Experiment
Standard profile
Terminology
Alphabet
Word
Sentence
Verb
Noun
Number
Negative number
Primitive
Name
Comment
Error
Ambivalence
Dyad
Monad
Vocabulary
Checkpoint A

Numeric constant
String
Word formation
Space
Precedence
Parentheses
Order of evaluation
Verb definition
Monad/dyad defined
Script file



file:///home/gary/j602/help/jforc/contents.htm

>> << Ndx Usr Pri JfC LJ Phr Dic Rel Voc !: wd Help J for C Programmers

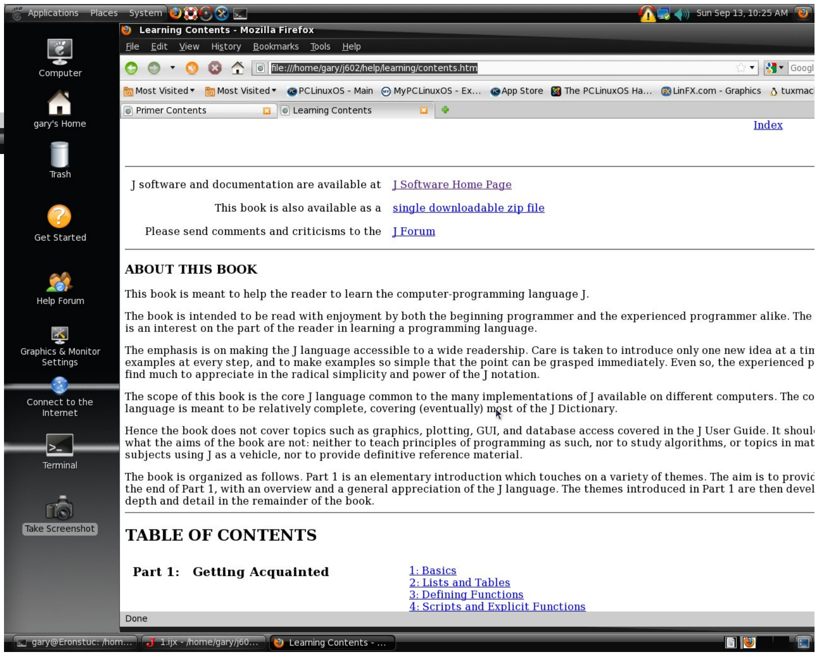
Contents

The primer is one source of easy lessons to introduce you to J. Another introduction is in the Labs. However, these get into the use of J much faster. Also, many of the labs will require acquiring other packages from J software. When you read the prices on the J software page, should you desire the source code, you'll quickly discover that these begin at $10,000 and continue to $400,000. You quickly begin to really appreciate what value open source software has. And you also come to appreciate those who devote their time and effort into providing you with Linux.

Another source is the book J For C Programmers. This book is written for those who already use the C language. It states early on that the more a person has used C, the more foreign they are likely to find the J language. It states that most computer languages deal with numbers as single items, and must create programming loops to deal with extracting items from lists or larger structures, one number at a time. The J language and APL, for example, deal with lists and arrays as the basic unit of computations. My first encounter with APL, and the experience of duplicating what had taken four pages in BASIC in only 67 APL characters, was just such an awakening.

And still another helpful book is "Learning J." by Roger Stokes. This provides an introduction to all the features of the dictionary, which are common to all ports of the language. This refrains from treatment of some items as load 'plot' or load 'debug,' which are covered in the labs.

Again, you will need to learn the language on your own. This article simply points out that there is a very powerful computer language named J, which may be obtained and tried. If you enjoyed APL, you should find J equally exciting.



file:///home/gary/j602/help/learning/contents.htm

Index

J software and documentation are available at J Software Home Page

This book is also available as a single downloadable zip file

Please send comments and criticisms to the J Forum

ABOUT THIS BOOK

This book is meant to help the reader to learn the computer-programming language J.

The book is intended to be read with enjoyment by both the beginning programmer and the experienced programmer alike. The
is an interest on the part of the reader in learning a programming language.

The emphasis is on making the J language accessible to a wide readership. Care is taken to introduce only one new idea at a tim
examples at every step, and to make examples so simple that the point can be grasped immediately. Even so, the experienced p
find much to appreciate in the radical simplicity and power of the J notation.

The scope of this book is the core J language common to the many implementations of J available on different computers. The co
language is meant to be relatively complete, covering (eventually) most of the J Dictionary.

Hence the book does not cover topics such as graphics, plotting, GUI, and database access covered in the J User Guide. It shoul
what the aims of the book are not: neither to teach principles of programming as such, nor to study algorithms, or topics in mat
subjects using J as a vehicle, nor to provide definitive reference material.

The book is organized as follows. Part 1 is an elementary introduction which touches on a variety of themes. The aim is to provic
the end of Part 1, with an overview and a general appreciation of the J language. The themes introduced in Part 1 are then devel
depth and detail in the remainder of the book.

TABLE OF CONTENTS

Forum Foibles: Forum Fools

1. This would be a great thread for the magazine forum foibles. Although, if this is publicized we're in trouble.

2. I would reply, but I've chosen to ignore you all, so I don't know what you said.

3. The future is already here for a few among us .. my vote for those who are fortunate to live the future.

4. I'm not certified but I'm definitly certifiable.

5. I am really confused. After that I am lost.

6. I don't trust anyone or anything on the Internet.

7. Now all these points are now cleared up.

8. I think I'll "Re-Boot".

9. Don't know how many times I have to keep posting this but here goes again for the 100th time.

10. Well at least there is someone that is smart in the group!

FORUM FOOLS are my delight
I can spot them a mile away
Always posting day and night
Don't think before they say

What's this you ask some mistake
Surely no FOOLS exist
Sorry to give you a double take
But I really must insist

Here's a quiz for you to solve
I'm sure none you will miss
Use your brain with much resolve
To find what FOOL said this

ms_meme

Answers on page 48

11. I've spent endless hours trying to sort out a silly little problem. Why is it that the simple solutions NEVER present themselves FIRST?

12. The best of us started off asking stupid questions.

13. Love it, got to admit though, some of us have done some real dumb things.

14. Man's quest for answers is the shadow of his ignorance.

15. Does that mean that is going to rain cheese from the moon?

16. Yep! Start off with coffee, and the golf balls will be much much better.

17. No problems here.

18. Sniff, sniff. Do I smell smoke in here? Who just lit up a cig to go with their coffee?

19. I figured out how to create a link!

20. Love knows no boundaries.

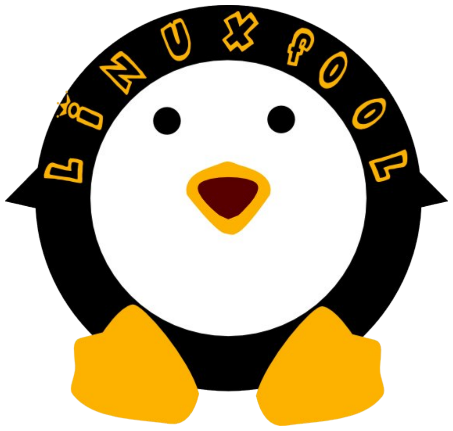
ms_meme's Nook: Linux Fool

What kind of fool was I that took so long to try
An OS like Linux Oh Little Tux I don't know why
What kind of 'puter had I an empty shell
Where all my files and folders were discontent to dwell

What kind of user was this who was so remiss
Almost missed the best system that ever exists
I'm so glad I found PCLOS
Now I know this fool's been blessed

MP3

OGG



What kind of fool was I that never ever knew
'Twas time to try Linux Oh Little Tux I was over due
What kind of 'puter had I with one desktop
When would this long boot up and scanning ever stop

What kind of clown was I my days were full of strife
Scanning for viruses was all I ever knew of life
I'm going to use Linux till I don't give a ram
And maybe then I'll know what kind of fool I am

Testimonial: Loving The Beta!

by Mark Miles (BubbaBlues)

I've been a PCLinuxOS lover for close to two years now. When I tried it the first time with the incredibly stable and simple KDE 3.5, I was in hog heaven. I absolutely loved it far better than any other distro I'd tried. Almost all of the rest were using that hideous KDE 4, which was just unusable for me. I do a lot of music and video burning and trans-coding, but I also like tweaking and changing looks of things (themes, icons, etc.). Anyway, KDE4 just didn't work, but 3.5 was literally flawless.

Then one day the devs made the dreaded announcement that PCLinuxOS is switching to KDE 4.something!! "OMG how can they do this?" I exclaimed with a few colorful adjectives not repeatable here. Well, I eventually came to understand that 3.5 is no longer supported and the change had to come, like it or not. Being the laid back patient man that I am (not), I made the comment that KDE 4 is and always will be nothing but a huge bumbling mess, or something to that effect.

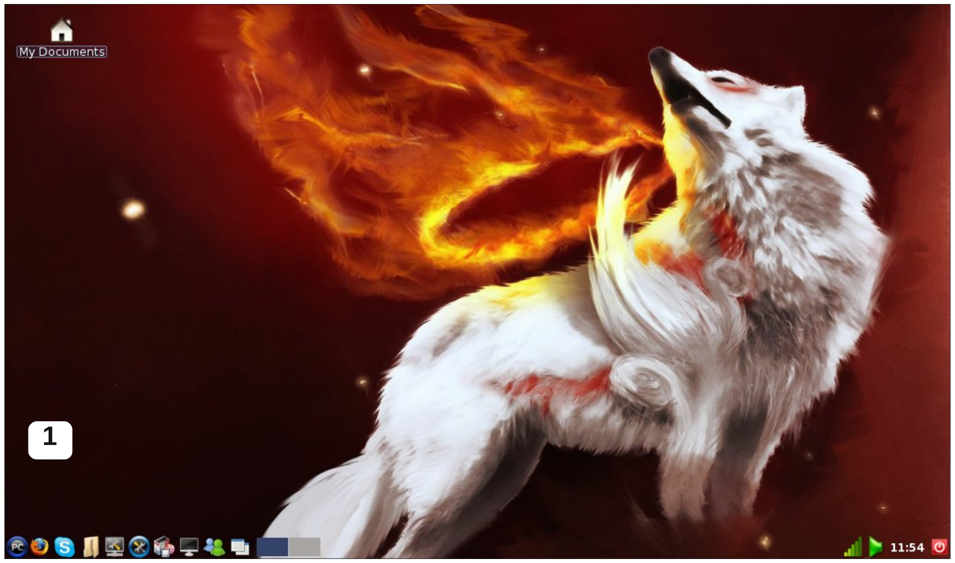
Well here I am running 2010.1, the first beta release and LOVING it! I've been running it through the mill too, opening several apps at a time, transcoding, burning, changing themes, moving things around on the main panel, and I haven't managed to break it. And it's fast — real fast. The response time for many apps is less than a second, when before the same thing would take three or four seconds. Tex and the dev crew have kept up their standard of excellence with this ultra fine distro, and it isn't even the final release yet.

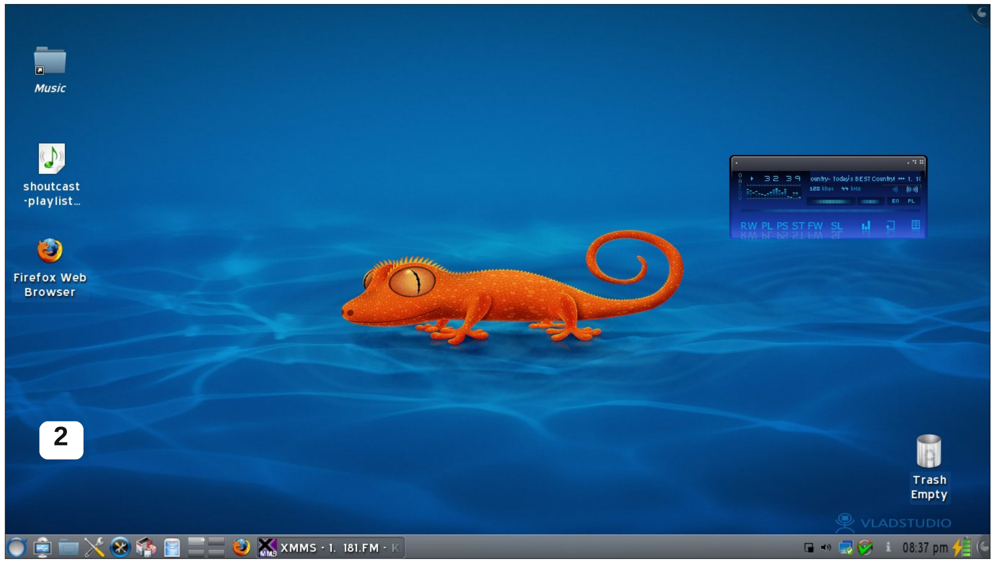
Now I can say that this one isn't perfect, though close. The only glitch that has showed up for me is the device notification window, which has some behavioral problems occasionally, and only slightly then. I'm sure there are other little things that would eventually show up, but like I said, this is the first beta release. I have no doubt whatsoever that the final release of PCLOS will be nothing short of perfection. Thank you Texstar and the rest for your hard work and standard of excellence that make us love our computers. Now I'm going to go eat a nice dish of crow.

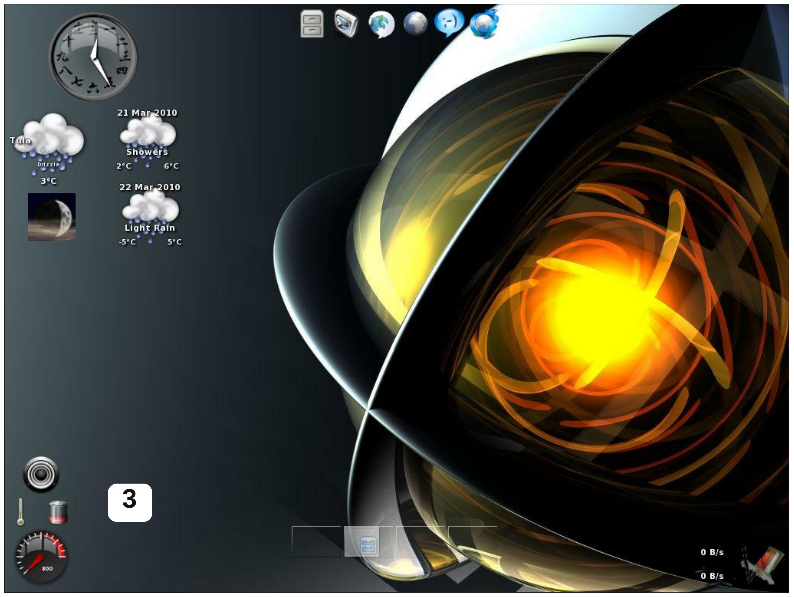
Copyright © 2009, Mark Szorady.
All Rights Reserved.

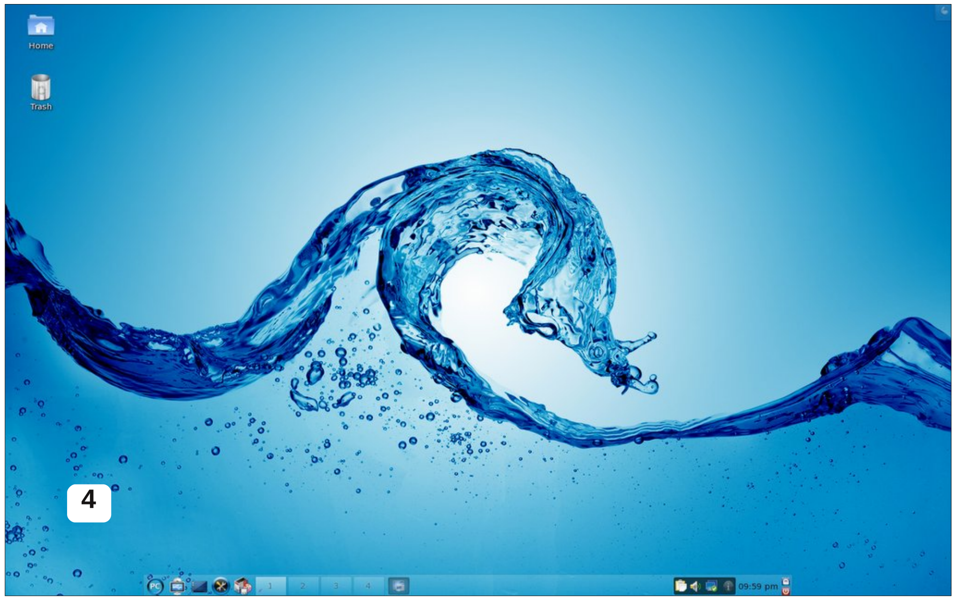
Answers to Forum Foibles *1. gseaman 2. joble 3. muungwana 4. maddogF16 5. Johnboy 6. ElCuervo 7. Hootiegibbon 8. Wildman 9. Texstar 10. YouCanToo 11. jaydot 12. davecs 13. coolbreeze 14. Archie 15. T6 16. Neal 17. coffeetime 18. linuxera 19. TheChief 20. Crow*

More Screenshot Showcase









1. Posted by coffeetime, March 4, 2010, running PCLXDE (LXDE).

2. Posted by bones113, March 1, 2010, running KDE 4.

3. Posted by drhadidy, March 21, 2010, running e17.

4. Posted by mmesantos1, March 13, 2010, running KDE 4.